aws

Panduan Developer

Amazon Elastic Container Service

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Elastic Container Service: Panduan Developer

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon ECS? 1
Terminologi dan komponen 1
Fitur 3
Penyediaan 3
Harga 4
Layanan terkait 4
Memulai 6
Penyiapan 6
AWS Management Console 6
Mendaftar untuk Akun AWS 7
Buat pengguna dengan akses administratif 7
Buat virtual private cloud 9
Membuat grup keamanan 10
Buat kredensil untuk terhubung ke instans Anda EC2 14
Instal AWS CLI 14
Langkah selanjutnya untuk menggunakan Amazon ECS 15
Membuat gambar kontainer 16
Prasyarat 16
Membuat citra Docker 18
Dorong gambar Anda ke Amazon Elastic Container Registry 20
Bersihkan 22
Langkah selanjutnya 22
Pelajari cara membuat tugas Linux untuk jenis peluncuran Fargate 23
Prasyarat 23
Langkah 1: Buat cluster 24
Langkah 2: Buat definisi tugas 25
Langkah 3: Buat layanan 26
Langkah 4: Melihat layanan Anda 26
Langkah 5: Bersihkan 27
Pelajari cara membuat tugas Windows untuk jenis peluncuran Fargate 27
Prasyarat 27
Langkah 1: Buat cluster 28
Langkah 2: Daftarkan ketentuan tugas Windows 29
Langkah 3: Buat layanan dengan ketentuan tugas Anda 30

Langkah 4: Melihat layanan Anda 31
Langkah 5: Bersihkan 31
Pelajari cara membuat tugas Windows untuk jenis EC2 peluncuran 32
Prasyarat 32
Langkah 1: Buat cluster 33
Langkah 2: Daftarkan definisi tugas 35
Langkah 3: Buat Layanan 36
Langkah 4: Lihat Layanan Anda 37
Langkah 5: Bersihkan 37
Menggunakan AWS CDK 38
Langkah 1: Siapkan AWS CDK proyek Anda 39
Langkah 2: Gunakan AWS CDK untuk mendefinisikan server web kontainer di Fargate 41
Langkah 3: Uji server web 49
Langkah 4: Membersihkan 49
Langkah selanjutnya 50
Membuat sumber daya menggunakan AWS Copilot CLI 50
Memasang AWS Copilot CLI 51
Menyebarkan contoh aplikasi Amazon ECS menggunakan Copilot CLI AWS 60
Amazon ECS dengan AWS CloudFormation 62
Contoh AWS CloudFormation template 63
Ketentuan tugas 64
Klaster 76
Layanan 89
Peran IAM untuk Amazon ECS 146
Membuat sumber daya Amazon ECS menggunakan konsol AWS CloudFormation 148
Prasyarat 149
Langkah 1: Buat template tumpukan 149
Langkah 2: Buat tumpukan untuk sumber daya Amazon ECS 155
Langkah 3: Verifikasi 156
Langkah 4: Bersihkan Sumber Daya 156
Membuat sumber daya Amazon ECS menggunakan AWS CLI perintah untuk AWS CloudFormation 157
Prasyarat 157
Langkah 1: Buat tumpukan 157
Langkah 2: Verifikasi pembuatan sumber daya 163
Langkah 3: Bersihkan 164

Praktik terbaik 166
AWS Fargate 170
Panduan 170
Penyedia kapasitas 171
Ketentuan tugas 171
Versi platform 171
Penyeimbangan beban layanan 172
Metrik penggunaan 172
Pertimbangan keamanan kapan harus menggunakan tipe peluncuran Fargate 173
Praktik terbaik keamanan Fargate 173
Gunakan AWS KMS untuk mengenkripsi penyimpanan sementara untuk Fargate 173
Kemampuan SYS_PTRACE untuk penelusuran syscall kernel dengan Fargate 174
Gunakan Amazon GuardDuty dengan Fargate Runtime Monitoring 174
Pertimbangan keamanan Fargate 175
Versi platform Fargate 176
..... 176
Migrasi ke platform Linux versi 1.4.0 177
Log perubahan versi Platform Linux 178
Penghentian versi platform Linux 181
Log perubahan versi platform Windows 183
Wadah Windows tentang pertimbangan Fargate untuk Amazon ECS 184
Penyimpanan sementara tugas Fargate 185
Versi platform kontainer Fargate Linux 185
Fargate versi platform kontainer Windows 187
Kunci yang dikelola pelanggan untuk AWS Fargate penyimpanan sementara 187
Tugas pensiun dan pemeliharaan 200
Ikhtisar pemberitahuan pensiun tugas 202
Bisakah saya memilih keluar dari pensiun tugas? 205
Bisakah saya mendapatkan pemberitahuan pensiun tugas melalui AWS layanan lain? 205
Bisakah saya mengubah pensiun tugas setelah dijadwalkan? 206
Bagaimana Amazon ECS menangani tugas yang merupakan bagian dari layanan? 206
Bisakah Amazon ECS secara otomatis menangani tugas mandiri? 206
Bersiaplah untuk pensiun tugas AWS Fargate di Amazon ECS 206
AWS Wilayah Fargate 209
Kontainer Linux di AWS Fargate 209
Wadah Windows di AWS Fargate 211

Arsitek solusi Anda untuk Amazon ECS 214
Kapasitas 215
Titik akhir layanan 216
Jaringan 216
Akses fitur 217
Peran IAM 218
Pencatatan log 218
Jenis peluncuran 219
Jenis peluncuran Fargate 219
EC2 jenis peluncuran 223
Jenis peluncuran eksternal 226
Menggunakan titik akhir tumpukan ganda 227
Menggunakan titik akhir tumpukan ganda (AWS CLI) 228
Menggunakan titik akhir tumpukan ganda (AWS SDKs) 229
Menggunakan titik akhir tumpukan ganda dari API REST 229
Aplikasi di subnet bersama, Local Zones, dan Wavelength Zones 230
Subnet bersama 230
Zona Lokal 232
Wavelength Zones 232
Layanan Kontainer Elastis Amazon aktif AWS Outposts 233
Pertimbangan 233
Prasyarat 234
Ikhtisar pembuatan cluster pada AWS Outposts 234
Praktik terbaik jaringan 236
Connect aplikasi ke internet 237
Praktik terbaik untuk menerima koneksi masuk ke Amazon ECS 241
Praktik terbaik untuk menghubungkan ke AWS layanan 246
Praktik terbaik untuk menghubungkan layanan 249
Praktik terbaik untuk layanan jaringan di seluruh AWS akun dan VPCs 254
AWS layanan untuk pemecahan masalah jaringan 255
Optimalkan waktu peluncuran tugas 256
Beroperasi pada skala 258
Kuota layanan Amazon ECS dan batas pembatasan API 258
Menangani masalah pelambatan 263
Praktik terbaik penskalaan otomatis dan manajemen kapasitas 267
Menentukan ukuran tugas 267

Mengoptimalkan penskalaan otomatis layanan 269
Kapasitas dan ketersediaan 275
Kapasitas cluster 280
Memilih ukuran tugas Fargate 280
Mempercepat penskalaan otomatis cluster 281
Akses fitur dengan pengaturan akun 283
Nama Sumber Daya Amazon (ARNs) dan IDs 284
Lini masa format ARN dan ID sumber daya 286
Wawasan Kontainer 287
AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140) 289
Otorisasi penandaan 290
Menandai garis waktu otorisasi 291
AWS Fargate tugas pensiun waktu tunggu 292
Meningkatkan antarmuka jaringan instance kontainer Linux 293
Pemantauan Runtime (GuardDuty integrasi Amazon) 294
IPv6 VPC tumpukan ganda 294
Mode driver log default 295
Melihat pengaturan akun menggunakan konsol 296
Mengubah pengaturan akun 296
Mengembalikan ke pengaturan akun default 298
Mengelola pengaturan akun menggunakan AWS CLI 298
Peran IAM untuk Amazon ECS 300
Ketentuan tugas 304
Status definisi tugas 305
Sumber daya Amazon ECS yang dapat memblokir penghapusan 306
Arsitek aplikasi Anda 307
Praktik terbaik untuk gambar kontainer 308
Praktik terbaik untuk ukuran tugas 310
Jaringan tugas untuk jenis EC2 peluncuran 312
Jaringan tugas untuk tipe peluncuran Fargate 324
Opsi penyimpanan untuk tugas 328
Mengelola ruang memori swap kontainer 415
Perbedaan definisi tugas untuk jenis peluncuran Fargate 417
Perbedaan definisi tugas untuk EC2 instance yang menjalankan Windows 426
Membuat definisi tugas menggunakan konsol 427
Validasi JSON 428

AWS CloudFormation tumpukan 428
Prosedur 428
Memperbarui definisi tugas menggunakan konsol 459
Validasi JSON 459
Prosedur 460
Membatalkan pendaftaran revisi definisi tugas menggunakan konsol 460
AWS CloudFormation tumpukan 428
Prosedur 462
Menghapus revisi definisi tugas menggunakan konsol 462
Sumber daya Amazon ECS yang dapat memblokir penghapusan 306
Prosedur 463
Kasus penggunaan definisi tugas 464
Definisi tugas untuk beban kerja GPU 464
Definisi tugas untuk beban kerja trancoding video 475
Definisi tugas untuk beban kerja pembelajaran mesin AWS Neuron 489
Definisi tugas untuk contoh pembelajaran mendalam 497
Definisi tugas untuk beban kerja ARM 64-bit 500
Kirim log ke CloudWatch 502
Kirim log ke AWS layanan atau AWS Partner 506
Menggunakan gambar AWS non-kontainer 519
Mulai ulang wadah individu dalam tugas 522
Meneruskan data sensitif ke wadah 525
Parameter definisi tugas untuk jenis peluncuran Fargate 548
Rangkaian 548
Jenis peluncuran 549
Peran tugas 549
Peran eksekusi tugas 550
Mode jaringan 550
Platform runtime 550
Ukuran tugas 551
Definisi kontainer 555
Nama akselerator Elastic Inference 600
Konfigurasi proxy 601
Volume 603
Tanda 610
Parameter ketentuan tugas lainnya 611

Parameter definisi tugas untuk jenis EC2 peluncuran ... 613
Rangkaian ... 614
Jenis peluncuran ... 614
Peran tugas ... 614
Peran eksekusi tugas ... 615
Mode jaringan ... 615
Platform runtime ... 617
Ukuran tugas ... 618
Definisi kontainer ... 619
Nama akselerator Elastic Inference ... 668
Kendala penempatan tugas ... 669
Konfigurasi proxy ... 669
Volume ... 672
Tanda ... 678
Parameter ketentuan tugas lainnya ... 679
Templat ketentuan tugas ... 681
Contoh ketentuan tugas ... 692
Peladen Web ... 693
`splunk`driver log ... 695
`fluentd`driver log ... 696
`gelf`driver log ... 696
Beban kerja pada instance eksternal ... 697
Peran IAM definisi gambar dan tugas Amazon ECR ... 699
Entrypoint dengan perintah ... 699
Dependensi kontainer ... 700
Volume dalam definisi tugas ... 702
Ketentuan tugas sampel Windows ... 702
Klaster ... 704
Penyedia kapasitas ... 705
Cluster untuk Fargate ... 706
Pemberitahuan penghentian Fargate Spot ... 708
Membuat cluster untuk tipe peluncuran Fargate ... 709
Penyedia kapasitas untuk jenis EC2 peluncuran ... 712
EC2 keamanan contoh kontainer ... 715
Membuat cluster untuk jenis EC2 peluncuran Amazon ... 715
Auto Scaling klaster ... 721

Contoh EC2 wadah Amazon 756
Cluster untuk tipe peluncuran eksternal 898
Sistem pengoperasian dan arsitektur sistem yang didukung 899
Pertimbangan 900
Membuat cluster untuk tipe peluncuran Eksternal 903
Mendaftarkan instans eksternal ke kluster Amazon ECS 906
Membatalkan pendaftaran instans eksternal 911
Memperbarui AWS Systems Manager agen dan agen kontainer Amazon ECS 917
Memperbarui klaster 922
Menghapus klaster 924
Membatalkan pendaftaran instance kontainer 925
Prosedur 926
Pengurasan instans kontainer 926
Menguras perilaku untuk layanan 927
Menguras perilaku untuk tugas mandiri 928
Prosedur 928
Contoh EC2 Kontainer Amazon 929
Siklus hidup 930
Support Docker 931
AMI Amazon ECS yang dioptimalkan 931
Informasi tambahan 932
Konfigurasi agen kontainer 932
Memasang agen kontainer Amazon ECS 935
Parameter konfigurasi log agen kontainer 942
Mengkonfigurasi instance kontainer untuk gambar Docker pribadi 943
Bersihkan tugas dan gambar 948
Jadwalkan wadah Anda 951
Opsi komputasi 953
Siklus hidup tugas 954
Status siklus hidup 955
Bagaimana Amazon ECS menempatkan tugas pada instans kontainer 957
EC2 jenis peluncuran 957
Jenis peluncuran Fargate 958
Gunakan strategi untuk menentukan penempatan tugas 959
Tugas terkait kelompok 964
Tentukan instance kontainer mana yang digunakan untuk tugas 965

Tugas mandiri 976
Alur kerja tugas 976
Menjalankan aplikasi sebagai tugas 977
Menggunakan Amazon EventBridge Scheduler untuk menjadwalkan tugas 988
Menghentikan tugas 996
Layanan 997
Opsi komputasi infrastruktur 998
Penskalaan otomatis layanan 999
Penyeimbangan beban layanan 999
Layanan interkoneksi 1001
Pengontrol dan strategi penyebaran layanan 1002
Penyebaran layanan 1153
Revisi layanan 1163
Rebalancing Zona Ketersediaan 1171
Gunakan load balancing untuk mendistribusikan lalu lintas layanan 1181
Penskalaan otomatis layanan 1195
Layanan interkoneksi 1236
Perlindungan skala tugas 1296
Injeksi kesalahan dengan Amazon ECS dan Fargate 1305
Perbarui parameter layanan Amazon ECS 1314
Menghapus layanan 1344
Migrasikan ARN singkat layanan 1345
Logika throttle layanan 1351
Parameter ketentuan layanan 1353
Templat definisi layanan 1380
Pemberian tag pada sumber daya 1388
Bagaimana sumber daya ditandai 1389
Menandai sumber daya pada pembuatan 1392
Pembatasan 1393
Tag yang dikelola Amazon ECS 1393
Gunakan tag untuk penagihan 1394
Menambahkan tag ke sumber daya 1395
Menambahkan tag ke instance container 1397
Contoh kontainer eksternal 1399
Laporan Penggunaan 1400
Biaya dan penggunaan tingkat tugas 1401

Pemantauan 1403
Praktik terbaik untuk memantau Amazon ECS 1404
Alat-alat pemantauan 1404
Alat Otomatis 1404
Alat Manual 1406
Pantau Amazon ECS menggunakan CloudWatch 1407
Pertimbangan 1408
Metrik-metrik yang direkomendasikan 1408
Melihat metrik Amazon ECS 1409
Metrik Amazon ECS CloudWatch 1410
AWS Fargate metrik penggunaan 1439
Metrik reservasi klaster Amazon ECS 1441
Metrik pemanfaatan klaster Amazon ECS 1443
Metrik pemanfaatan layanan Amazon ECS 1445
Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge 1449
Peristiwa Amazon ECS 1450
Menangani acara 1474
Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan 1478
Tentukan kesehatan tugas menggunakan pemeriksaan kesehatan kontainer 1479
Bagaimana kesehatan tugas ditentukan 1481
Pemeriksaan kesehatan dan pemutusan agen 1482
Lihat kesehatan kontainer 1482
Pantau kesehatan instans kontainer Amazon ECS 1483
Masalah kesehatan instans kontainer 1484
Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi 1484
Izin IAM yang diperlukan untuk AWS Distro untuk integrasi dengan OpenTelemetry AWS X-Ray 1484
Menentukan AWS Distro untuk OpenTelemetry sespan untuk AWS X-Ray integrasi dalam definisi tugas Anda 1486
Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi 1488
Mengekspor metrik aplikasi ke Amazon CloudWatch 1488
Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus 1492
Log panggilan Amazon ECS API menggunakan AWS CloudTrail 1497
Acara manajemen Amazon ECS di CloudTrail 1498
Contoh acara Amazon ECS 1499

Melihat CloudWatch Log Live Tail untuk layanan dan tugas Amazon ECS 1500
Izin yang diperlukan 1501
Prosedur 1501
AWS CLI referensi 1502
Pantau beban kerja menggunakan metadata 1502
Variabel-variabel lingkungan 1503
File metadata kontainer 1504
Metadata tugas tersedia untuk tugas Amazon ECS EC2 1511
Metadata tugas tersedia untuk tugas di Fargate 1554
Introspeksi wadah 1577
Identifikasi perilaku yang tidak sah menggunakan Runtime Monitoring 1580
Bagaimana Runtime Monitoring bekerja dengan Amazon ECS 1581
Pertimbangan 1581
Pemanfaatan Sumber Daya 1582
Runtime Monitoring untuk beban kerja Fargate 1582
Runtime Monitoring untuk beban kerja EC2 1586
Pemecahan Masalah Runtime Monitoring 1590
Pantau wadah Amazon ECS dengan ECS Exec 1594
Pertimbangan 1595
Prasyarat 1597
Arsitektur 1598
Menggunakan ECS Exec 1598
Logging menggunakan ECS Exec 1601
Menggunakan kebijakan IAM untuk membatasi akses ke ECS Exec 1605
Rekomendasi Compute Optimizer 1609
Rekomendasi ukuran tugas untuk Fargate 1609
Pemecahan Masalah 1610
Mengatasi kesalahan tugas yang dihentikan 1613
Pembaruan pesan kesalahan tugas yang dihentikan 1613
Melihat kesalahan tugas yang dihentikan 1618
Pesan kesalahan tugas yang dihentikan 1620
Memverifikasi konektivitas tugas 1640
Melihat permintaan peran IAM 1645
Melihat pesan acara layanan 1646
Pesan acara layanan Amazon ECS 1647
Layanan Amazon ECS pesan acara tidak sehat 1658

Amazon ECS Availability Zone layanan penyeimbangan kembali pesan acara layanan 1659
Memecahkan masalah penyeimbang beban layanan di Amazon ECS 1661
Pemecahan masalah penskalaan otomatis layanan di Amazon ECS 1663
Memecahkan masalah definisi tugas kesalahan CPU atau memori yang tidak valid 1664
Melihat log agen kontainer 1666
Mengumpulkan log kontainer dengan kolektor log Amazon ECS 1668
Introspeksi agen 1670
Diagnostik Docker di Amazon ECS 1672
Daftar kontainer Docker di Amazon ECS 1673
Lihat Log Docker di Amazon ECS 1674
Periksa Kontainer Docker di Amazon ECS 1675
Mengonfigurasi keluaran verbose dari daemon Docker di Amazon ECS 1676
Memecahkan masalah Docker di Amazon `API error (500): devmapper` ECS 1677
Memecahkan masalah ECS Exec 1679
Verifikasi menggunakan Pemeriksa Exec 1679
Terjadi kesalahan saat memanggil `execute-command` 1679
Memecahkan masalah Amazon ECS Anywhere 1680
Permasalahan registrasi instans eksternal 1680
Masalah jaringan instans eksternal 1681
Masalah menjalankan tugas 1681
AWS Fargate kuota pelambatan 1682
Melambatkan `RunTask` API di Fargate 1682
Menyesuaikan kuota tingkat di Fargate 1683
Alasan kegagalan API 1683
Keamanan 1694
Identity and Access Management 1695
Audiens 1695
Mengautentikasi dengan identitas 1696
Mengelola akses menggunakan kebijakan 1700
Bagaimana Amazon Elastic Container Service bekerja dengan IAM 1702
Contoh kebijakan berbasis identitas 1714
AWS kebijakan terkelola untuk Amazon ECS 1726
Menggunakan peran terkait layanan 1741
Peran IAM untuk Amazon ECS 1745
Izin diperlukan untuk konsol Amazon ECS 1809
Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green 1818

Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS 1819
Memberi tanda pada sumber daya saat sumber daya dibuat 1821
Pemecahan Masalah 1825
Praktik terbaik IAM 1827
Pembuatan Log dan Pemantauan 1831
Validasi kepatuhan 1833
Praktik terbaik kepatuhan dan keamanan 1834
AWS Fargate Kepatuhan FIPS-140 1836
AWS Fargate Pertimbangan FIPS-140 1836
Gunakan FIPS di Fargate 1836
Gunakan CloudTrail untuk audit Fargate FIPS-140 1837
Keamanan Infrastruktur 1838
Antarmuka VPC endpoint (AWS PrivateLink) 1839
Model tanggung jawab bersama 1846
Jenis peluncuran Fargate 1846
EC2 jenis peluncuran 1847
Praktik terbaik keamanan jaringan 1848
Enkripsi bergerak 1848
Jaringan tugas 1850
AWS PrivateLink dan Amazon ECS 1850
Pengaturan agen kontainer 1851
Rekomendasi keamanan jaringan 1852
Praktik terbaik keamanan tugas dan kontainer 1853
Buat minimal atau gunakan gambar distroless 1854
Pindai gambar Anda untuk kerentanan 1855
Hapus izin khusus dari gambar Anda 1856
Buat satu set gambar yang dikuratori 1856
Pindai paket aplikasi dan pustaka untuk mencari kerentanan 1855
Lakukan analisis kode statis 1857
Jalankan kontainer sebagai pengguna non-root 1857
Gunakan sistem file root read-only 1858
Konfigurasikan tugas dengan batas CPU dan Memori (Amazon EC2) 1858
Gunakan tag yang tidak dapat diubah dengan Amazon ECR 1859
Hindari menjalankan kontainer sebagai hak istimewa (Amazon EC2) 1859
Hapus kemampuan Linux yang tidak perlu dari wadah 1859

Gunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi gambar yang didorong ke Amazon ECR 1860
Tutorial 1861
Membuat tugas Linux untuk tipe peluncuran Fargate dengan AWS CLI 1863
Prasyarat 1864
Langkah 1: Buat Klaster 1865
Langkah 2: Daftarkan Definisi Tugas Linux 1866
Langkah 3: Cantumkan Ketentuan tugas 1867
Langkah 4: Buat Layanan 1868
Langkah 5: Cantumkan Layanan 1868
Langkah 6: Jelaskan Layanan yang Berjalan 1869
Langkah 7: Uji 1871
Langkah 8: Bersihkan 1875
Membuat tugas Windows untuk tipe peluncuran Fargate dengan AWS CLI 1875
Prasyarat 1876
Langkah 1: Buat Klaster 1877
Langkah 2: Daftarkan Definisi Tugas Windows 1878
Langkah 3: Daftar definisi tugas 1879
Langkah 4: Buat layanan 1879
Langkah 5: Daftar layanan 1880
Langkah 6: Jelaskan Layanan yang Berjalan 1880
Langkah 7: Bersihkan 1883
Membuat tugas untuk jenis EC2 peluncuran dengan AWS CLI 1883
Prasyarat 1884
Membuat klaster 1885
Luncurkan instance kontainer dengan Amazon ECS AMI 1885
Daftar contoh kontainer 1888
Jelaskan contoh kontainer Anda 1888
Daftarkan definisi tugas 1891
Daftar definisi tugas 1893
Membuat layanan 1894
Daftar layanan 1894
Jelaskan layanannya 1895
Jelaskan tugas yang sedang berjalan 1896
Uji server web 1897
Pembersihan sumber daya 1898

Mengonfigurasi Amazon ECS untuk mendengarkan acara Acara CloudWatch 1899
Prasyarat: Atur klaster pengujian 1899
Langkah 1: Buat fungsi Lambda 1900
Langkah 2: Mendaftarkan aturan peristiwa 1900
Langkah 3: Membuat sebuah penetapan tugas 1901
Langkah 4: Uji aturan Anda 1903
Mengirim peringatan Amazon Simple Notification Service untuk acara yang dihentikan tugas 1903
Prasyarat: Atur klaster pengujian 1903
Prasyarat: Konfigurasikan izin untuk Amazon SNS 1904
Langkah 1: Buat dan berlangganan ke topik Amazon SNS 1904
Langkah 2: Mendaftarkan aturan peristiwa 1905
Langkah 3: Uji aturan Anda 1906
Menggabungkan pesan log multiline atau stack-trace 1907
Izin IAM yang diperlukan 1907
Tentukan kapan harus menggunakan pengaturan log multiline 1909
Parse dan concatenate pilihan 1910
Menerapkan Fluent Bit pada wadah Windows 1929
Prasyarat 1932
Langkah 1: Buat peran akses IAM 1932
Langkah 2: Buat instance penampung Amazon ECS Windows 1933
Langkah 3: Konfigurasikan Bit Lancar 1935
Langkah 4: Daftarkan definisi tugas Windows Fluent Bit yang merutekan log CloudWatch . 1937
Langkah 5: Jalankan definisi `ecs-windows-fluent-bit` tugas sebagai layanan Amazon ECS menggunakan strategi penjadwalan daemon 1938
Langkah 6: Daftarkan definisi tugas Windows yang menghasilkan log 1940
Langkah 7: Jalankan definisi `windows-app-task` tugas 1941
Langkah 8: Verifikasi log CloudWatch 1942
Langkah 9: Membersihkan 1943
Menggunakan gMSA untuk EC2 Linux wadah 1944
Pertimbangan 1944
Prasyarat 1945
Pengaturan 1946
File CredSpec 1953
Menggunakan gMSA untuk Linux wadah di Fargate 1955
Pertimbangan 1955
Prasyarat 1955

Pengaturan 1956
File CredSpec 1959
Menggunakan wadah Windows dengan domainless gMSA menggunakan AWS CLI 1960
Prasyarat 1961
Langkah 1: Buat dan konfigurasikan gMSA akun di Layanan Domain Direktori Aktif (AD DS) 1963
Langkah 2: Unggah Kredensial ke Secrets Manager 1965
Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA 1966
Langkah 4: Unggah CredSpec ke Amazon S3 1967
Langkah 5: (Opsional) Buat cluster Amazon ECS 1968
Langkah 6: Buat peran IAM untuk instance kontainer 1968
Langkah 7: Buat peran eksekusi tugas khusus 1969
Langkah 8: Buat peran tugas untuk Amazon ECS Exec 1970
Langkah 9: Daftarkan definisi tugas 1972
Langkah 10: Daftarkan contoh wadah Windows 1973
Langkah 11: Verifikasi contoh kontainer 1974
Langkah 12: Jalankan tugas Windows 1975
Langkah 13: Verifikasi wadah memiliki gMSA kredensil 1976
Langkah 14: Bersihkan 1976
Debugging 1978
Pelajari cara menggunakan g MSAs untuk wadah EC2 Windows 1979
Pertimbangan 1979
Prasyarat 1980
Pengaturan 1981
Menggunakan Image Builder untuk membuat Amazon ECS yang disesuaikan yang dioptimalkan AMIs 1987
Menggunakan gambar ARN dengan infrastruktur sebagai kode (IAc) 1989
Menggunakan gambar ARN dengan AWS CloudFormation 1991
Menggunakan gambar ARN dengan Terraform 1993
Menggunakan AWS Deep Learning Containers 1993
Kuota layanan 1994
Mengelola kuota layanan Anda di AWS Management Console 1994
Menangani kuota layanan dan batas pembatasan API 1996
Penyeimbang Beban Elastis 259
Antarmuka jaringan elastis 261

AWS Cloud Map 263
Referensi API Amazon ECS 2002
Riwayat dokumen 2003
..... mmli

Apa itu Layanan Kontainer Elastis Amazon?

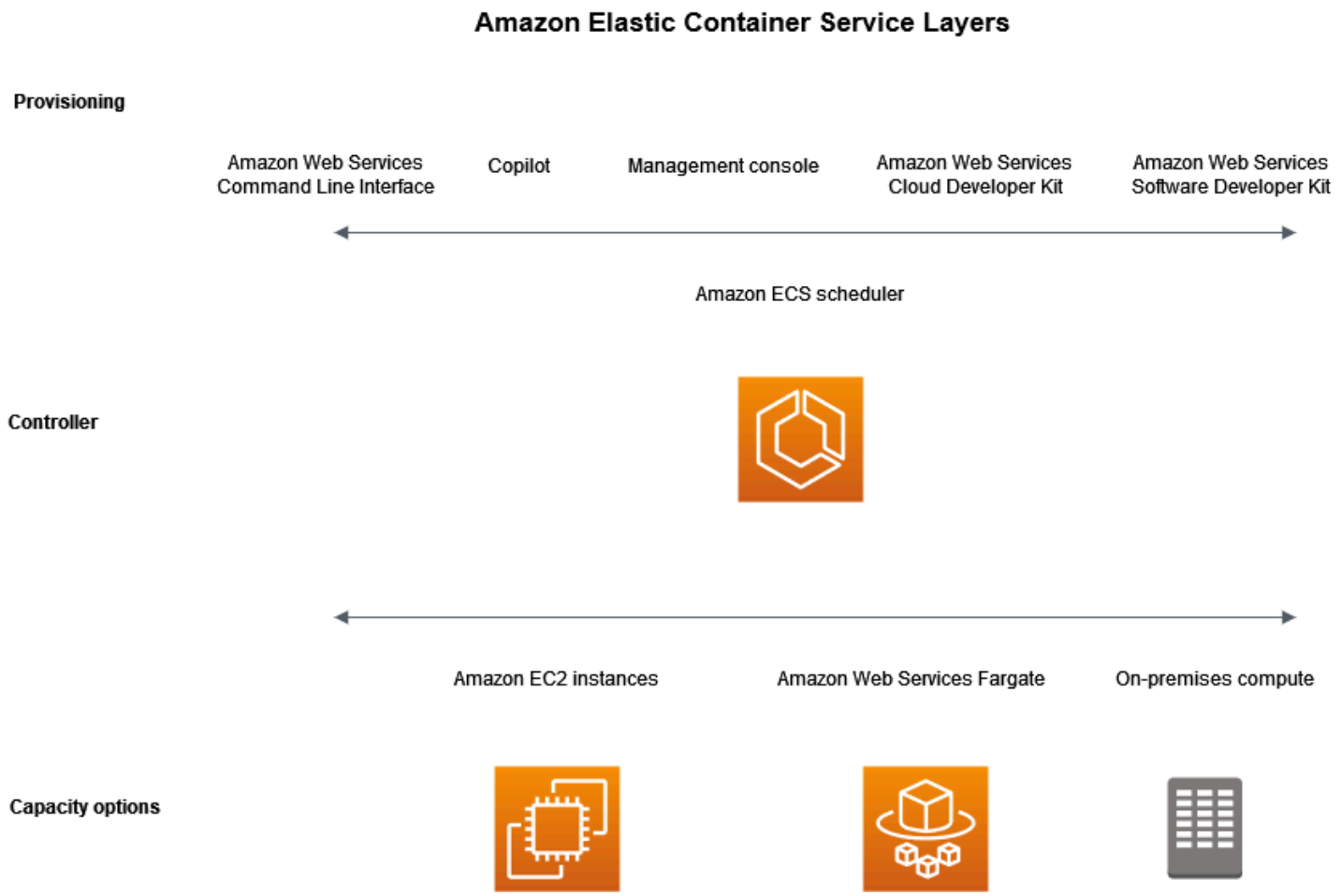
Amazon Elastic Container Service (Amazon ECS) adalah layanan orkestrasi kontainer terkelola penuh yang membantu Anda menerapkan, mengelola, dan menskalakan aplikasi kontainer dengan mudah. Sebagai layanan yang dikelola sepenuhnya, Amazon ECS hadir dengan AWS konfigurasi dan praktik terbaik operasional bawaan. Ini terintegrasi dengan kedua AWS alat, seperti Amazon Elastic Container Registry, dan alat pihak ketiga, seperti Docker. Integrasi ini memudahkan tim untuk fokus membangun aplikasi, bukan lingkungan. Anda dapat menjalankan dan menskalakan beban kerja kontainer Anda Wilayah AWS di cloud, dan lokal, tanpa kerumitan mengelola bidang kontrol.

Terminologi dan komponen

Ada tiga lapisan di Amazon ECS:

- Kapasitas - Infrastruktur tempat kontainer Anda berjalan
- Controller - Menyebarkan dan mengelola aplikasi Anda yang berjalan pada kontainer
- Penyediaan - Alat yang dapat Anda gunakan untuk berinteraksi dengan penjadwal untuk menyebarkan dan mengelola aplikasi dan kontainer Anda

Diagram berikut menunjukkan lapisan Amazon ECS.



Kapasitasnya adalah infrastruktur tempat kontainer Anda berjalan. Berikut ini adalah ikhtisar opsi kapasitas:

- EC2 Instans Amazon di cloud AWS

  Anda memilih jenis instans, jumlah instans, dan mengelola kapasitas.

- Tanpa server (AWS Fargate) di awan AWS

  Fargate adalah mesin komputasi tanpa server. pay-as-you-go Dengan Fargate Anda tidak perlu mengelola server, menangani perencanaan kapasitas, atau mengisolasi beban kerja kontainer untuk keamanan.

- Mesin virtual lokal (VM) atau server

  Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke kluster Amazon ECS Anda.

Penjadwal Amazon ECS adalah perangkat lunak yang mengelola aplikasi Anda.

Fitur

Amazon ECS menyediakan fitur tingkat tinggi berikut:

Ketentuan tugas

: Cetak biru untuk aplikasi.

Klaster

: Infrastruktur aplikasi Anda berjalan.

Tugas

: Aplikasi seperti pekerjaan batch yang melakukan pekerjaan, dan kemudian berhenti.

Layanan

: Aplikasi stateless yang berjalan lama.

Pengaturan Akun

: Memungkinkan akses ke fitur.

Auto Scaling Klaster

: Amazon ECS mengelola penskalaan EC2 instans Amazon yang terdaftar ke klaster Anda.

Layanan Auto Scaling

: Amazon ECS meningkatkan atau mengurangi jumlah tugas yang diinginkan dalam layanan Anda secara otomatis.

Penyediaan

Ada beberapa opsi untuk penyediaan Amazon ECS:

- AWS Management Console— Menyediakan antarmuka web yang dapat Anda gunakan untuk mengakses sumber daya Amazon ECS Anda.
- AWS Command Line Interface (AWS CLI) — Menyediakan perintah untuk serangkaian AWS layanan yang luas, termasuk Amazon ECS. Ini didukung di Windows, Mac, dan Linux. Untuk informasi selengkapnya, lihat AWS Command Line Interface.

- AWS SDKs— Menyediakan bahasa khusus APIs dan menangani banyak detail koneksi. Ini termasuk menghitung tanda tangan, menangani percobaan ulang permintaan, dan penanganan kesalahan. Untuk informasi selengkapnya, lihat AWS SDKs.
- AWS CDKMenyediakan kerangka pengembangan perangkat lunak open-source yang dapat Anda gunakan untuk memodelkan dan menyediakan sumber daya aplikasi cloud Anda menggunakan bahasa pemrograman yang sudah dikenal. AWS CDK Menyediakan sumber daya Anda dengan cara yang aman dan berulang. AWS CloudFormation

Harga

Harga Amazon ECS tergantung pada opsi kapasitas yang Anda pilih untuk wadah Anda.

- Harga Amazon ECS - Informasi harga untuk Amazon ECS.
- AWS Fargate harga — Informasi harga untuk Fargate.

Layanan terkait

Layanan untuk digunakan dengan Amazon ECS

Anda dapat menggunakan yang lain Layanan AWS untuk membantu Anda menyebarkan tugas dan layanan Anda di Amazon ECS.

EC2 Auto Scaling Amazon

Membantu memastikan Anda memiliki jumlah EC2 instans Amazon yang benar yang tersedia untuk menangani pemuatan aplikasi Anda.

Amazon CloudWatch

Pantau layanan dan tugas Anda.

Amazon Elastic Container Registry

Simpan dan kelola gambar kontainer.

Elastic Load Balancing

Secara otomatis mendistribusikan lalu lintas layanan masuk.

Amazon GuardDuty

Mendeteksi penggunaan instans dan beban kerja kontainer yang berpotensi tidak sah atau berbahaya.

Pelajari cara membuat dan menggunakan sumber daya Amazon ECS

Panduan berikut memberikan pengantar alat yang tersedia untuk mengakses Amazon ECS dan prosedur pengantar untuk menjalankan kontainer. Dasar-dasar Docker membawa Anda melalui langkah-langkah dasar untuk membuat gambar kontainer Docker dan mengunggahnya ke repositori pribadi Amazon ECR. Panduan memulai memandu Anda menggunakan antarmuka baris perintah AWS Copilot dan AWS Management Console untuk menyelesaikan tugas umum untuk menjalankan wadah Anda di Amazon ECS dan. AWS Fargate

Daftar Isi

- Siapkan untuk menggunakan Amazon ECS
- Membuat gambar kontainer untuk digunakan di Amazon ECS
- Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis EC2 peluncuran
- Membuat sumber daya Amazon ECS menggunakan AWS CDK
- Membuat sumber daya Amazon ECS menggunakan antarmuka baris AWS perintah Copilot

Siapkan untuk menggunakan Amazon ECS

Jika Anda sudah mendaftar untuk Amazon Web Services (AWS) dan telah menggunakan Amazon Elastic Compute Cloud (Amazon EC2), Anda hampir dapat menggunakan Amazon ECS. Proses pengaturan untuk kedua layanan ini serupa. Panduan berikut mempersiapkan Anda untuk meluncurkan cluster Amazon ECS pertama Anda.

Selesaikan tugas-tugas berikut untuk menyiapkan Amazon ECS.

AWS Management Console

AWS Management Console Ini adalah antarmuka berbasis browser untuk mengelola sumber daya Amazon ECS. Konsol ini memberikan gambaran visual tentang layanan, sehingga mudah untuk menjelajahi fitur dan fungsi Amazon ECS tanpa perlu menggunakan alat tambahan. Banyak tutorial dan panduan terkait tersedia yang dapat memandu Anda menggunakan konsol tersebut.

Untuk tutorial panduan menggunakan konsol tersebut, lihat Pelajari cara membuat dan menggunakan sumber daya Amazon ECS.

Saat memulai, banyak pelanggan lebih suka menggunakan konsol karena memberikan umpan balik visual instan tentang apakah tindakan yang mereka ambil berhasil. AWS pelanggan yang akrab dengan AWS Management Console, dapat dengan mudah mengelola sumber daya terkait seperti penyeimbang beban dan instans Amazon EC2 .

Mulailah dengan AWS Management Console.

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimkan email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Buat virtual private cloud

Anda dapat menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meluncurkan AWS sumber daya ke jaringan virtual yang telah Anda tentukan. Kami sangat menyarankan agar Anda meluncurkan instance container Anda di VPC.

Jika Anda memiliki VPC default, Anda dapat melewati bagian ini dan pindah ke tugas berikutnya,. Membuat grup keamanan Untuk menentukan apakah Anda memiliki VPC default, lihat Bekerja dengan VPC default dan subnet default Anda di Panduan Pengguna Amazon VPC. Jika tidak, Anda dapat membuat VPC non-default di akun Anda menggunakan langkah-langkah di bawah ini.

Untuk informasi tentang cara membuat VPC, lihat Membuat VPC di Panduan Pengguna VPC Amazon, dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Sumber daya untuk dibuat	Hanya VPC
Nama	Secara opsional berikan nama untuk VPC Anda.
IPv4 Blok CIDR	IPv4 Masukan manual CIDR Ukuran blok CIDR harus memiliki ukuran antara /16 dan /28.
IPv6 Blok CIDR	Tidak ada IPv6 blok CIDR
Penghunian	Default

Untuk informasi lebih lanjut tentang Amazon VPC, lihat Apa itu Amazon VPC? dalam Panduan Pengguna Amazon VPC.

Membuat grup keamanan

Grup keamanan bertindak sebagai firewall untuk instans kontainer terkait, yang mengontrol lalu lintas masuk dan keluar di tingkat instans kontainer. Anda dapat menambahkan aturan ke grup keamanan yang memungkinkan Anda terhubung ke instans kontainer dari alamat IP Anda menggunakan SSH. Anda juga dapat menambahkan aturan yang mengizinkan akses HTTP dan HTTPS masuk dan keluar dari mana saja. Tambahkan aturan untuk membuka port yang diperlukan oleh tugas Anda. Instans kontainer memerlukan akses jaringan eksternal untuk berkomunikasi dengan titik akhir layanan Amazon ECS.

Jika Anda merencanakan untuk meluncurkan instans kontainer di beberapa Wilayah, Anda perlu membuat grup keamanan di setiap Wilayah. Untuk informasi selengkapnya, lihat Wilayah dan Zona Ketersediaan di Panduan EC2 Pengguna Amazon.

> **Tip**
>
> Anda memerlukan alamat IP publik komputer lokal Anda, yang bisa Anda dapatkan menggunakan layanan. Misalnya, kami menyediakan layanan berikut: http://checkip.amazonaws.com/ atau https://checkip.amazonaws.com/. Untuk menemukan layanan lain yang menyediakan alamat IP Anda, gunakan frasa pencarian "apa alamat IP saya." Jika Anda terhubung melalui penyedia layanan Internet (ISP) atau dari belakang firewall tanpa alamat IP statis, Anda perlu mengetahui rentang alamat IP yang digunakan oleh komputer klien.

Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan untuk EC2 instans Amazon Anda di Panduan EC2 Pengguna Amazon dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Wilayah	Wilayah yang sama di mana Anda membuat key pair Anda.
Nama	Nama yang mudah diingat, seperti ecs-instances-default-cluster.

Opsi	Nilai
VPC	VPC default (ditandai dengan "(default)"). **Note** Jika akun Anda mendukung Amazon EC2 Classic, pilih VPC yang Anda buat di tugas sebelumnya.

Untuk informasi tentang aturan keluar yang akan ditambahkan untuk kasus penggunaan Anda, lihat Aturan grup keamanan untuk kasus penggunaan yang berbeda di Panduan EC2 Pengguna Amazon.

Instans kontainer Amazon ECS tidak memerlukan port masuk untuk dibuka. Namun, Anda mungkin ingin menambahkan aturan SSH sehingga Anda dapat masuk ke instans kontainer dan memeriksa tugas-tugas dengan perintah Docker. Anda juga dapat menambahkan aturan untuk HTTP dan HTTPS jika Anda ingin instans kontainer Anda untuk meng-host tugas yang menjalankan web server. Instans kontainer juga memerlukan akses jaringan eksternal untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Selesaikan langkah-langkah berikut ini untuk menambahkan aturan grup keamanan opsional ini.

Tambahkan tiga aturan masuk berikut ke grup keamanan Anda.Untuk informasi tentang cara membuat grup keamanan, lihat Mengonfigurasi aturan grup keamanan di Panduan Pengguna Amazon EC2 .

Opsi	Nilai
Aturan HTTP	Jenis: HTTP Sumber: Di mana saja (`0.0.0.0/0`) Opsi ini secara otomatis menambahkan blok IPv4

Opsi	Nilai	
	CIDR 0.0.0.0/0 sebagai sumbernya. Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman dalam lingkungan produksi. Dalam produksi, Anda hanya dapat memberikan otorisasi pada alamat IP atau rentang alamat tertentu saja untuk mengakses instans Anda.	
Aturan HTTPS	Jenis: HTTPS Sumber: Di mana saja (`0.0.0.0/0`) Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman dalam lingkunga n produksi. Dalam produksi, Anda hanya dapat memberika n otorisasi pada alamat IP atau rentang alamat tertentu saja untuk mengakses instans Anda.	

Opsi	Nilai
Aturan SSH	Jenis: SSH Sumber: Kustom, tentukan alamat IP publik komputer atau jaringan Anda dalam notasi CIDR. Untuk menentukan alamat IP individu dalam notasi CIDR, tambahkan prefiks perutean `/32`. Misalnya, jika alamat IP Anda adalah `203.0.113.25` , sebutkan `203.0.113.25/32` . Jika perusahaan Anda mengalokasikan alamat-alamat dari sebuah rentang, tentukan keseluruhan rentang, seperti `203.0.113.0/24` . **Important** Demi alasan keamanan, kami tidak menyarankan Anda untuk mengizinkan akses SSH dari semua alamat IP (`0.0.0.0/0`) ke instans Anda, kecuali untuk tujuan pengujian dan hanya untuk waktu yang singkat.

Buat kredensil untuk terhubung ke instans Anda EC2

Untuk Amazon ECS, key pair hanya diperlukan jika Anda berniat menggunakan tipe EC2 peluncuran.

AWS menggunakan kriptografi kunci publik untuk mengamankan informasi login untuk instans Anda. Instans Linux, seperti instans penampung Amazon ECS, tidak memiliki kata sandi untuk digunakan untuk akses SSH. Anda menggunakan pasangan kunci untuk masuk dengan aman ke instans Anda. Anda menentukan nama pasangan kunci saat meluncurkan instans kontainer Anda, kemudian berikan kunci privat saat masuk menggunakan SSH.

Jika Anda belum membuat key pair, Anda dapat membuatnya menggunakan EC2 konsol Amazon. Jika Anda merencanakan untuk meluncurkan instans di beberapa Wilayah, Anda perlu membuat pasangan kunci di setiap Wilayah. Untuk informasi selengkapnya tentang wilayah, lihat Wilayah dan Zona Ketersediaan di Panduan EC2 Pengguna Amazon.

Untuk membuat pasangan kunci

- Gunakan EC2 konsol Amazon untuk membuat key pair. Untuk informasi selengkapnya tentang membuat key pair, lihat Membuat key pair di Panduan EC2 Pengguna Amazon.

Untuk informasi tentang cara menyambung ke instans, lihat Connect ke instans Linux Anda di Panduan EC2 Pengguna Amazon.

Instal AWS CLI

Ini AWS Management Console dapat digunakan untuk mengelola semua operasi secara manual dengan Amazon ECS. Namun, Anda dapat menginstal AWS CLI di desktop lokal atau kotak pengembang sehingga Anda dapat membuat skrip yang dapat mengotomatiskan tugas manajemen umum di Amazon ECS.

Untuk menggunakan Amazon ECS, instal AWS CLI versi terbaru. AWS CLI Untuk informasi tentang menginstal AWS CLI atau memutakhirkannya ke versi terbaru, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI di Panduan AWS Command Line Interface Pengguna.

The AWS Command Line Interface (AWS CLI) adalah alat terpadu yang dapat Anda gunakan untuk mengelola AWS layanan Anda. Dengan alat yang satu ini saja, Anda dapat mengontrol beberapa AWS layanan dan mengotomatiskan layanan ini melalui skrip. Perintah Amazon ECS di dalamnya AWS CLI adalah cerminan dari Amazon ECS API.

AWS CLI Ini cocok untuk pelanggan yang lebih suka dan terbiasa membuat skrip dan berinteraksi dengan alat baris perintah dan tahu persis tindakan mana yang ingin mereka lakukan pada sumber daya Amazon ECS mereka. AWS CLI Ini juga membantu pelanggan yang ingin membiasakan diri dengan Amazon ECS. APIs Pelanggan dapat menggunakannya AWS CLI untuk melakukan sejumlah operasi pada sumber daya Amazon ECS, termasuk operasi Buat, Baca, Perbarui, dan Hapus, langsung dari antarmuka baris perintah.

Gunakan AWS CLI jika Anda ingin atau ingin terbiasa dengan Amazon ECS APIs dan perintah CLI yang sesuai dan ingin menulis skrip otomatis dan melakukan tindakan spesifik pada sumber daya Amazon ECS.

AWS juga menyediakan alat baris perintah AWS Tools for Windows PowerShell. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Tools for Windows PowerShell.

Langkah selanjutnya untuk menggunakan Amazon ECS

Setelah menginstal AWS CLI, ada banyak alat berbeda yang dapat Anda gunakan saat Anda terus menggunakan Amazon ECS. Tautan berikut menjelaskan apa saja beberapa alat tersebut dan memberikan contoh cara menggunakannya dengan Amazon ECS.

- Buat gambar kontainer pertama Anda dengan Docker dan dorong ke Amazon ECR untuk digunakan dalam definisi tugas Amazon ECS Anda.
- Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate.
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate.
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis EC2 peluncuran.
- Menggunakan bahasa pemrograman pilihan Anda, tentukan infrastruktur atau arsitektur sebagai kode dengan Membuat sumber daya Amazon ECS menggunakan AWS CDK.
- Tentukan dan kelola semua AWS sumber daya di lingkungan Anda dengan penggunaan Menggunakan Amazon ECS dengan AWS CloudFormation penerapan otomatis.
- Gunakan alur kerja Membuat sumber daya Amazon ECS menggunakan antarmuka baris AWS perintah Copilot end-to-end pengembang lengkap untuk membuat, merilis, dan mengoperasikan aplikasi kontainer yang sesuai dengan praktik AWS terbaik untuk infrastruktur.

Membuat gambar kontainer untuk digunakan di Amazon ECS

Amazon ECS menggunakan gambar Docker dalam definisi tugas untuk meluncurkan wadah. Docker adalah teknologi yang menyediakan alat bagi Anda untuk membangun, menjalankan, menguji, dan menyebarkan aplikasi terdistribusi dalam wadah.

Amazon ECS menjadwalkan aplikasi kontainer ke instans kontainer atau ke Fargate. AWS Aplikasi kontainer dikemas sebagai gambar kontainer. Contoh ini membuat gambar kontainer untuk server web.

Anda dapat membuat gambar Docker pertama Anda, dan kemudian mendorong gambar itu ke Amazon ECR, yang merupakan registri penampung, untuk digunakan dalam definisi tugas Amazon ECS Anda. Panduan ini mengasumsikan bahwa Anda memiliki pemahaman dasar tentang apa itu Docker dan cara kerjanya. Untuk informasi selengkapnya tentang Docker, lihat Apa itu Docker? dan dokumentasi Docker.

Prasyarat

Sebelum Anda mulai, pastikan prasyarat berikut terpenuhi.

- Pastikan Anda telah menyelesaikan langkah-langkah penyiapan Amazon ECR. Untuk informasi selengkapnya, lihat Memindahkan gambar melalui siklus hidupnya di Amazon ECR di Panduan Pengguna Amazon Elastic Container Registry.
- Pengguna Anda memiliki izin IAM yang diperlukan untuk mengakses dan menggunakan layanan Amazon ECR. Untuk informasi selengkapnya, lihat kebijakan terkelola Amazon ECR.
- Anda telah menginstal Docker. Untuk langkah-langkah instalasi Docker untuk Amazon Linux 2023, lihat. Menginstal Docker pada 023 AL2 Untuk semua sistem operasi lainnya, lihat dokumentasi Docker di ikhtisar Docker Desktop.
- Anda telah AWS CLI diinstal dan dikonfigurasi. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru dari Panduan AWS Command Line Interface Pengguna. AWS CLI

Jika Anda tidak memiliki atau memerlukan lingkungan pengembangan lokal dan Anda lebih suka menggunakan EC2 instans Amazon untuk menggunakan Docker, kami menyediakan langkah-langkah berikut untuk meluncurkan EC2 instans Amazon menggunakan Amazon Linux 2023 dan menginstal Docker Engine dan CLI Docker.

Menginstal Docker pada 023 AL2

Docker tersedia dalam banyak sistem operasi yang berbeda, termasuk sebagian besar distribusi Linux modern, seperti Ubuntu, dan bahkan macOS dan Windows. Untuk informasi lebih lanjut tentang cara menginstal Docker pada sistem operasi tertentu Anda, kunjungi situs webpanduan penginstalan Docker.

Anda tidak memerlukan sistem pengembangan lokal untuk menggunakan Docker. Jika Anda EC2 sudah menggunakan Amazon, Anda dapat meluncurkan instans Amazon Linux 2023 dan menginstal Docker untuk memulai.

Jika Anda sudah menginstal Docker, langsung ke Membuat citra Docker.

Untuk menginstal Docker di EC2 instans Amazon menggunakan Amazon Linux 2023 AMI

1. Luncurkan instance dengan AMI Amazon Linux 2023 terbaru. Untuk informasi selengkapnya, lihat Meluncurkan EC2 instance menggunakan wizard instans peluncuran di konsol di Panduan EC2 Pengguna Amazon.
2. Terhubung ke instans Anda. Untuk informasi selengkapnya, lihat Connect ke EC2 instans Anda di Panduan EC2 Pengguna Amazon.
3. Perbarui paket yang diinstal dan paket cache pada instans Anda.

   ```
   sudo yum update -y
   ```

4. Instal paket Edisi Komunitas Docker terbaru.

   ```
   sudo yum install docker
   ```

5. Mulai layanan Docker.

   ```
   sudo service docker start
   ```

6. Tambahkan `ec2-user` ke grup `docker` sehingga Anda dapat menjalankan perintah Docker tanpa menggunakan `sudo`.

   ```
   sudo usermod -a -G docker ec2-user
   ```

7. Keluar dan masuk kembali untuk mengambil izin grup `docker` yang baru. Anda dapat melakukannya dengan menutup jendela terminal SSH Anda saat ini dan menghubungkan

kembali ke instans Anda yang baru. Sesi SSH baru Anda akan memiliki izin grup `docker` yang sesuai.

8. Verifikasi bahwa `ec2-user` dapat menjalankan perintah Docker tanpa `sudo`.

```
docker info
```

> **Note**
>
> Dalam beberapa kasus, Anda mungkin perlu melakukan booting ulang pada instans Anda untuk memberikan izin bagi `ec2-user` untuk mengakses daemon Docker. Coba booting ulang instans Anda jika Anda melihat kesalahan berikut ini:
>
> ```
> Cannot connect to the Docker daemon. Is the docker daemon running on this
>  host?
> ```

Membuat citra Docker

Definisi tugas Amazon ECS menggunakan gambar kontainer untuk meluncurkan kontainer pada instance kontainer di cluster Anda. Di bagian ini, Anda membuat gambar Docker dari aplikasi web sederhana, dan mengujinya di sistem lokal atau EC2 instans Amazon, lalu mendorong gambar ke registri penampung Amazon ECR sehingga Anda dapat menggunakannya dalam definisi tugas Amazon ECS.

Untuk membuat citra Docker dari aplikasi web sederhana

1. Buat file bernama `Dockerfile`. Dockerfile adalah manifes yang menjelaskan citra dasar yang akan digunakan untuk citra Docker Anda dan apa yang ingin Anda instal dan jalankan di atasnya. Untuk informasi selengkapnya tentang Dockerfiles, buka Referensi Dockerfile.

```
touch Dockerfile
```

2. Edit `Dockerfile` yang baru saja Anda buat dan tambahkan konten berikut.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Update installed packages and install Apache
RUN yum update -y && \
```

```
 yum install -y httpd

# Write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure Apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

Dockerfile ini menggunakan gambar Amazon Linux 2023 publik yang dihosting di Amazon ECR Public. `RUN`Instruksi memperbarui cache paket, menginstal beberapa paket perangkat lunak untuk server web, dan kemudian menulis “Hello World!” konten ke root dokumen server web. `EXPOSE`Instruksi berarti bahwa port 80 pada wadah adalah yang mendengarkan, dan `CMD` instruksi memulai server web.

3. Membangun citra Docker dari Dockerfile Anda.

> **Note**
>
> Beberapa versi Docker mungkin memerlukan jalur lengkap ke Dockerfile Anda dalam perintah berikut, bukan jalur relatif yang ditunjukkan di bawah ini.
> Jika Anda menjalankan perintah sistem berbasis ARM, seperti Apple Silicon, gunakan opsi --platform “--platform linux/amd64".

```
docker build -t hello-world .
```

4. Buat daftar gambar kontainer Anda.

```
docker images --filter reference=hello-world
```

Output:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. Jalankan citra yang baru dibuat. Opsi `-p 80:80` memetakan port 80 yang terbuka pada kontainer ke port 80 pada sistem host.

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> Output dari server web Apache ditampilkan di jendela terminal. Anda dapat mengabaikan pesan "`Could not reliably determine the fully qualified domain name`".

6. Buka peramban dan arahkan ke server yang menjalankan Docker dan meng-host kontainer Anda.

   - Jika Anda menggunakan EC2 instance, ini adalah nilai DNS Publik untuk server, yang merupakan alamat yang sama yang Anda gunakan untuk terhubung ke instance dengan SSH. Pastikan bahwa grup keamanan untuk instans Anda mengizinkan lalu lintas masuk pada port 80.
   - Jika Anda menjalankan Docker secara lokal, arahkan peramban Anda ke http://localhost/.

   Anda akan melihat halaman web dengan pernyataan "Hello World!" .

7. Hentikan kontainer Docker dengan mengetik Ctrl + c.

Dorong gambar Anda ke Amazon Elastic Container Registry

Amazon ECR adalah layanan registri gambar terkelola yang AWS dikelola. Anda dapat menggunakan CLI Docker untuk mendorong, menarik, dan mengelola gambar di repositori Amazon ECR Anda. Untuk detail produk Amazon ECR, studi kasus pelanggan unggulan, dan FAQs, lihat halaman detail produk Amazon Elastic Container Registry.

Untuk menandai gambar Anda dan mendorongnya ke Amazon ECR

1. Buat repositori Amazon ECR untuk menyimpan gambar Anda. `hello-world` Perhatikan `repositoryUri` pada outputnya.

   Gantikan`region`, dengan Anda Wilayah AWS, misalnya,`us-east-1`.

   ```
   aws ecr create-repository --repository-name hello-repository --region region
   ```

   Output:

   ```
   {
       "repository": {
           "registryId": "aws_account_id",
           "repositoryName": "hello-repository",
           "repositoryArn": "arn:aws:ecr:region:aws_account_id:repository/hello-
   repository",
           "createdAt": 1505337806.0,
           "repositoryUri": "aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository"
       }
   }
   ```

2. Tandai citra `hello-world` dengan nilai `repositoryUri` dari langkah sebelumnya.

   ```
   docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
   ```

3. Jalankan perintah aws ecr get-login-password. Tentukan URI registri yang ingin Anda autentikasi. Untuk informasi selengkapnya, lihat Autentikasi Registri dalam Panduan Pengguna Amazon Elastic Container Registry.

   ```
   aws ecr get-login-password --region region | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

   Output:

   ```
   Login Succeeded
   ```

> **Important**
>
> Jika Anda menerima kesalahan, instal atau perbarui ke versi terbaru AWS CLI. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru dari Panduan AWS Command Line Interface Pengguna. AWS CLI

4. Dorong gambar ke Amazon ECR dengan `repositoryUri` nilai dari langkah sebelumnya.

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

Bersihkan

Untuk melanjutkan dengan membuat definisi tugas Amazon ECS dan meluncurkan tugas dengan gambar penampung Anda, lewati ke file. Langkah selanjutnya Setelah selesai bereksperimen dengan gambar Amazon ECR Anda, Anda dapat menghapus repositori sehingga Anda tidak dikenakan biaya untuk penyimpanan gambar.

```
aws ecr delete-repository --repository-name hello-repository --region region --force
```

Langkah selanjutnya

Definisi tugas Anda memerlukan peran eksekusi tugas. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Setelah Anda membuat dan mendorong gambar penampung Anda ke Amazon ECR, Anda dapat menggunakan gambar itu dalam definisi tugas. Untuk informasi selengkapnya, lihat salah satu dari berikut ini:

- the section called “Pelajari cara membuat tugas Linux untuk jenis peluncuran Fargate”
- the section called “Pelajari cara membuat tugas Windows untuk jenis peluncuran Fargate”
- Membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate dengan AWS CLI

Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate

Amazon Elastic Container Service (Amazon ECS) adalah layanan manajemen kontainer yang sangat skalabel, cepat, yang memudahkan untuk menjalankan, menghentikan, dan mengelola kontainer Anda. Anda dapat meng-host kontainer Anda di infrastruktur tanpa server yang dikelola oleh Amazon ECS dengan meluncurkan layanan atau tugas Anda. AWS Fargate Untuk informasi lebih lanjut tentang Fargate, lihat. AWS Fargateuntuk Amazon ECS

Mulai menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Anda di Wilayah tempat Amazon ECS AWS mendukung Fargate.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS aktif. AWS Fargate

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa pengguna IAM Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS
- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan untuk EC2 instans Amazon Anda di Panduan EC2 Pengguna Amazon.

- Jenis: HTTP
- Protokol: TCP
- Rentang port: 80
- Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Buat cluster yang menggunakan VPC default.

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Pada Konfigurasi klaster, untuk Nama klaster, masukkan nama unik.

   Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
6. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
7. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
8. Pilih Buat.

Langkah 2: Buat definisi tugas

Ketentuan tugas adalah seperti cetak biru untuk aplikasi Anda. Setiap kali Anda meluncurkan tugas di Amazon ECS, Anda menentukan definisi tugas. Layanan kemudian tahu citra Docker mana yang digunakan untuk kontainer, berapa banyak kontainer untuk digunakan dalam tugas, dan alokasi sumber daya untuk setiap kontainer.

1. Di panel navigasi, pilih Ketentuan Tugas.
2. Pilih Buat Definisi Tugas baru, Buat revisi baru dengan JSON.
3. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
```

```
    ],
    "cpu": "256",
    "memory": "512"
}
```

4. Pilih Buat.

Langkah 3: Buat layanan

Buat layanan menggunakan definisi tugas.

1. Di panel navigasi, pilih Clusters, lalu pilih cluster yang Anda buat. Langkah 1: Buat cluster
2. Dari tab Layanan, pilih Buat.
3. Di bawah konfigurasi Deployment, tentukan cara aplikasi Anda di-deploy.
   a. Untuk definisi Tugas, pilih definisi tugas yang Anda buatLangkah 2: Buat definisi tugas.
   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
   c. Untuk tugas yang diinginkan, masukkan 1.
4. Di bawah Jaringan, Anda dapat membuat grup keamanan baru atau memilih grup keamanan yang ada untuk tugas Anda. Pastikan grup keamanan yang Anda gunakan memiliki aturan masuk yang tercantum di bawahPrasyarat.
5. Pilih Buat.

Langkah 4: Melihat layanan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat layanan.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti EC2 instans Amazon, penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara manual di konsol EC2 Amazon atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster yang Anda buat untuk tutorial ini.
3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus. Atau, Anda dapat menggunakan `Force delete` opsi untuk meminta Amazon ECS menurunkan layanan atas nama Anda sebelum menghapusnya.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***`cluster-name`***, lalu pilih Hapus. Menghapus klaster akan membersihkan sumber daya terkait yang dibuat dengan klaster, termasuk grup Auto Scaling, atau penyeimbang VPCs beban.

Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate

Mulai menggunakan Amazon ECS AWS Fargate dengan menggunakan jenis peluncuran Fargate untuk tugas Anda di Wilayah tempat Amazon ECS AWS mendukung Fargate.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS aktif. AWS Fargate

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa pengguna IAM Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS
- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Important

Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan untuk EC2 instans Amazon Anda di Panduan EC2 Pengguna Amazon.

- Jenis: HTTP
- Protokol: TCP
- Rentang port: 80
- Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Anda dapat membuat cluster baru yang disebut windows yang menggunakan VPC default.

Untuk membuat cluster dengan AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan jendela.

6. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.
7. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
8. Pilih Buat.

Langkah 2: Daftarkan ketentuan tugas Windows

Sebelum Anda dapat menjalankan wadah Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Contoh ketentuan tugas berikut menampilkan halaman web sederhana pada port 8080 dari instans kontainer dengan citra kontainer `mcr.microsoft.com/windows/servercore/iis`.

Untuk mendaftarkan definisi tugas sampel dengan AWS Management Console

1. Di panel navigasi, pilih Definisi tugas.
2. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
3. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
```

```
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

4. Verifikasi informasi Anda lalu pilih Buat.

Langkah 3: Buat layanan dengan ketentuan tugas Anda

Setelah Anda mendaftarkan ketentuan tugas, Anda dapat menempatkan tugas di klaster Anda dengan ketentuan tersebut. Prosedur berikut membuat layanan dengan definisi tugas Anda dan menempatkan satu tugas di klaster Anda.

Untuk membuat layanan dari ketentuan tugas Anda dengan konsol

1. Di panel navigasi, pilih Clusters, lalu pilih cluster yang Anda buat. Langkah 1: Buat cluster
2. Dari tab Layanan, pilih Buat.
3. Di bawah konfigurasi Deployment, tentukan cara aplikasi Anda di-deploy.
    a. Untuk definisi Tugas, pilih definisi tugas yang Anda buatLangkah 2: Daftarkan ketentuan tugas Windows.

b. Untuk nama Layanan, masukkan nama untuk layanan Anda.

c. Untuk tugas yang diinginkan, masukkan 1.

4. Di bawah Jaringan, Anda dapat membuat grup keamanan atau memilih grup yang ada. Pastikan grup keamanan yang Anda gunakan memiliki aturan masuk yang tercantum di bawahPrasyarat.
5. Pilih Buat.

Langkah 4: Melihat layanan Anda

Setelah layanan Anda meluncurkan tugas ke klaster Anda, Anda dapat melihat layanan dan membuka halaman uji IIS di peramban untuk memverifikasi bahwa kontainer berjalan.

> **Note**
>
> Hal ini dapat memakan waktu hingga 15 menit bagi instans kontainer Anda untuk mengunduh dan mengekstrak lapisan dasar kontainer Windows.

Untuk melihat layanan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat layanan dengan ketentuan tugas Anda.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti EC2 instans Amazon, penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara

manual di konsol EC2 Amazon atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster yang Anda buat untuk tutorial ini.
3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***`cluster-name`***, lalu pilih Hapus. Menghapus klaster akan membersihkan sumber daya terkait yang dibuat dengan klaster, termasuk grup Auto Scaling, atau penyeimbang VPCs beban.

Pelajari cara membuat tugas Amazon ECS Windows untuk jenis EC2 peluncuran

Memulai Amazon ECS menggunakan tipe EC2 peluncuran dengan mendaftarkan definisi tugas, membuat klaster, dan membuat layanan di konsol.

Selesaikan langkah-langkah berikut untuk memulai Amazon ECS menggunakan jenis EC2 peluncuran.

Prasyarat

Sebelum memulai, selesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan bahwa pengguna IAM Anda memiliki izin yang ditentukan dalam contoh kebijakan `AdministratorAccess` IAM.

Konsol mencoba untuk secara otomatis membuat peran IAM eksekusi tugas, yang diperlukan untuk tugas Fargate. Untuk memastikan bahwa konsol dapat membuat peran IAM ini, salah satu dari berikut ini harus benar:

- Pengguna Anda memiliki akses administrator. Untuk informasi selengkapnya, lihat Siapkan untuk menggunakan Amazon ECS.
- Pengguna Anda memiliki izin IAM untuk membuat peran layanan. Untuk informasi selengkapnya, lihat Membuat Peran untuk Mendelegasikan Izin ke Layanan. AWS

- Pengguna dengan akses administrator telah secara manual membuat peran eksekusi tugas sehingga peran itu tersedia pada akun yang akan digunakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Important

Grup keamanan yang Anda pilih saat membuat layanan dengan definisi tugas Anda harus memiliki port 80 terbuka untuk lalu lintas masuk. Tambahkan aturan masuk berikut ke grup keamanan Anda. Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan untuk EC2 instans Amazon Anda di Panduan EC2 Pengguna Amazon.

- Jenis: HTTP
- Protokol: TCP
- Rentang port: 80
- Sumber: Di mana saja (`0.0.0.0/0`)

Langkah 1: Buat cluster

Cluster Amazon ECS adalah pengelompokan tugas, layanan, dan instance kontainer yang logis.

Langkah-langkah berikut memandu Anda membuat cluster dengan satu EC2 instance Amazon terdaftar di dalamnya yang akan memungkinkan kami menjalankan tugas di atasnya. Jika bidang tertentu tidak disebutkan, tinggalkan nilai konsol default.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Pada Konfigurasi klaster, untuk Nama klaster, masukkan nama unik.

Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.

6. (Opsional) Untuk mengubah VPC dan subnet tempat tugas dan layanan Anda diluncurkan, di bawah Jaringan, lakukan salah satu operasi berikut:

   - Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
   - Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, lalu di bawah Subnet, pilih setiap subnet.

7. Untuk menambahkan EC2 instans Amazon ke klaster Anda, perluas Infrastruktur, lalu pilih EC2instans Amazon. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:

   a. Untuk menggunakan grup Auto Scaling yang ada, dari grup Auto Scaling (ASG), pilih grup.

   b. Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:

      - Untuk Sistem Operasi/Arsitektur, pilih AMI Amazon ECS yang dioptimalkan untuk instans grup Auto Scaling.
      - EC2 Misalnya jenis, pilih jenis instance untuk beban kerja Anda. Untuk informasi selengkapnya tentang berbagai jenis instans, lihat EC2 Instans Amazon.

        Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa.
      - Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
      - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling. EC2 Instans Amazon mengeluarkan biaya saat ada di sumber daya Anda AWS . Untuk informasi selengkapnya, lihat EC2 Harga Amazon.

8. (Opsional) Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.

9. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

- Untuk Kunci, masukkan nama kunci.
- Untuk Nilai, masukkan nilai kunci.

[Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.

10. Pilih Buat.

Langkah 2: Daftarkan definisi tugas

Untuk mendaftarkan definisi tugas sampel dengan AWS Management Console

1. Di panel navigasi, pilih Ketentuan Tugas.
2. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
3. Salin dan tempel contoh definisi tugas berikut ke dalam kotak, lalu pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
```

```
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
}
```

4. Verifikasi informasi Anda lalu pilih Buat.

Langkah 3: Buat Layanan

Layanan Amazon ECS membantu Anda menjalankan dan memelihara sejumlah instans definisi tugas yang ditentukan secara bersamaan di kluster Amazon ECS. Jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantinya guna mempertahankan jumlah tugas yang diinginkan dalam layanan. Untuk informasi selengkapnya tentang layanan, lihat Layanan-layanan Amazon ECS.

Untuk membuat layanan

1. Pada panel navigasi, silakan pilih Klaster.
2. Pilih cluster yang Anda buatLangkah 1: Buat cluster.
3. Pada tab Layanan, pilih Buat.
4. Di bagian Lingkungan, lakukan hal berikut:
    a. Untuk opsi Komputasi, pilih Jenis peluncuran.
    b. Untuk jenis Peluncuran, pilih EC2
5. Di bagian konfigurasi Deployment, lakukan hal berikut:
    a. Untuk Keluarga, pilih definisi tugas yang Anda buatLangkah 2: Daftarkan definisi tugas.
    b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
    c. Untuk tugas yang diinginkan, masukkan 1.
6. Tinjau opsi dan pilih Buat.
7. Pilih Lihat layanan untuk meninjau layanan Anda.

Langkah 4: Lihat Layanan Anda

Layanan ini adalah aplikasi berbasis web sehingga Anda dapat melihat kontainer dengan peramban web.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster tempat Anda menjalankan layanan.
4. Di tab Layanan, di bawah Nama layanan, pilih layanan yang Anda buatLangkah 3: Buat Layanan.
5. Pilih tab Tugas, lalu pilih tugas di layanan Anda.
6. Pada halaman tugas, di bagian Konfigurasi, di bawah IP Publik, pilih Buka alamat. Screen shot di bawah ini adalah output yang diharapkan.

Langkah 5: Bersihkan

Setelah selesai menggunakan kluster Amazon ECS, Anda harus membersihkan sumber daya yang terkait dengannya untuk menghindari biaya untuk sumber daya yang tidak Anda gunakan.

Beberapa sumber daya Amazon ECS, seperti tugas, layanan, cluster, dan instans kontainer, dibersihkan menggunakan konsol Amazon ECS. Sumber daya lain, seperti EC2 instans Amazon, penyeimbang beban Elastic Load Balancing, dan grup Auto Scaling, harus dibersihkan secara manual di konsol EC2 Amazon atau dengan menghapus tumpukan yang membuatnya. AWS CloudFormation

1. Pada panel navigasi, silakan pilih Klaster.
2. Pada halaman Clusters, pilih cluster cluster yang Anda buat untuk tutorial ini.

3. Pilih tab Layanan.
4. Pilih layanan, lalu pilih Hapus.
5. Pada prompt konfirmasi, masukkan hapus dan kemudian pilih Hapus.

   Tunggu hingga layanan dihapus.
6. Pilih Hapus klaster. Pada prompt konfirmasi, masukkan hapus ***cluster-name***, lalu pilih Hapus. Menghapus klaster akan membersihkan sumber daya terkait yang dibuat dengan klaster, termasuk grup Auto Scaling, atau penyeimbang VPCs beban.

Membuat sumber daya Amazon ECS menggunakan AWS CDK

AWS Cloud Development Kit (AWS CDK) Ini adalah kerangka kerja Infrastructure-as-Code (IAC) yang dapat Anda gunakan untuk mendefinisikan infrastruktur AWS cloud dengan menggunakan bahasa pemrograman pilihan Anda. Untuk menentukan infrastruktur cloud Anda sendiri, pertama-tama Anda menulis aplikasi (dalam salah satu bahasa yang didukung CDK) yang berisi satu atau beberapa tumpukan. Kemudian, Anda mensintesisnya ke AWS CloudFormation template dan menyebarkan sumber daya Anda ke template Anda. Akun AWS Ikuti langkah-langkah dalam topik ini untuk menerapkan server web dalam kontainer dengan Amazon Elastic Container Service (Amazon ECS) Container Service (Amazon ECS) dan di Fargate. AWS CDK

AWS Construct Library, disertakan dengan CDK, menyediakan modul yang dapat Anda gunakan untuk memodelkan sumber daya yang Layanan AWS disediakan. Untuk layanan populer, perpustakaan menyediakan konstruksi yang dikuratori dengan default cerdas dan praktik terbaik. Salah satu modul ini, khususnya`aws-ecs-patterns`, menyediakan abstraksi tingkat tinggi yang dapat Anda gunakan untuk menentukan layanan kontainer Anda dan semua sumber daya pendukung yang diperlukan dalam beberapa baris kode.

Topik ini menggunakan `ApplicationLoadBalancedFargateService`konstruksi. Konstruksi ini menyebarkan layanan Amazon ECS di Fargate di belakang penyeimbang beban aplikasi. `aws-ecs-patterns`Modul ini juga mencakup konstruksi yang menggunakan penyeimbang beban jaringan dan berjalan di Amazon. EC2

Sebelum memulai tugas ini, atur lingkungan AWS CDK pengembangan Anda, dan instal AWS CDK dengan menjalankan perintah berikut. Untuk petunjuk tentang cara mengatur lingkungan AWS CDK pengembangan Anda, lihat Memulai Dengan AWS CDK - Prasyarat.

```
npm install -g aws-cdk
```

> **Note**
>
> Instruksi ini mengasumsikan Anda menggunakan AWS CDK v2.

Topik

- Langkah 1: Siapkan AWS CDK proyek Anda
- Langkah 2: Gunakan AWS CDK untuk mendefinisikan server web kontainer di Fargate
- Langkah 3: Uji server web
- Langkah 4: Membersihkan
- Langkah selanjutnya

Langkah 1: Siapkan AWS CDK proyek Anda

Buat direktori untuk AWS CDK aplikasi baru Anda dan inisialisasi proyek.

TypeScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language typescript
```

JavaScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language javascript
```

Python

```
mkdir hello-ecs
cd hello-ecs
cdk init --language python
```

Setelah proyek dimulai, aktifkan lingkungan virtual proyek dan instal AWS CDK dependensi dasar.

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir hello-ecs
cd hello-ecs
cdk init --language java
```

Impor proyek Maven ini ke IDE Java Anda. Misalnya, di Eclipse, gunakan File > Import > Maven > Existing Maven Projects.

C#

```
mkdir hello-ecs
cd hello-ecs
cdk init --language csharp
```

Go

```
mkdir hello-ecs
cd hello-ecs
cdk init --language go
```

> **Note**
>
> Template AWS CDK aplikasi menggunakan nama direktori proyek untuk menghasilkan nama untuk file sumber dan kelas. Dalam contoh ini, direktori diberi nama`hello-ecs`. Jika Anda menggunakan nama direktori project yang berbeda, aplikasi Anda tidak akan cocok dengan petunjuk ini.

AWS CDK v2 menyertakan konstruksi stabil untuk semua Layanan AWS dalam satu paket yang dipanggil`aws-cdk-lib`. Paket ini diinstal sebagai dependensi ketika Anda menginisialisasi proyek. Saat bekerja dengan bahasa pemrograman tertentu, paket diinstal saat Anda membangun proyek untuk pertama kalinya. Topik ini mencakup cara menggunakan konstruksi Pola Amazon ECS, yang menyediakan abstraksi tingkat tinggi untuk bekerja dengan Amazon ECS. Modul ini bergantung pada konstruksi Amazon ECS dan konstruksi lainnya untuk menyediakan sumber daya yang dibutuhkan aplikasi Amazon ECS Anda.

Nama-nama yang Anda gunakan untuk mengimpor pustaka ini ke dalam aplikasi CDK Anda mungkin sedikit berbeda tergantung pada bahasa pemrograman yang Anda gunakan. Sebagai referensi,

berikut ini adalah nama-nama yang digunakan dalam setiap bahasa pemrograman CDK yang didukung.

TypeScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

JavaScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

Python

```
aws_cdk.aws_ecs
aws_cdk.aws_ecs_patterns
```

Java

```
software.amazon.awscdk.services.ecs
software.amazon.awscdk.services.ecs.patterns
```

C#

```
Amazon.CDK.AWS.ECS
Amazon.CDK.AWS.ECS.Patterns
```

Go

```
github.com/aws/aws-cdk-go/awscdk/v2/awsecs
github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns
```

Langkah 2: Gunakan AWS CDK untuk mendefinisikan server web kontainer di Fargate

Gunakan gambar kontainer `amazon-ecs-sample`. Gambar ini berisi aplikasi web PHP yang berjalan di Ngingx.

Dalam AWS CDK proyek yang Anda buat, edit file yang berisi definisi tumpukan agar menyerupai salah satu contoh berikut.

> **Note**
>
> Tumpukan adalah unit penyebaran. Semua sumber daya harus dalam tumpukan, dan semua sumber daya yang ada di tumpukan digunakan pada saat yang sama. Jika sumber daya gagal diterapkan, sumber daya lain yang sudah digunakan akan digulirkan kembali. AWS CDK Aplikasi dapat berisi beberapa tumpukan, dan sumber daya dalam satu tumpukan dapat merujuk ke sumber daya di tumpukan lain.

TypeScript

Perbarui `lib/hello-ecs-stack.ts` sehingga menyerupai yang berikut ini.

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsp from 'aws-cdk-lib/aws-ecs-patterns';

export class HelloEcsStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('public.ecr.aws/ecs-sample-image/
amazon-ecs-sample:latest'),
      },
      publicLoadBalancer: true
    });
  }
}
```

JavaScript

Perbarui `lib/hello-ecs-stack.js` sehingga menyerupai yang berikut ini.

```
const cdk = require('aws-cdk-lib');
const { Construct } = require('constructs');
```

```
const ecs = require('aws-cdk-lib/aws-ecs');
const ecsp = require('aws-cdk-lib/aws-ecs-patterns');

class HelloEcsStack extends cdk.Stack {
  constructor(scope = Construct, id = string, props = cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}

module.exports = { HelloEcsStack }
```

Python

Perbarui `hello-ecs/hello_ecs_stack.py` sehingga menyerupai yang berikut ini.

```
import aws_cdk as cdk
from constructs import Construct

import aws_cdk.aws_ecs as ecs
import aws_cdk.aws_ecs_patterns as ecsp

class HelloEcsStack(cdk.Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ecsp.ApplicationLoadBalancedFargateService(self, "MyWebServer",
            task_image_options=ecsp.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry("amazon/amazon-ecs-sample")),
            public_load_balancer=True
        )
```

Java

Perbarui `src/main/java/com.myorg/HelloEcsStack.java` sehingga menyerupai yang berikut ini.

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;

import software.amazon.awscdk.services.ecs.ContainerImage;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedFargateService;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedTaskImageOptions;

public class HelloEcsStack extends Stack {
    public HelloEcsStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public HelloEcsStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        ApplicationLoadBalancedFargateService.Builder.create(this, "MyWebServer")
         .taskImageOptions(ApplicationLoadBalancedTaskImageOptions.builder()
           .image(ContainerImage.fromRegistry("amazon/amazon-ecs-sample"))
           .build())
         .publicLoadBalancer(true)
         .build();
    }
}
```

C#

Perbarui `src/HelloEcs/HelloEcsStack.cs` sehingga menyerupai yang berikut ini.

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.ECS;
using Amazon.CDK.AWS.ECS.Patterns;
namespace HelloEcs
{
    public class HelloEcsStack : Stack
    {
```

```
        internal HelloEcsStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            new ApplicationLoadBalancedFargateService(this, "MyWebServer",
                new ApplicationLoadBalancedFargateServiceProps
                {
                    TaskImageOptions = new ApplicationLoadBalancedTaskImageOptions
                    {
                        Image = ContainerImage.FromRegistry("amazon/amazon-ecs-
sample")
                    },
                    PublicLoadBalancer = true
                });
        }
    }
}
```

Go

Perbarui `hello-ecs.go` sehingga menyerupai yang berikut ini.

```
package main

import (
 "github.com/aws/aws-cdk-go/awscdk/v2"
 // "github.com/aws/aws-cdk-go/awscdk/v2/awssqs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns"
 "github.com/aws/constructs-go/constructs/v10"
 "github.com/aws/jsii-runtime-go"
)

type HelloEcsStackProps struct {
 awscdk.StackProps
}

func NewHelloEcsStack(scope constructs.Construct, id string, props
 *HelloEcsStackProps) awscdk.Stack {
 var sprops awscdk.StackProps
 if props != nil {
  sprops = props.StackProps
 }
 stack := awscdk.NewStack(scope, &id, &sprops)
```

```
 // The code that defines your stack goes here

 // example resource
 // queue := awssqs.NewQueue(stack, jsii.String("HelloEcsQueue"),
 &awssqs.QueueProps{
 //  VisibilityTimeout: awscdk.Duration_Seconds(jsii.Number(300)),
 // })
 res := awsecspatterns.NewApplicationLoadBalancedFargateService(stack,
 jsii.String("MyWebServer"),
  &awsecspatterns.ApplicationLoadBalancedFargateServiceProps{
   TaskImageOptions: &awsecspatterns.ApplicationLoadBalancedTaskImageOptions{
    Image: awsecs.ContainerImage_FromRegistry(jsii.String("amazon/amazon-ecs-
sample"), &awsecs.RepositoryImageProps{}),
   },
  },
 )
 awscdk.NewCfnOutput(stack, jsii.String("LoadBalancerDNS"),
 &awscdk.CfnOutputProps{Value: res.LoadBalancer().LoadBalancerDnsName()})

 return stack
}

func main() {
 defer jsii.Close()

 app := awscdk.NewApp(nil)

 NewHelloEcsStack(app, "HelloEcsStack", &HelloEcsStackProps{
  awscdk.StackProps{
   Env: env(),
  },
 })

 app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
func env() *awscdk.Environment {
 // If unspecified, this stack will be "environment-agnostic".
 // Account/Region-dependent features and context lookups will not work, but a
 // single synthesized template can be deployed anywhere.
 //---------------------------------------------------------------------------
```

```
 return nil

 // Uncomment if you know exactly what account and region you want to deploy
 // the stack to. This is the recommendation for production stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String("123456789012"),
 //  Region:  jsii.String("us-east-1"),
 // }

 // Uncomment to specialize this stack for the AWS Account and Region that are
 // implied by the current CLI configuration. This is recommended for dev
 // stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
 //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
 // }
}
```

Cuplikan pendek sebelumnya mencakup yang berikut:

- Nama logis layanan:`MyWebServer`.
- Gambar wadah yang diperoleh dari Galeri Publik Amazon ECR:`amazon/amazon-ecs-sample`.
- Informasi relevan lainnya, seperti fakta bahwa penyeimbang beban memiliki alamat publik dan dapat diakses dari Internet.

AWS CDK Akan membuat semua sumber daya yang diperlukan untuk menyebarkan server web termasuk sumber daya berikut. Sumber daya ini dihilangkan dalam contoh ini.

- Kluster Amazon ECS
- Amazon VPC dan instans Amazon EC2
- Grup Auto Scaling
- Penyeimbang Beban Aplikasi
- Peran dan kebijakan IAM

Beberapa sumber daya yang disediakan secara otomatis dibagikan oleh semua layanan Amazon ECS yang ditentukan dalam tumpukan.

Simpan file sumber, lalu jalankan `cdk synth` perintah di direktori utama aplikasi Anda. AWS CDK Menjalankan aplikasi dan mensintesis AWS CloudFormation template darinya, dan kemudian menampilkan template. Template adalah file YAML sekitar 600 baris. Awal file ditampilkan di sini. Template Anda mungkin berbeda dari contoh ini.

```
Resources:
  MyWebServerLB3B5FD3AB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      LoadBalancerAttributes:
        - Key: deletion_protection.enabled
          Value: "false"
      Scheme: internet-facing
      SecurityGroups:
        - Fn::GetAtt:
            - MyWebServerLBSecurityGroup01B285AA
            - GroupId
      Subnets:
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1Subnet3C273B99
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2Subnet95FF715A
      Type: application
    DependsOn:
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1DefaultRouteFF4E2178
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2DefaultRouteB1375520
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/Resource
  MyWebServerLBSecurityGroup01B285AA:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Automatically created Security Group for ELB
 HelloEcsStackMyWebServerLB06757F57
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          Description: Allow from anyone on port 80
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
      VpcId:
        Ref: EcsDefaultClusterMnL3mNNYNVpc7788A521
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/SecurityGroup/Resource
```

```
# and so on for another few hundred lines
```

Untuk menyebarkan layanan di Anda Akun AWS, jalankan `cdk deploy` perintah di direktori utama aplikasi Anda. Anda diminta untuk menyetujui kebijakan IAM yang dihasilkan. AWS CDK

Penyebaran memakan waktu beberapa menit di mana AWS CDK menciptakan beberapa sumber daya. Beberapa baris terakhir dari output dari penyebaran termasuk nama host publik penyeimbang beban dan URL server web baru Anda. Mereka adalah sebagai berikut.

```
Outputs:
HelloEcsStack.MyWebServerLoadBalancerDNSXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZ-
ZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZ-
ZZZZZZZZZ.us-west-2.elb.amazonaws.com
```

Langkah 3: Uji server web

Salin URL dari output penyebaran dan tempel ke browser web Anda. Pesan selamat datang berikut dari server web ditampilkan.

Langkah 4: Membersihkan

Setelah Anda selesai dengan server web, akhiri layanan menggunakan CDK dengan menjalankan `cdk destroy` perintah di direktori utama aplikasi Anda. Melakukan hal ini mencegah Anda dari menimbulkan biaya yang tidak diinginkan di masa depan.

Langkah selanjutnya

Untuk mempelajari lebih lanjut tentang cara mengembangkan AWS infrastruktur menggunakan AWS CDK, lihat Panduan AWS CDK Pengembang.

Untuk informasi tentang menulis AWS CDK aplikasi dalam bahasa pilihan Anda, lihat berikut ini:

TypeScript

Bekerja dengan AWS CDK di TypeScript

JavaScript

Bekerja dengan AWS CDK di JavaScript

Python

Bekerja dengan AWS CDK in Python

Java

Bekerja dengan AWS CDK di Jawa

C#

Bekerja dengan AWS CDK di C #

Go

Bekerja dengan AWS CDK in Go

Untuk informasi selengkapnya tentang modul AWS Construct Library yang digunakan dalam topik ini, lihat ikhtisar Referensi AWS CDK API berikut.

- aws-ecs
- aws-ecs-patterns

Membuat sumber daya Amazon ECS menggunakan antarmuka baris AWS perintah Copilot

Perintah AWS Copilot command line interface (CLI) menyederhanakan pembuatan, pelepasan, dan pengoperasian aplikasi kontainer siap produksi di Amazon ECS dari lingkungan pengembangan

lokal. AWS CLI Copilot selaras dengan alur kerja pengembang yang mendukung praktik terbaik aplikasi modern: mulai dari menggunakan infrastruktur sebagai kode hingga membuat CI/CD pipeline yang disediakan atas nama pengguna. Gunakan AWS Copilot CLI sebagai bagian dari siklus pengembangan dan pengujian sehari-hari Anda sebagai alternatif. AWS Management Console

AWS Copilot saat ini mendukung sistem Linux, macOS, dan Windows. Untuk informasi selengkapnya tentang AWS CLI Copilot versi terbaru, lihat Rilis.

> **Note**
>
> Kode sumber untuk AWS Copilot CLI tersedia di. GitHub Kami merekomendasikan agar Anda mengirimkan masalah dan menarik permintaan untuk perubahan yang ingin Anda sertakan. Namun, Amazon Web Services saat ini tidak mendukung menjalankan salinan kode AWS Copilot yang dimodifikasi. Laporkan masalah dengan AWS Copilot dengan menghubungkan kami di Gitter atau di GitHubmana Anda dapat membuka masalah, memberikan umpan balik, dan melaporkan bug.
> Untuk informasi tentang pengembangan Copilot, lihat Apa masa depan Copilot? .

Topik

- Memasang AWS Copilot CLI
- Menyebarkan contoh aplikasi Amazon ECS menggunakan Copilot CLI AWS

Dokumentasi tambahan untuk AWS CLI Copilot tersedia di situs web Copilot AWS

Memasang AWS Copilot CLI

Anda dapat menginstal AWS Copilot CLI dengan menggunakan Homebrew atau dengan mengunduh biner secara manual dengan langkah-langkah berikut.

Gunakan Homebrew

Perintah berikut digunakan untuk menginstal AWS Copilot CLI pada sistem macOS atau Linux Anda menggunakan Homebrew. Sebelum instalasi, Anda harus sudah menginstal Homebrew. Untuk informasi selengkapnya, lihat Homebrew.

```
brew install aws/tap/copilot-cli
```

Unduh biner

Sebagai alternatif dari Homebrew, Anda dapat menginstal AWS Copilot CLI secara manual di sistem macOS, Windows, atau Linux Anda. Gunakan perintah berikut untuk sistem operasi Anda untuk mengunduh biner. Contoh macOS dan Linux juga menyertakan perintah yang menerapkan izin eksekusi ke biner, dan daftar menu bantuan untuk memverifikasi bahwa instalasi berfungsi.

macOS

Untuk macOS:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Untuk sistem ARM macOS:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Linux

Pada sistem Linux x86 (64-bit):

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Pada sistem Linux ARM:

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Windows

Menggunakan Powershell, jalankan perintah berikut:

```
New-Item -Path 'C:\copilot' -ItemType directory; `
  Invoke-WebRequest -OutFile 'C:\copilot\copilot.exe' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe
```

(Opsional) Verifikasi CLI AWS Copilot yang diinstal secara manual menggunakan tanda tangan PGP

Executable AWS Copilot CLI ditandatangani secara kriptografis menggunakan tanda tangan PGP. Tanda tangan PGP dapat digunakan untuk memverifikasi validitas Copilot CLI yang dapat dieksekusi. AWS Gunakan langkah-langkah berikut untuk memverifikasi tanda tangan menggunakan alat GnuPG.

1. Unduh dan instal GnuPG. Untuk informasi selengkapnya, lihat situs web GnuPG.

   macOS

   Kami merekomendasikan menggunakan Homebrew. Instal Homebrew menggunakan instruksi dari situs web mereka. Untuk informasi selengkapnya, lihat Homebrew. Setelah Homebrew diinstal, gunakan perintah berikut dari terminal macOS Anda.

   ```
   brew install gnupg
   ```

   Linux

   Instal gpg menggunakan manajer paket pada rasa Linux Anda.

   Windows

   Unduh penginstal sederhana Windows dari situs web GnuPG dan instal sebagai Administrator. Setelah Anda menginstal GnuPG, tutup dan buka kembali Administrator. PowerShell

   Untuk informasi selengkapnya, lihat Unduhan GnuPG.
2. Verifikasi jalur GnuPG ditambahkan ke jalur lingkungan Anda.

   macOS

   ```
   echo $PATH
   ```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

Jika Anda tidak melihat jalur GnuPG di output, jalankan perintah berikut untuk menambahkannya ke jalur.

```
$Env:PATH += ";<path to GnuPG executable files>"
```

3. Buat file teks biasa lokal.

macOS

Di terminal, masukkan:

```
touch <public_key_filename.txt>
```

Buka file dengan TextEdit.

Linux

Buat file teks di editor teks seperti gedit. Simpan sebagai `public_key_filename.txt`

Windows

Buat file teks di editor teks seperti Notepad. Simpan sebagai `public_key_filename.txt`

4. Tambahkan konten berikut dari kunci publik Amazon ECS PGP dan simpan file.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
```

```
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
```

```
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

Rincian kunci publik Amazon ECS PGP untuk referensi:

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

5. Impor file dengan kunci publik Amazon ECS PGP dengan perintah berikut di terminal.

```
gpg --import <public_key_filename.txt>
```

6. Unduh tanda tangan AWS Copilot CLI. Tanda tangan tersebut adalah tandatangan PGP ASCII-terpisah yang tersimpan di file dengan ekstensi `.asc`. File standar memiliki nama yang sama dengan pelaksanaan yang sesuai, dengan `.asc` yang telah ditambahkan.

   macOS

   Untuk sistem macOS, jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-darwin.asc
   ```

   Linux

   Untuk sistem Linux x86 (64-bit), jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux.asc
   ```

   Untuk sistem Linux ARM, jalankan perintah berikut.

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux-arm64.asc
   ```

Windows

Menggunakan Powershell, jalankan perintah berikut.

```
Invoke-WebRequest -OutFile 'C:\copilot\copilot.asc' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe.asc
```

7. Verifikasi tanda tangan dengan menggunakan perintah berikut.

   - Untuk sistem macOS dan Linux:

   ```
   gpg --verify copilot.asc /usr/local/bin/copilot
   ```

   - Untuk sistem Windows:

   ```
   gpg --verify 'C:\copilot\copilot.asc' 'C:\copilot\copilot.exe'
   ```

   Keluaran yang diharapkan

   ```
   gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
   gpg:                using RSA key DE3CBD61ADAF8B8E
   gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
        Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
   ```

   Important

   Peringatan dalam output diharapkan dan tidak bermasalah. Itu terjadi karena tidak ada rantai kepercayaan antara kunci PGP pribadi Anda (jika Anda memilikinya) dan kunci Amazon ECS PGP. Untuk informasi selengkapnya, lihat Web kepercayaan.

8. Untuk instalasi Windows, jalankan perintah berikut di Powershell untuk menambahkan direktori AWS Copilot ke jalur.

   ```
   $Env:PATH += ";<path to Copilot executable files>"
   ```

Menyebarkan contoh aplikasi Amazon ECS menggunakan Copilot CLI AWS

Setelah menginstal AWS Copilot CLI, Anda dapat mengikuti langkah-langkah berikut untuk menerapkan aplikasi sampel, memverifikasi penerapan, dan membersihkan sumber daya.

Prasyarat

Sebelum memulai, pastikan Anda memenuhi persyaratan berikut:

- Instal dan konfigurasikan AWS CLI. Untuk informasi selengkapnya, lihat AWS Command Line Interface.
- Jalankan `aws configure` untuk mengatur profil default yang akan digunakan AWS CLI Copilot untuk mengelola aplikasi dan layanan Anda.
- Instal dan jalankan Docker. Untuk informasi selengkapnya, lihat Memulai dengan Docker.

Menyebarkan contoh aplikasi Amazon ECS menggunakan satu perintah

1. Menyebarkan contoh aplikasi web yang dikloning dari GitHub repositori menggunakan perintah berikut. Untuk informasi selengkapnya tentang AWS Copilot `init` dan benderanya, lihat dokumentasi Copilot.AWS

```
git clone https://github.com/aws-samples/aws-copilot-sample-service.git demo-app &&
 \
cd demo-app &&                                \
copilot init --app demo                       \
  --name api                                  \
  --type 'Load Balanced Web Service'          \
  --dockerfile './Dockerfile'                 \
  --port 80                                   \
  --tag  latest                               \
  --deploy
```

2. Setelah penerapan selesai, CLI AWS Copilot akan mengembalikan URL yang dapat Anda gunakan untuk memverifikasi penerapan. Anda juga dapat menggunakan perintah berikut untuk memverifikasi status aplikasi.

   - Daftar semua aplikasi AWS Copilot Anda.

```
copilot app ls
```

- Tampilkan informasi tentang lingkungan dan layanan dalam aplikasi Anda.

```
copilot app show
```

- Tampilkan informasi tentang lingkungan Anda.

```
copilot env ls
```

- Tampilkan informasi tentang layanan, termasuk titik akhir, kapasitas, dan sumber daya terkait.

```
copilot svc show
```

- Cantumkan semua layanan dalam aplikasi.

```
copilot svc ls
```

- Tampilkan log dari layanan yang di-deploy.

```
copilot svc logs
```

- Tampilkan status layanan.

```
copilot svc status
```

3. Setelah selesai dengan demo ini, jalankan perintah berikut untuk membersihkan sumber daya terkait dan menghindari biaya untuk sumber daya yang tidak digunakan.

```
copilot app delete
```

Menggunakan Amazon ECS dengan AWS CloudFormation

Amazon ECS terintegrasi dengan AWS CloudFormation, layanan yang dapat Anda gunakan untuk memodelkan dan menyiapkan AWS sumber daya dengan templat yang Anda tentukan. AWS CloudFormation menggunakan template yang merupakan file teks `YAML` atau `JSON` diformat. Template seperti cetak biru untuk AWS sumber daya yang ingin Anda buat. Saat Anda membuat dan mengirimkan template, AWS CloudFormation buat tumpukan. Anda mengelola sumber daya yang Anda tentukan dalam template Anda melalui tumpukan. Saat Anda ingin membuat, memperbarui, atau menghapus sumber daya, Anda membuat, memperbarui, atau menghapus tumpukan yang dibuat dari sumber daya tersebut. Ketika datang untuk memperbarui tumpukan Anda, Anda perlu membuat set perubahan terlebih dahulu. Set perubahan menunjukkan kepada Anda apa yang dipengaruhi oleh perubahan sebelum Anda membuatnya. Ini membuat Anda tidak menghapus database secara tidak sengaja dengan mengubah nama database Anda, misalnya. Untuk informasi selengkapnya tentang templat, tumpukan, dan set perubahan, lihat Cara AWS CloudFormation kerjanya di Panduan AWS CloudFormation Pengguna.

Dengan menggunakan AWS CloudFormation, Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda dapat membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan, seperti klaster Amazon ECS, definisi tugas, layanan. Kemudian, AWS CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

AWS CloudFormation juga memungkinkan Anda untuk menggunakan kembali template Anda untuk mengatur sumber daya Amazon ECS Anda secara konsisten dan berulang. Anda mendeskripsikan sumber daya Anda satu kali dan kemudian menyediakan sumber daya yang sama lagi di beberapa Akun AWS dan Wilayah AWS.

AWS CloudFormation template dapat digunakan dengan keduanya AWS Management Console atau AWS Command Line Interface untuk membuat sumber daya.

Untuk mempelajari selengkapnya AWS CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormation Panduan Pengguna
- AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah

Topik

- AWS CloudFormation contoh template untuk Amazon ECS
- Membuat sumber daya Amazon ECS menggunakan konsol AWS CloudFormation
- Membuat sumber daya Amazon ECS menggunakan AWS CLI perintah untuk AWS CloudFormation

AWS CloudFormation contoh template untuk Amazon ECS

Untuk menyediakan dan mengonfigurasi sumber daya untuk Amazon ECS dan layanan terkait, pastikan Anda terbiasa dengan AWS CloudFormation templat. Untuk informasi selengkapnya, lihat AWS CloudFormation templat di Panduan AWS CloudFormation Pengguna. AWS CloudFormation template adalah file teks dalam format JSON atau YAMB yang menjelaskan sumber daya yang ingin Anda sediakan di tumpukan Anda AWS CloudFormation . Jika Anda tidak terbiasa dengan format JSON atau YAMM, atau keduanya, Anda dapat menggunakannya AWS Infrastructure Composer untuk mulai menggunakan templat. AWS CloudFormation Untuk informasi selengkapnya, lihat Membuat template secara visual dengan Infrastructure Composer di Panduan AWS CloudFormation Pengguna.

Anda dapat membuat klaster Amazon ECS, definisi tugas, dan layanan menggunakan. AWS CloudFormation Topik berikut mencakup template yang menunjukkan cara membuat sumber daya dengan konfigurasi yang berbeda. Anda dapat membuat sumber daya ini dengan templat ini dengan menggunakan AWS CloudFormation konsol atau AWS CLI. Untuk informasi selengkapnya tentang cara membuat sumber daya menggunakan AWS CloudFormation konsol, lihat Panduan AWS CloudFormation Pengguna.

Topik berikut mencantumkan contoh template untuk definisi tugas Amazon ECS, cluster, dan layanan.

Topik

- Ketentuan tugas
- Klaster
- Layanan
- Peran IAM untuk Amazon ECS

Ketentuan tugas

Definisi tugas adalah cetak biru untuk aplikasi Anda yang menjelaskan parameter dan satu atau beberapa kontainer yang membentuk aplikasi Anda. Berikut ini adalah contoh AWS CloudFormation template untuk definisi tugas Amazon ECS. Untuk informasi selengkapnya tentang definisi tugas Amazon ECS, lihatDefinisi tugas Amazon ECS.

Definisi tugas Fargate Linux

Anda dapat menggunakan template berikut untuk membuat contoh tugas Fargate Linux.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "ECS Task Definition with parameterized values",
  "Parameters": {
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/docker/library/httpd:2.4",
      "Description": "The container image to use for the task"
    },
    "ContainerCpu": {
      "Type": "Number",
      "Default": 256,
      "Description": "The number of CPU units to reserve for the container",
      "AllowedValues": [256, 512, 1024, 2048, 4096]
    },
    "ContainerMemory": {
      "Type": "Number",
      "Default": 512,
      "Description": "The amount of memory (in MiB) to reserve for the container",
      "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
    },
    "TaskFamily": {
      "Type": "String",
      "Default": "task-definition-cfn",
      "Description": "The name of the task definition family"
    },
    "ContainerName": {
      "Type": "String",
      "Default": "sample-fargate-app",
      "Description": "The name of the container"
```

```
    },
    "ContainerPort": {
      "Type": "Number",
      "Default": 80,
      "Description": "The port number on the container"
    },
    "HostPort": {
      "Type": "Number",
      "Default": 80,
      "Description": "The port number on the host"
    },
    "ExecutionRoleArn": {
      "Type": "String",
      "Default": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
      "Description": "The ARN of the task execution role"
    },
    "LogGroup": {
      "Type": "String",
      "Default": "/ecs/fargate-task-definition",
      "Description": "The CloudWatch log group for container logs"
    },
    "NetworkMode": {
      "Type": "String",
      "Default": "awsvpc",
      "Description": "The Docker networking mode to use",
      "AllowedValues": ["awsvpc", "bridge", "host", "none"]
    },
    "OperatingSystemFamily": {
      "Type": "String",
      "Default": "LINUX",
      "Description": "The operating system for the task",
      "AllowedValues": ["LINUX", "WINDOWS_SERVER_2019_FULL",
 "WINDOWS_SERVER_2019_CORE", "WINDOWS_SERVER_2022_FULL", "WINDOWS_SERVER_2022_CORE"]
    }
  },
  "Resources": {
    "ECSTaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "ContainerDefinitions": [
          {
            "Command": [
              "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
```

```
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && -
foreground\""
            ],
            "EntryPoint": [
              "sh",
              "-c"
            ],
            "Essential": true,
            "Image": {"Ref": "ContainerImage"},
            "LogConfiguration": {
              "LogDriver": "awslogs",
              "Options": {
                "mode": "non-blocking",
                "max-buffer-size": "25m",
                "awslogs-create-group": "true",
                "awslogs-group": {"Ref": "LogGroup"},
                "awslogs-region": {"Ref": "AWS::Region"},
                "awslogs-stream-prefix": "ecs"
              }
            },
            "Name": {"Ref": "ContainerName"},
            "PortMappings": [
              {
                "ContainerPort": {"Ref": "ContainerPort"},
                "HostPort": {"Ref": "HostPort"},
                "Protocol": "tcp"
              }
            ]
          }
        ],
        "Cpu": {"Ref": "ContainerCpu"},
        "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
        "Family": {"Ref": "TaskFamily"},
        "Memory": {"Ref": "ContainerMemory"},
        "NetworkMode": {"Ref": "NetworkMode"},
        "RequiresCompatibilities": [
          "FARGATE"
        ],
        "RuntimePlatform": {
          "OperatingSystemFamily": {"Ref": "OperatingSystemFamily"}
        }
      }
```

```
    }
  },
  "Outputs": {
    "TaskDefinitionArn": {
      "Description": "The ARN of the created task definition",
      "Value": {"Ref": "ECSTaskDefinition"}
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: 'ECS Task Definition to deploy a sample app'
Parameters:
  ContainerImage:
    Type: String
    Default: 'public.ecr.aws/docker/library/httpd:2.4'
    Description: The container image to use for the task
  ContainerCpu:
    Type: Number
    Default: 256
    Description: The number of CPU units to reserve for the container
    AllowedValues: [256, 512, 1024, 2048, 4096]
  ContainerMemory:
    Type: Number
    Default: 512
    Description: The amount of memory (in MiB) to reserve for the container
    AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
  TaskFamily:
    Type: String
    Default: 'task-definition-cfn'
    Description: The name of the task definition family
  ContainerName:
    Type: String
    Default: 'sample-fargate-app'
    Description: The name of the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: The port number on the container
  HostPort:
    Type: Number
```

```
    Default: 80
    Description: The port number on the host
  ExecutionRoleArn:
    Type: String
    Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'
    Description: The ARN of the task execution role
  LogGroup:
    Type: String
    Default: '/ecs/fargate-task-definition'
    Description: The CloudWatch log group for container logs
  NetworkMode:
    Type: String
    Default: 'awsvpc'
    Description: The Docker networking mode to use
    AllowedValues: ['awsvpc', 'bridge', 'host', 'none']
  OperatingSystemFamily:
    Type: String
    Default: 'LINUX'
    Description: The operating system for the task
    AllowedValues: ['LINUX', 'WINDOWS_SERVER_2019_FULL', 'WINDOWS_SERVER_2019_CORE',
 'WINDOWS_SERVER_2022_FULL', 'WINDOWS_SERVER_2022_CORE']
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: !Ref ContainerImage
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
```

```
              max-buffer-size: 25m
              awslogs-create-group: 'true'
              awslogs-group: !Ref LogGroup
              awslogs-region: !Ref AWS::Region
              awslogs-stream-prefix: ecs
          Name: !Ref ContainerName
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              HostPort: !Ref HostPort
              Protocol: tcp
      Cpu: !Ref ContainerCpu
      ExecutionRoleArn: !Ref ExecutionRoleArn
      Family: !Ref TaskFamily
      Memory: !Ref ContainerMemory
      NetworkMode: !Ref NetworkMode
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: !Ref OperatingSystemFamily
Outputs:
  TaskDefinitionArn:
    Description: The ARN of the created task definition
    Value: !Ref ECSTaskDefinition
```

Definisi tugas Amazon EFS

Anda dapat menggunakan template berikut untuk membuat tugas yang menggunakan sistem file Amazon EFS yang Anda buat. Untuk informasi selengkapnya tentang menggunakan volume Amazon EFS dengan Amazon ECS, lihatGunakan volume Amazon EFS dengan Amazon ECS.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a task definition for a web server with parameterized
 values.",
  "Parameters": {
    "ExecutionRoleArn": {
      "Type": "String",
      "Default": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
      "Description": "The ARN of the task execution role"
    },
```

```
    "NetworkMode": {
      "Type": "String",
      "Default": "awsvpc",
      "Description": "The Docker networking mode to use",
      "AllowedValues": ["awsvpc", "bridge", "host", "none"]
    },
    "TaskFamily": {
      "Type": "String",
      "Default": "my-ecs-task",
      "Description": "The name of the task definition family"
    },
    "ContainerCpu": {
      "Type": "String",
      "Default": "256",
      "Description": "The number of CPU units to reserve for the container",
      "AllowedValues": ["256", "512", "1024", "2048", "4096"]
    },
    "ContainerMemory": {
      "Type": "String",
      "Default": "512",
      "Description": "The amount of memory (in MiB) to reserve for the container",
      "AllowedValues": ["512", "1024", "2048", "3072", "4096", "5120", "6144",
"7168", "8192"]
    },
    "ContainerName": {
      "Type": "String",
      "Default": "nginx",
      "Description": "The name of the container"
    },
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/nginx/nginx:latest",
      "Description": "The container image to use for the task"
    },
    "ContainerPort": {
      "Type": "Number",
      "Default": 80,
      "Description": "The port number on the container"
    },
    "InitProcessEnabled": {
      "Type": "String",
      "Default": "true",
      "Description": "Whether to enable the init process inside the container",
      "AllowedValues": ["true", "false"]
```

```
    },
    "EfsVolumeName": {
      "Type": "String",
      "Default": "efs-volume",
      "Description": "The name of the EFS volume"
    },
    "EfsContainerPath": {
      "Type": "String",
      "Default": "/usr/share/nginx/html",
      "Description": "The path in the container where the EFS volume will be
mounted"
    },
    "LogGroup": {
      "Type": "String",
      "Default": "LogGroup",
      "Description": "The CloudWatch log group for container logs"
    },
    "LogStreamPrefix": {
      "Type": "String",
      "Default": "efs-task",
      "Description": "The prefix for the log stream"
    },
    "EfsFilesystemId": {
      "Type": "String",
      "Default": "fs-1234567890abcdef0",
      "Description": "The ID of the EFS filesystem"
    },
    "EfsRootDirectory": {
      "Type": "String",
      "Default": "/",
      "Description": "The root directory in the EFS filesystem"
    },
    "EfsTransitEncryption": {
      "Type": "String",
      "Default": "ENABLED",
      "Description": "Whether to enable transit encryption for EFS",
      "AllowedValues": ["ENABLED", "DISABLED"]
    }
  },
  "Resources": {
    "ECSTaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
```

```
        "NetworkMode": {"Ref": "NetworkMode"},
        "RequiresCompatibilities": ["FARGATE"],
        "Family": {"Ref": "TaskFamily"},
        "Cpu": {"Ref": "ContainerCpu"},
        "Memory": {"Ref": "ContainerMemory"},
        "ContainerDefinitions": [
          {
            "Name": {"Ref": "ContainerName"},
            "Image": {"Ref": "ContainerImage"},
            "Essential": true,
            "PortMappings": [
              {
                "ContainerPort": {"Ref": "ContainerPort"},
                "Protocol": "tcp"
              }
            ],
            "LinuxParameters": {
              "InitProcessEnabled": {"Ref": "InitProcessEnabled"}
            },
            "MountPoints": [
              {
                "SourceVolume": {"Ref": "EfsVolumeName"},
                "ContainerPath": {"Ref": "EfsContainerPath"}
              }
            ],
            "LogConfiguration": {
              "LogDriver": "awslogs",
              "Options": {
                "mode": "non-blocking",
                "max-buffer-size": "25m",
                "awslogs-group": {"Ref": "LogGroup"},
                "awslogs-region": {"Ref": "AWS::Region"},
                "awslogs-create-group": "true",
                "awslogs-stream-prefix": {"Ref": "LogStreamPrefix"}
              }
            }
          }
        ],
        "Volumes": [
          {
            "Name": {"Ref": "EfsVolumeName"},
            "EFSVolumeConfiguration": {
              "FilesystemId": {"Ref": "EfsFilesystemId"},
              "RootDirectory": {"Ref": "EfsRootDirectory"},
```

```
              "TransitEncryption": {"Ref": "EfsTransitEncryption"}
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "TaskDefinitionArn": {
      "Description": "The ARN of the created task definition",
      "Value": {"Ref": "ECSTaskDefinition"}
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a task definition for a web server with parameterized values.
Parameters:
  ExecutionRoleArn:
    Type: String
    Default: arn:aws:iam::123456789012:role/ecsTaskExecutionRole
    Description: The ARN of the task execution role
  NetworkMode:
    Type: String
    Default: awsvpc
    Description: The Docker networking mode to use
    AllowedValues: [awsvpc, bridge, host, none]
  TaskFamily:
    Type: String
    Default: my-ecs-task
    Description: The name of the task definition family
  ContainerCpu:
    Type: String
    Default: "256"
    Description: The number of CPU units to reserve for the container
    AllowedValues: ["256", "512", "1024", "2048", "4096"]
  ContainerMemory:
    Type: String
    Default: "512"
    Description: The amount of memory (in MiB) to reserve for the container
```

```
    AllowedValues: ["512", "1024", "2048", "3072", "4096", "5120", "6144", "7168",
"8192"]
  ContainerName:
    Type: String
    Default: nginx
    Description: The name of the container
  ContainerImage:
    Type: String
    Default: public.ecr.aws/nginx/nginx:latest
    Description: The container image to use for the task
  ContainerPort:
    Type: Number
    Default: 80
    Description: The port number on the container
  InitProcessEnabled:
    Type: String
    Default: "true"
    Description: Whether to enable the init process inside the container
    AllowedValues: ["true", "false"]
  EfsVolumeName:
    Type: String
    Default: efs-volume
    Description: The name of the EFS volume
  EfsContainerPath:
    Type: String
    Default: /usr/share/nginx/html
    Description: The path in the container where the EFS volume will be mounted
  LogGroup:
    Type: String
    Default: LogGroup
    Description: The CloudWatch log group for container logs
  LogStreamPrefix:
    Type: String
    Default: efs-task
    Description: The prefix for the log stream
  EfsFilesystemId:
    Type: String
    Default: fs-1234567890abcdef0
    Description: The ID of the EFS filesystem
  EfsRootDirectory:
    Type: String
    Default: /
    Description: The root directory in the EFS filesystem
  EfsTransitEncryption:
```

```
    Type: String
    Default: ENABLED
    Description: Whether to enable transit encryption for EFS
    AllowedValues: [ENABLED, DISABLED]
Resources:
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ExecutionRoleArn: !Ref ExecutionRoleArn
      NetworkMode: !Ref NetworkMode
      RequiresCompatibilities:
        - FARGATE
      Family: !Ref TaskFamily
      Cpu: !Ref ContainerCpu
      Memory: !Ref ContainerMemory
      ContainerDefinitions:
        - Name: !Ref ContainerName
          Image: !Ref ContainerImage
          Essential: true
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              Protocol: tcp
          LinuxParameters:
            InitProcessEnabled: !Ref InitProcessEnabled
          MountPoints:
            - SourceVolume: !Ref EfsVolumeName
              ContainerPath: !Ref EfsContainerPath
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-group: !Ref LogGroup
              awslogs-region: !Ref AWS::Region
              awslogs-create-group: "true"
              awslogs-stream-prefix: !Ref LogStreamPrefix
      Volumes:
        - Name: !Ref EfsVolumeName
          EFSVolumeConfiguration:
            FilesystemId: !Ref EfsFilesystemId
            RootDirectory: !Ref EfsRootDirectory
            TransitEncryption: !Ref EfsTransitEncryption
Outputs:
  TaskDefinitionArn:
```

```
    Description: The ARN of the created task definition
    Value: !Ref ECSTaskDefinition
```

Klaster

Cluster Amazon ECS adalah pengelompokan tugas atau layanan yang logis. Anda dapat menggunakan template berikut untuk membuat cluster dengan konfigurasi yang berbeda. Untuk informasi selengkapnya tentang klaster Amazon ECS, lihat. Klaster-klaster Amazon ECS

Buat cluster kosong dengan pengaturan default

Anda dapat menggunakan template berikut untuk membuat cluster kosong dengan pengaturan default.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "MyEmptyCluster"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: MyEmptyCluster
```

Buat klaster kosong dengan enkripsi penyimpanan terkelola dan Wawasan Kontainer yang disempurnakan

Anda dapat menggunakan template berikut untuk membuat klaster dengan penyimpanan terkelola tingkat cluster dan Wawasan Kontainer yang disempurnakan diaktifkan. Enkripsi tingkat cluster berlaku untuk volume data terkelola Amazon ECS seperti volume Amazon EBS. Untuk informasi selengkapnya tentang enkripsi Amazon EBS, lihatEnkripsi data yang disimpan dalam volume Amazon EBS yang dilampirkan ke tugas Amazon ECS. Untuk informasi selengkapnya tentang penggunaan Wawasan Kontainer dengan observabilitas terpesona, lihat. Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
        "Cluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "EncryptedEnhancedCluster",
                "ClusterSettings": [
                    {
                        "Name": "containerInsights",
                        "Value": "enhanced"
                    }
                ],
                "Configuration": {
                    "ManagedStorageConfiguration": {
                        "KmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
                    }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Cluster:
    Type: AWS::ECS::Cluster
```

```
    Properties:
      ClusterName: EncryptedEnhancedCluster
      ClusterSettings:
        - Name: containerInsights
          Value: enhanced
      Configuration:
        ManagedStorageConfiguration:
          KmsKeyId: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

Buat cluster dengan Amazon AL2 023 ECS-Optimized-AMI

Anda dapat menggunakan template berikut untuk membuat klaster yang menggunakan penyedia kapasitas yang meluncurkan AL2 023 instans di Amazon. EC2

> **Important**
>
> Untuk AMI terbaru IDs, lihat AMI yang dioptimalkan Amazon ECS di Panduan Pengembang Layanan Kontainer Elastis Amazon.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances
 yet. An ECS capacity provider automatically launches more EC2 instances as required
 on the fly when you request ECS to launch services or standalone tasks.",
    "Parameters": {
        "InstanceType": {
            "Type": "String",
            "Description": "EC2 instance type",
            "Default": "t2.medium",
            "AllowedValues": [
                "t1.micro",
                "t2.2xlarge",
                "t2.large",
                "t2.medium",
                "t2.micro",
                "t2.nano",
                "t2.small",
                "t2.xlarge",
```

```
                "t3.2xlarge",
                "t3.large",
                "t3.medium",
                "t3.micro",
                "t3.nano",
                "t3.small",
                "t3.xlarge"
            ]
        },
        "DesiredCapacity": {
            "Type": "Number",
            "Default": "0",
            "Description": "Number of EC2 instances to launch in your ECS cluster."
        },
        "MaxSize": {
            "Type": "Number",
            "Default": "100",
            "Description": "Maximum number of EC2 instances that can be launched in
 your ECS cluster."
        },
        "ECSAMI": {
            "Description": "The Amazon Machine Image ID used for the cluster",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id"
        },
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "VPC ID where the ECS cluster is launched",
            "Default": "vpc-1234567890abcdef0"
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "List of subnet IDs where the EC2 instances will be
 launched",
            "Default": "subnet-021345abcdef67890"
        }
    },
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterSettings": [
                    {
```

```
                    "Name": "containerInsights",
                    "Value": "enabled"
                }
            ]
        }
    },
    "ECSAutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "DependsOn": [
            "ECSCluster",
            "EC2Role"
        ],
        "Properties": {
            "VPCZoneIdentifier": {
                "Ref": "SubnetIds"
            },
            "LaunchTemplate": {
                "LaunchTemplateId": {
                    "Ref": "ContainerInstances"
                },
                "Version": {
                    "Fn::GetAtt": [
                        "ContainerInstances",
                        "LatestVersionNumber"
                    ]
                }
            },
            "MinSize": 0,
            "MaxSize": {
                "Ref": "MaxSize"
            },
            "DesiredCapacity": {
                "Ref": "DesiredCapacity"
            },
            "NewInstancesProtectedFromScaleIn": true
        },
        "UpdatePolicy": {
            "AutoScalingReplacingUpdate": {
                "WillReplace": "true"
            }
        }
    },
    "ContainerInstances": {
        "Type": "AWS::EC2::LaunchTemplate",
```

```
            "Properties": {
                "LaunchTemplateName": "asg-launch-template-2",
                "LaunchTemplateData": {
                    "ImageId": {
                        "Ref": "ECSAMI"
                    },
                    "InstanceType": {
                        "Ref": "InstanceType"
                    },
                    "IamInstanceProfile": {
                        "Name": {
                            "Ref": "EC2InstanceProfile"
                        }
                    },
                    "SecurityGroupIds": [
                        {
                            "Ref": "ContainerHostSecurityGroup"
                        }
                    ],
                    "UserData": {
                        "Fn::Base64": {
                            "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=
${ECSCluster} >> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/
bin/cfn-init -v --stack ${AWS::StackId} --resource ContainerInstances --configsets
 full_install --region ${AWS::Region} &\n"
                        }
                    },
                    "MetadataOptions": {
                        "HttpEndpoint": "enabled",
                        "HttpTokens": "required"
                    }
                }
            }
        },
        "EC2InstanceProfile": {
            "Type": "AWS::IAM::InstanceProfile",
            "Properties": {
                "Path": "/",
                "Roles": [
                    {
                        "Ref": "EC2Role"
                    }
                ]
            }
```

```
        },
        "CapacityProvider": {
            "Type": "AWS::ECS::CapacityProvider",
            "Properties": {
                "AutoScalingGroupProvider": {
                    "AutoScalingGroupArn": {
                        "Ref": "ECSAutoScalingGroup"
                    },
                    "ManagedScaling": {
                        "InstanceWarmupPeriod": 60,
                        "MinimumScalingStepSize": 1,
                        "MaximumScalingStepSize": 100,
                        "Status": "ENABLED",
                        "TargetCapacity": 100
                    },
                    "ManagedTerminationProtection": "ENABLED"
                }
            }
        },
        "CapacityProviderAssociation": {
            "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
            "Properties": {
                "CapacityProviders": [
                    {
                        "Ref": "CapacityProvider"
                    }
                ],
                "Cluster": {
                    "Ref": "ECSCluster"
                },
                "DefaultCapacityProviderStrategy": [
                    {
                        "Base": 0,
                        "CapacityProvider": {
                            "Ref": "CapacityProvider"
                        },
                        "Weight": 1
                    }
                ]
            }
        },
        "ContainerHostSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
```

```
                "GroupDescription": "Access to the EC2 hosts that run containers",
                "VpcId": {
                    "Ref": "VpcId"
                }
            }
        },
        "EC2Role": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Path": "/",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                    "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
                ]
            }
        },
        "ECSTaskExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ecs-tasks.amazonaws.com"
                                ]
                            },
```

```
                            "Action": [
                                "sts:AssumeRole"
                            ],
                            "Condition": {
                                "ArnLike": {
                                    "aws:SourceArn": {
                                        "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
                                    }
                                },
                                "StringEquals": {
                                    "aws:SourceAccount": {
                                        "Fn::Sub": "${AWS::AccountId}"
                                    }
                                }
                            }
                        }
                    ]
                },
                "Path": "/",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The ECS cluster into which to launch resources",
            "Value": "ECSCluster"
        },
        "ECSTaskExecutionRole": {
            "Description": "The role used to start up a task",
            "Value": "ECSTaskExecutionRole"
        },
        "CapacityProvider": {
            "Description": "The cluster capacity provider that the service should
 use to request capacity when it wants to start up a task",
            "Value": "CapacityProvider"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet. An
 ECS capacity provider automatically launches more EC2 instances as required on the
 fly when you request ECS to launch services or standalone tasks.
Parameters:
  InstanceType:
    Type: String
    Description: EC2 instance type
    Default: t2.medium
    AllowedValues:
      - t1.micro
      - t2.2xlarge
      - t2.large
      - t2.medium
      - t2.micro
      - t2.nano
      - t2.small
      - t2.xlarge
      - t3.2xlarge
      - t3.large
      - t3.medium
      - t3.micro
      - t3.nano
      - t3.small
      - t3.xlarge
  DesiredCapacity:
    Type: Number
    Default: '0'
    Description: Number of EC2 instances to launch in your ECS cluster.
  MaxSize:
    Type: Number
    Default: '100'
    Description: Maximum number of EC2 instances that can be launched in your ECS
 cluster.
  ECSAMI:
    Description: The Amazon Machine Image ID used for the cluster
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID where the ECS cluster is launched
    Default: vpc-1234567890abcdef0
```

```
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: List of subnet IDs where the EC2 instances will be launched
    Default: subnet-021345abcdef67890
Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterSettings:
        - Name: containerInsights
          Value: enabled
  ECSAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
      - ECSCluster
      - EC2Role
    Properties:
      VPCZoneIdentifier: !Ref SubnetIds
      LaunchTemplate:
        LaunchTemplateId: !Ref ContainerInstances
        Version: !GetAtt ContainerInstances.LatestVersionNumber
      MinSize: 0
      MaxSize: !Ref MaxSize
      DesiredCapacity: !Ref DesiredCapacity
      NewInstancesProtectedFromScaleIn: true
    UpdatePolicy:
      AutoScalingReplacingUpdate:
        WillReplace: 'true'
  ContainerInstances:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: asg-launch-template-2
      LaunchTemplateData:
        ImageId: !Ref ECSAMI
        InstanceType: !Ref InstanceType
        IamInstanceProfile:
          Name: !Ref EC2InstanceProfile
        SecurityGroupIds:
          - !Ref ContainerHostSecurityGroup
        UserData: !Base64
          Fn::Sub: |
            #!/bin/bash -xe
            echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
            yum install -y aws-cfn-bootstrap
```

```
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
 EC2InstanceProfile:
   Type: AWS::IAM::InstanceProfile
   Properties:
     Path: /
     Roles:
       - !Ref EC2Role
 CapacityProvider:
   Type: AWS::ECS::CapacityProvider
   Properties:
     AutoScalingGroupProvider:
       AutoScalingGroupArn: !Ref ECSAutoScalingGroup
       ManagedScaling:
         InstanceWarmupPeriod: 60
         MinimumScalingStepSize: 1
         MaximumScalingStepSize: 100
         Status: ENABLED
         TargetCapacity: 100
       ManagedTerminationProtection: ENABLED
 CapacityProviderAssociation:
   Type: AWS::ECS::ClusterCapacityProviderAssociations
   Properties:
     CapacityProviders:
       - !Ref CapacityProvider
     Cluster: !Ref ECSCluster
     DefaultCapacityProviderStrategy:
       - Base: 0
         CapacityProvider: !Ref CapacityProvider
         Weight: 1
 ContainerHostSecurityGroup:
   Type: AWS::EC2::SecurityGroup
   Properties:
     GroupDescription: Access to the EC2 hosts that run containers
     VpcId: !Ref VpcId
 EC2Role:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Statement:
         - Effect: Allow
```

```
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
            Action:
              - sts:AssumeRole
            Condition:
              ArnLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task
    Value: CapacityProvider
```

Layanan

Anda dapat menggunakan layanan Amazon ECS untuk menjalankan dan mempertahankan jumlah instans definisi tugas yang ditentukan secara bersamaan di kluster Amazon ECS. Jika salah satu tugas Anda gagal atau berhenti, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantikannya. Ini membantu mempertahankan jumlah tugas yang Anda inginkan dalam layanan. Template berikut dapat digunakan untuk menyebarkan layanan. Untuk informasi selengkapnya tentang layanan Amazon ECS, lihatLayanan-layanan Amazon ECS.

Menyebarkan aplikasi

Template berikut membuat layanan Amazon ECS yang diberi nama `cfn-service` yang meluncurkan dan memelihara tugas yang ditentukan oleh definisi tugas. Template membuat cluster yang diberi nama`CFNCluster`. Cluster berisi definisi tugas Fargate Linux yang mengatur server web. Template juga menciptakan sumber daya jaringan dan logging yang diperlukan. Template juga membuat peran eksekusi tugas Amazon ECS. Untuk informasi selengkapnya tentang peran pelaksanaan tugas, lihat Peran IAM eksekusi tugas Amazon ECS.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A template that deploys an application that is built on an
 Apache web server Docker image by creating an Amazon ECS cluster, task definition,
 and service. The template also creates networking and logging resources, and an
 Amazon ECS task execution role.",
    "Parameters": {
        "ClusterName": {
            "Type": "String",
            "Default": "CFNCluster",
            "Description": "Name of the ECS Cluster"
        },
        "TaskFamily": {
            "Type": "String",
            "Default": "task-definition-cfn",
            "Description": "Family name for the Task Definition"
        },
        "ServiceName": {
            "Type": "String",
            "Default": "cfn-service",
            "Description": "Name of the ECS Service"
```

```
        },
        "ContainerImage": {
            "Type": "String",
            "Default": "public.ecr.aws/docker/library/httpd:2.4",
            "Description": "Container image to use for the task"
        },
        "TaskCpu": {
            "Type": "Number",
            "Default": 256,
            "AllowedValues": [
                256,
                512,
                1024,
                2048,
                4096
            ],
            "Description": "CPU units for the task"
        },
        "TaskMemory": {
            "Type": "Number",
            "Default": 512,
            "AllowedValues": [
                512,
                1024,
                2048,
                4096,
                8192,
                16384
            ],
            "Description": "Memory (in MiB) for the task"
        },
        "DesiredCount": {
            "Type": "Number",
            "Default": 1,
            "Description": "Desired number of tasks to run"
        },
        "LogGroupName": {
            "Type": "String",
            "Default": "/ecs/fargate-task-definition",
            "Description": "CloudWatch Log Group name"
        },
        "VpcCidr": {
            "Type": "String",
            "Default": "10.0.0.0/16",
```

```
            "Description": "CIDR block for the VPC"
        },
        "PublicSubnet1Cidr": {
            "Type": "String",
            "Default": "10.0.0.0/24",
            "Description": "CIDR block for public subnet 1"
        },
        "PublicSubnet2Cidr": {
            "Type": "String",
            "Default": "10.0.1.0/24",
            "Description": "CIDR block for public subnet 2"
        }
    },
    "Resources": {
        "VPC": {
            "Type": "AWS::EC2::VPC",
            "Properties": {
                "CidrBlock": {
                    "Ref": "VpcCidr"
                },
                "EnableDnsSupport": true,
                "EnableDnsHostnames": true,
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {
                            "Fn::Sub": "${AWS::StackName}-VPC"
                        }
                    }
                ]
            }
        },
        "InternetGateway": {
            "Type": "AWS::EC2::InternetGateway",
            "Properties": {
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {
                            "Fn::Sub": "${AWS::StackName}-IGW"
                        }
                    }
                ]
            }
```

```
        },
        "InternetGatewayAttachment": {
            "Type": "AWS::EC2::VPCGatewayAttachment",
            "Properties": {
                "InternetGatewayId": {
                    "Ref": "InternetGateway"
                },
                "VpcId": {
                    "Ref": "VPC"
                }
            }
        },
        "PublicSubnet1": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "VpcId": {
                    "Ref": "VPC"
                },
                "AvailabilityZone": {
                    "Fn::Select": [
                        0,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                "CidrBlock": {
                    "Ref": "PublicSubnet1Cidr"
                },
                "MapPublicIpOnLaunch": true,
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {
                            "Fn::Sub": "${AWS::StackName}-PublicSubnet1"
                        }
                    }
                ]
            }
        },
        "PublicSubnet2": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "VpcId": {
```

```
                    "Ref": "VPC"
                },
                "AvailabilityZone": {
                    "Fn::Select": [
                        1,
                        {
                            "Fn::GetAZs": ""
                        }
                    ]
                },
                "CidrBlock": {
                    "Ref": "PublicSubnet2Cidr"
                },
                "MapPublicIpOnLaunch": true,
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {
                            "Fn::Sub": "${AWS::StackName}-PublicSubnet2"
                        }
                    }
                ]
            }
        },
        "PublicRouteTable": {
            "Type": "AWS::EC2::RouteTable",
            "Properties": {
                "VpcId": {
                    "Ref": "VPC"
                },
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {
                            "Fn::Sub": "${AWS::StackName}-PublicRouteTable"
                        }
                    }
                ]
            }
        },
        "DefaultPublicRoute": {
            "Type": "AWS::EC2::Route",
            "DependsOn": "InternetGatewayAttachment",
            "Properties": {
```

```
                "RouteTableId": {
                    "Ref": "PublicRouteTable"
                },
                "DestinationCidrBlock": "0.0.0.0/0",
                "GatewayId": {
                    "Ref": "InternetGateway"
                }
            }
        },
        "PublicSubnet1RouteTableAssociation": {
            "Type": "AWS::EC2::SubnetRouteTableAssociation",
            "Properties": {
                "RouteTableId": {
                    "Ref": "PublicRouteTable"
                },
                "SubnetId": {
                    "Ref": "PublicSubnet1"
                }
            }
        },
        "PublicSubnet2RouteTableAssociation": {
            "Type": "AWS::EC2::SubnetRouteTableAssociation",
            "Properties": {
                "RouteTableId": {
                    "Ref": "PublicRouteTable"
                },
                "SubnetId": {
                    "Ref": "PublicSubnet2"
                }
            }
        },
        "ECSSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Security group for ECS tasks",
                "VpcId": {
                    "Ref": "VPC"
                },
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
```

```
                    },
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 443,
                        "ToPort": 443,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        },
        "ECSTaskExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "ecs-tasks.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        },
        "TaskLogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "DeletionPolicy": "Retain",
            "UpdateReplacePolicy": "Retain",
            "Properties": {
                "LogGroupName": {
                    "Ref": "LogGroupName"
                },
                "RetentionInDays": 30
            }
        },
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
```

```
            "Properties": {
                "ClusterName": {
                    "Ref": "ClusterName"
                }
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Command": [
                        "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": {
                            "Ref": "ContainerImage"
                        },
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "mode": "non-blocking",
                                "max-buffer-size": "25m",
                                "awslogs-create-group": "true",
                                "awslogs-group": {
                                    "Ref": "LogGroupName"
                                },
                                "awslogs-region": {
                                    "Ref": "AWS::Region"
                                },
                                "awslogs-stream-prefix": "ecs"
                            }
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
```

```
                        {
                            "ContainerPort": 80,
                            "HostPort": 80,
                            "Protocol": "tcp"
                        }
                    ]
                }
            ],
            "Cpu": {
                "Ref": "TaskCpu"
            },
            "ExecutionRoleArn": {
                "Fn::GetAtt": [
                    "ECSTaskExecutionRole",
                    "Arn"
                ]
            },
            "Family": {
                "Ref": "TaskFamily"
            },
            "Memory": {
                "Ref": "TaskMemory"
            },
            "NetworkMode": "awsvpc",
            "RequiresCompatibilities": [
                "FARGATE"
            ],
            "RuntimePlatform": {
                "OperatingSystemFamily": "LINUX"
            }
        }
    },
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "DependsOn": [
            "PublicSubnet1RouteTableAssociation",
            "PublicSubnet2RouteTableAssociation"
        ],
        "Properties": {
            "ServiceName": {
                "Ref": "ServiceName"
            },
            "Cluster": {
                "Ref": "ECSCluster"
```

```
                },
                "DesiredCount": {
                    "Ref": "DesiredCount"
                },
                "LaunchType": "FARGATE",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "SecurityGroups": [
                            {
                                "Ref": "ECSSecurityGroup"
                            }
                        ],
                        "Subnets": [
                            {
                                "Ref": "PublicSubnet1"
                            },
                            {
                                "Ref": "PublicSubnet2"
                            }
                        ]
                    }
                },
                "TaskDefinition": {
                    "Ref": "ECSTaskDefinition"
                }
            }
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The name of the ECS cluster",
            "Value": {
                "Ref": "ECSCluster"
            }
        },
        "TaskDefinitionArn": {
            "Description": "The ARN of the task definition",
            "Value": {
                "Ref": "ECSTaskDefinition"
            }
        },
        "ServiceName": {
            "Description": "The name of the ECS service",
```

```
            "Value": {
                "Ref": "ECSService"
            }
        },
        "VpcId": {
            "Description": "The ID of the VPC",
            "Value": {
                "Ref": "VPC"
            }
        },
        "PublicSubnet1": {
            "Description": "The ID of public subnet 1",
            "Value": {
                "Ref": "PublicSubnet1"
            }
        },
        "PublicSubnet2": {
            "Description": "The ID of public subnet 2",
            "Value": {
                "Ref": "PublicSubnet2"
            }
        },
        "SecurityGroup": {
            "Description": "The ID of the security group",
            "Value": {
                "Ref": "ECSSecurityGroup"
            }
        },
        "ExecutionRoleArn": {
            "Description": "The ARN of the task execution role",
            "Value": {
                "Fn::GetAtt": [
                    "ECSTaskExecutionRole",
                    "Arn"
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: A template that deploys an application that is built on an Apache
 web server Docker image by creating an Amazon ECS cluster, task definition, and
 service. The template also creates networking and logging resources, and an Amazon
 ECS task execution role.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  TaskFamily:
    Type: String
    Default: task-definition-cfn
    Description: Family name for the Task Definition
  ServiceName:
    Type: String
    Default: cfn-service
    Description: Name of the ECS Service
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
  TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
  DesiredCount:
    Type: Number
    Default: 1
    Description: Desired number of tasks to run
  LogGroupName:
    Type: String
    Default: /ecs/fargate-task-definition
    Description: CloudWatch Log Group name
  VpcCidr:
    Type: String
    Default: 10.0.0.0/16
    Description: CIDR block for the VPC
  PublicSubnet1Cidr:
```

```
    Type: String
    Default: 10.0.0.0/24
    Description: CIDR block for public subnet 1
  PublicSubnet2Cidr:
    Type: String
    Default: 10.0.1.0/24
    Description: CIDR block for public subnet 2
Resources:
  # VPC and Networking Resources
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-VPC
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-IGW
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Ref PublicSubnet1Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet1
  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs '']
```

```
      CidrBlock: !Ref PublicSubnet2Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet2
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicRouteTable
  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
  # Security Group
  ECSSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ECS tasks
      VpcId: !Ref VPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
```

```
# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
# CloudWatch Logs
TaskLogGroup:
  Type: AWS::Logs::LogGroup
  DeletionPolicy: Retain
  UpdateReplacePolicy: Retain
  Properties:
    LogGroupName: !Ref LogGroupName
    RetentionInDays: 30
# ECS Resources
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterName: !Ref ClusterName
ECSTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    ContainerDefinitions:
      - Command:
          - >-
            /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
            App</title> <style>body {margin-top: 40px; background-color:
            #333;} </style> </head><body> <div
            style=color:white;text-align:center> <h1>Amazon ECS Sample
            App</h1> <h2>Congratulations!</h2> <p>Your application is now
            running on a container in Amazon ECS.</p> </div></body></html>' >
            /usr/local/apache2/htdocs/index.html && httpd-foreground"s
        EntryPoint:
          - sh
          - '-c'
        Essential: true
```

```
          Image: !Ref ContainerImage
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-create-group: 'true'
              awslogs-group: !Ref LogGroupName
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: !Ref TaskCpu
      ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
      Family: !Ref TaskFamily
      Memory: !Ref TaskMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: AWS::ECS::Service
    DependsOn:
      - PublicSubnet1RouteTableAssociation
      - PublicSubnet2RouteTableAssociation
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ECSCluster
      DesiredCount: !Ref DesiredCount
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - !Ref ECSSecurityGroup
          Subnets:
            - !Ref PublicSubnet1
            - !Ref PublicSubnet2
      TaskDefinition: !Ref ECSTaskDefinition
Outputs:
```

```
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref ECSTaskDefinition
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ECSService
  VpcId:
    Description: The ID of the VPC
    Value: !Ref VPC
  PublicSubnet1:
    Description: The ID of public subnet 1
    Value: !Ref PublicSubnet1
  PublicSubnet2:
    Description: The ID of public subnet 2
    Value: !Ref PublicSubnet2
  SecurityGroup:
    Description: The ID of the security group
    Value: !Ref ECSSecurityGroup
  ExecutionRoleArn:
    Description: The ARN of the task execution role
    Value: !GetAtt ECSTaskExecutionRole.Arn
```

Menyebarkan layanan dengan ECS Exec diaktifkan

Anda dapat menggunakan template berikut untuk menyebarkan layanan dengan ECS Exec diaktifkan. Layanan berjalan dalam cluster dengan kunci KMS untuk mengenkripsi sesi ECS Exec dan konfigurasi log untuk mengarahkan ulang log sesi perintah eksekusi ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "ECS Cluster with Fargate Service and Task Definition and ECS
 Exec enabled.",
    "Parameters": {
        "ClusterName": {
            "Type": "String",
            "Default": "CFNCluster",
```

```
            "Description": "Name of the ECS Cluster"
        },
        "S3BucketName": {
            "Type": "String",
            "Default": "amzn-s3-demo-bucket",
            "Description": "S3 bucket for ECS execute command logs"
        },
        "KmsKeyId": {
            "Type": "String",
            "Default": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Description": "KMS Key ID for ECS execute command encryption"
        },
        "ContainerImage": {
            "Type": "String",
            "Default": "public.ecr.aws/docker/library/httpd:2.4",
            "Description": "Container image to use for the task"
        },
        "ContainerCpu": {
            "Type": "Number",
            "Default": 256,
            "AllowedValues": [256, 512, 1024, 2048, 4096],
            "Description": "CPU units for the container (256 = 0.25 vCPU)"
        },
        "ContainerMemory": {
            "Type": "Number",
            "Default": 512,
            "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192],
            "Description": "Memory for the container (in MiB)"
        },
        "DesiredCount": {
            "Type": "Number",
            "Default": 1,
            "Description": "Desired count of tasks in the service"
        },
      "SecurityGroups": {
  "Type": "List<AWS::EC2::SecurityGroup::Id>",
  "Description": "Security Group IDs for the ECS Service"
},
"Subnets": {
  "Type": "List<AWS::EC2::Subnet::Id>",
  "Description": "Subnet IDs for the ECS Service"
},
        "ServiceName": {
            "Type": "String",
```

```
            "Default": "cfn-service",
            "Description": "Name of the ECS service"
        },
        "TaskFamily": {
            "Type": "String",
            "Default": "task-definition-cfn",
            "Description": "Family name for the task definition"
        },
        "TaskExecutionRoleArn": {
            "Type": "String",
            "Description": "ARN of an existing IAM role for ECS task execution",
            "Default": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
        },
        "TaskRoleArn": {
            "Type": "String",
            "Description": "ARN of an existing IAM role for ECS tasks",
            "Default": "arn:aws:iam::111122223333:role/execTaskRole"
        }
    },
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": {"Ref": "ClusterName"},
                "Configuration": {
                    "ExecuteCommandConfiguration": {
                        "Logging": "OVERRIDE",
                        "LogConfiguration": {
                            "S3BucketName": {"Ref": "S3BucketName"}
                        },
                        "KmsKeyId": {"Ref": "KmsKeyId"}
                    }
                },
                "Tags": [
                    {
                        "Key": "Environment",
                        "Value": {"Ref": "AWS::StackName"}
                    }
                ]
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
```

```
                "ContainerDefinitions": [
                    {
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
 Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": {"Ref": "ContainerImage"},
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "mode": "non-blocking",
                                "max-buffer-size": "25m",
                                "awslogs-create-group": "true",
                                "awslogs-group": {"Fn::Sub": "/ecs/
${AWS::StackName}"},
                                "awslogs-region": {"Ref": "AWS::Region"},
                                "awslogs-stream-prefix": "ecs"
                            }
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
                            {
                                "ContainerPort": 80,
                                "HostPort": 80,
                                "Protocol": "tcp"
                            }
                        ]
                    }
                ],
                "Cpu": {"Ref": "ContainerCpu"},
                "ExecutionRoleArn": {"Ref": "TaskExecutionRoleArn"},
                "TaskRoleArn": {"Ref": "TaskRoleArn"},
                "Family": {"Ref": "TaskFamily"},
                "Memory": {"Ref": "ContainerMemory"},
                "NetworkMode": "awsvpc",
```

```
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                },
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {"Fn::Sub": "${AWS::StackName}-TaskDefinition"}
                    }
                ]
            }
        },
        "ECSService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "ServiceName": {"Ref": "ServiceName"},
                "Cluster": {"Ref": "ECSCluster"},
                "DesiredCount": {"Ref": "DesiredCount"},
                "EnableExecuteCommand": true,
                "LaunchType": "FARGATE",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "SecurityGroups": {"Ref": "SecurityGroups"},
                        "Subnets": {"Ref": "Subnets"}
                    }
                },
                "TaskDefinition": {"Ref": "ECSTaskDefinition"},
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": {"Fn::Sub": "${AWS::StackName}-Service"}
                    }
                ]
            }
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The name of the ECS cluster",
            "Value": {"Ref": "ECSCluster"}
        },
```

```
        "ServiceName": {
            "Description": "The name of the ECS service",
            "Value": {"Ref": "ServiceName"}
        },
        "TaskDefinitionArn": {
            "Description": "The ARN of the task definition",
            "Value": {"Ref": "ECSTaskDefinition"}
        },
        "ClusterArn": {
            "Description": "The ARN of the ECS cluster",
            "Value": {"Fn::GetAtt": ["ECSCluster", "Arn"]}
        },
        "StackName": {
            "Description": "The name of this stack",
            "Value": {"Ref": "AWS::StackName"}
        },
        "StackId": {
            "Description": "The unique identifier for this stack",
            "Value": {"Ref": "AWS::StackId"}
        },
        "Region": {
            "Description": "The AWS Region where the stack is deployed",
            "Value": {"Ref": "AWS::Region"}
        },
        "AccountId": {
            "Description": "The AWS Account ID",
            "Value": {"Ref": "AWS::AccountId"}
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: ECS Cluster with Fargate Service and Task Definition and ECS Exec
 enabled.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  S3BucketName:
    Type: String
```

```
    Default: amzn-s3-demo-bucket
    Description: S3 bucket for ECS execute command logs
  KmsKeyId:
    Type: String
    Default: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
    Description: KMS Key ID for ECS execute command encryption
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  ContainerCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the container (256 = 0.25 vCPU)
  ContainerMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
    Description: Memory for the container (in MiB)
  DesiredCount:
    Type: Number
    Default: 1
    Description: Desired count of tasks in the service
 SecurityGroups:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service
  Subnets:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service
  ServiceName:
    Type: String
    Default: cfn-service
    Description: Name of the ECS service
  TaskFamily:
    Type: String
    Default: task-definition-cfn
    Description: Family name for the task definition
  TaskExecutionRoleArn:
    Type: String
    Description: ARN of an existing IAM role for ECS task execution
    Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'
  TaskRoleArn:
    Type: String
```

```
    Description: ARN of an existing IAM role for ECS tasks
    Default: 'arn:aws:iam::111122223333:role/execTaskRole'
Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ClusterName
      Configuration:
        ExecuteCommandConfiguration:
          Logging: OVERRIDE
          LogConfiguration:
            S3BucketName: !Ref S3BucketName
          KmsKeyId: !Ref KmsKeyId
      Tags:
        - Key: Environment
          Value: !Ref AWS::StackName
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
             /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: !Ref ContainerImage
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-create-group: 'true'
              awslogs-group: !Sub /ecs/${AWS::StackName}
              awslogs-region: !Ref AWS::Region
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
```

```
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: !Ref ContainerCpu
      ExecutionRoleArn: !Ref TaskExecutionRoleArn
      TaskRoleArn: !Ref TaskRoleArn
      Family: !Ref TaskFamily
      Memory: !Ref ContainerMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-TaskDefinition
  ECSService:
    Type: AWS::ECS::Service
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ECSCluster
      DesiredCount: !Ref DesiredCount
      EnableExecuteCommand: true
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups: !Ref SecurityGroups
          Subnets: !Ref Subnets
      TaskDefinition: !Ref ECSTaskDefinition
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-Service
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ServiceName
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref ECSTaskDefinition
```

```
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  StackName:
    Description: The name of this stack
    Value: !Ref AWS::StackName
  StackId:
    Description: The unique identifier for this stack
    Value: !Ref AWS::StackId
  Region:
    Description: The AWS Region where the stack is deployed
    Value: !Ref AWS::Region
  AccountId:
    Description: The AWS Account ID
    Value: !Ref AWS::AccountId
```

Menyebarkan layanan yang menggunakan Amazon VPC Lattice

Anda dapat menggunakan template berikut untuk memulai membuat layanan Amazon ECS dengan VPC Lattice. Anda mungkin perlu menyelesaikan langkah-langkah tambahan berikut untuk menyiapkan VPC Lattice:

- Perbarui aturan masuk grup keamanan Anda untuk VPC Lattice untuk mengizinkan awalan `vpc-lattice` aturan masuk dan mengizinkan lalu lintas di port 80.
- Kaitkan VPC untuk layanan ke jaringan layanan VPC Lattice.
- Konfigurasikan zona yang dihosting pribadi atau publik dengan Amazon Route 53.
- Konfigurasikan aturan listener dan listener dalam layanan VPC Lattice.
- Verifikasi konfigurasi pemeriksaan kesehatan kelompok sasaran.

Untuk informasi selengkapnya tentang penggunaan Kisi VPC dengan Amazon ECS, lihat. Gunakan Amazon VPC Lattice untuk menghubungkan, mengamati, dan mengamankan layanan Amazon ECS Anda

JSON

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "The template used to create an ECS Service with VPC Lattice.",
 "Parameters": {
```

```
"ECSClusterName": {
 "Type": "String",
 "Default": "vpc-lattice-cluster"
},
"ECSServiceName": {
 "Type": "String",
 "Default": "vpc-lattice-service"
},
"SecurityGroupIDs": {
 "Type": "List<AWS::EC2::SecurityGroup::Id>",
 "Description": "Security Group IDs for the ECS Service"
},
"SubnetIDs": {
 "Type": "List<AWS::EC2::Subnet::Id>",
 "Description": "Subnet IDs for the ECS Service"
},
"VpcID": {
 "Type": "AWS::EC2::VPC::Id",
 "Description": "VPC ID for the resources"
},
"ContainerImage": {
 "Type": "String",
 "Default": "public.ecr.aws/docker/library/httpd:2.4",
 "Description": "Container image to use for the task"
},
"TaskCpu": {
 "Type": "Number",
 "Default": 256,
 "AllowedValues": [256, 512, 1024, 2048, 4096],
 "Description": "CPU units for the task"
},
"TaskMemory": {
 "Type": "Number",
 "Default": 512,
 "AllowedValues": [512, 1024, 2048, 4096, 8192, 16384],
 "Description": "Memory (in MiB) for the task"
},
"LogGroupName": {
 "Type": "String",
 "Default": "/ecs/vpc-lattice-task",
 "Description": "CloudWatch Log Group name"
},
"EnableContainerInsights": {
 "Type": "String",
```

```
    "Default": "enabled",
    "AllowedValues": ["enabled", "disabled"],
    "Description": "Enable or disable CloudWatch Container Insights for the cluster"
   }
  },
  "Resources": {
   "ECSCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {
     "ClusterName": {"Ref": "ECSClusterName"},
     "ClusterSettings": [
      {
       "Name": "containerInsights",
       "Value": {"Ref": "EnableContainerInsights"}
      }
     ],
     "Tags": [
      {
       "Key": "Name",
       "Value": {"Ref": "ECSClusterName"}
      }
     ]
    }
   },
   "ECSTaskExecutionRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
     "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
       {
        "Effect": "Allow",
        "Principal": {
         "Service": "ecs-tasks.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
       }
      ]
     },
     "ManagedPolicyArns": [
      "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
     ]
    }
   },
```

```
 "TaskLogGroup": {
  "Type": "AWS::Logs::LogGroup",
  "DeletionPolicy": "Retain",
  "UpdateReplacePolicy": "Retain",
  "Properties": {
   "LogGroupName": {"Ref": "LogGroupName"},
   "RetentionInDays": 30
  }
 },
 "VpcLatticeTaskDefinition": {
  "Type": "AWS::ECS::TaskDefinition",
  "Properties": {
   "ContainerDefinitions": [
    {
     "Command": [
      "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
<style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
<div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
<h2>Congratulations!</h2> <p>Your application is now running on a container in
Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
httpd-foreground\""
     ],
     "EntryPoint": [
      "sh",
      "-c"
     ],
     "Essential": true,
     "Image": {"Ref": "ContainerImage"},
     "LogConfiguration": {
      "LogDriver": "awslogs",
      "Options": {
       "mode": "non-blocking",
       "max-buffer-size": "25m",
       "awslogs-create-group": "true",
       "awslogs-group": {"Ref": "LogGroupName"},
       "awslogs-region": {"Ref": "AWS::Region"},
       "awslogs-stream-prefix": "ecs"
      }
     },
     "Name": "vpc-lattice-container",
     "PortMappings": [
      {
       "ContainerPort": 80,
       "HostPort": 80,
```

```
        "Protocol": "tcp",
        "Name": "vpc-lattice-port"
       }
      ]
     }
    ],
    "Cpu": {"Ref": "TaskCpu"},
    "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
    "Family": "vpc-lattice-task-definition",
    "Memory": {"Ref": "TaskMemory"},
    "NetworkMode": "awsvpc",
    "RequiresCompatibilities": [
     "FARGATE"
    ],
    "RuntimePlatform": {
     "OperatingSystemFamily": "LINUX"
    }
   }
  },
  "ECSService": {
   "Type": "AWS::ECS::Service",
   "Properties": {
    "Cluster": {"Ref": "ECSCluster"},
    "TaskDefinition": {"Ref": "VpcLatticeTaskDefinition"},
    "LaunchType": "FARGATE",
    "ServiceName": {"Ref": "ECSServiceName"},
    "SchedulingStrategy": "REPLICA",
    "DesiredCount": 2,
    "AvailabilityZoneRebalancing": "ENABLED",
    "NetworkConfiguration": {
     "AwsvpcConfiguration": {
      "AssignPublicIp": "ENABLED",
      "SecurityGroups": {
       "Ref": "SecurityGroupIDs"
      },
      "Subnets": {
       "Ref": "SubnetIDs"
      }
     }
    },
    "PlatformVersion": "LATEST",
    "VpcLatticeConfigurations": [
     {
      "RoleArn": "arn:aws:iam::111122223333:role/ecsInfrastructureRole",
```

```
      "PortName": "vpc-lattice-port",
      "TargetGroupArn": {
       "Ref": "TargetGroup1"
      }
     }
    ],
    "DeploymentConfiguration": {
     "DeploymentCircuitBreaker": {
      "Enable": true,
      "Rollback": true
     },
     "MaximumPercent": 200,
     "MinimumHealthyPercent": 100
    },
    "DeploymentController": {
     "Type": "ECS"
    },
    "ServiceConnectConfiguration": {
     "Enabled": false
    },
    "Tags": [],
    "EnableECSManagedTags": true
   }
  },
  "TargetGroup1": {
   "Type": "AWS::VpcLattice::TargetGroup",
   "Properties": {
    "Type": "IP",
    "Name": "first-target-group",
    "Config": {
     "Port": 80,
     "Protocol": "HTTP",
     "VpcIdentifier": {"Ref": "VpcID"},
     "HealthCheck": {
      "Enabled": true,
      "Path": "/"
     }
    },
    "Tags": [
     {
      "Key": "ecs-application-networking/ServiceName",
      "Value": {"Ref": "ECSServiceName"}
     },
     {
```

```
      "Key": "ecs-application-networking/ClusterName",
      "Value": {"Ref": "ECSClusterName"}
     },
     {
      "Key": "ecs-application-networking/TaskDefinition",
      "Value": {"Ref": "VpcLatticeTaskDefinition"}
     },
     {
      "Key": "ecs-application-networking/VpcId",
      "Value": {"Ref": "VpcID"}
     }
    ]
   }
  }
 },
 "Outputs": {
  "ClusterName": {
   "Description": "The cluster used to create the service.",
   "Value": {
    "Ref": "ECSCluster"
   }
  },
  "ClusterArn": {
   "Description": "The ARN of the ECS cluster",
   "Value": {
    "Fn::GetAtt": ["ECSCluster", "Arn"]
   }
  },
  "ECSService": {
   "Description": "The created service.",
   "Value": {
    "Ref": "ECSService"
   }
  },
  "TaskDefinitionArn": {
   "Description": "The ARN of the task definition",
   "Value": {
    "Ref": "VpcLatticeTaskDefinition"
   }
  }
 }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: The template used to create an ECS Service with VPC Lattice.

Parameters:
  ECSClusterName:
    Type: String
    Default: vpc-lattice-cluster
  ECSServiceName:
    Type: String
    Default: vpc-lattice-service
  SecurityGroupIDs:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service
  SubnetIDs:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service
  VpcID:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID for the resources
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
  TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
  LogGroupName:
    Type: String
    Default: /ecs/vpc-lattice-task
    Description: CloudWatch Log Group name
  EnableContainerInsights:
    Type: String
    Default: 'enhanced'
    AllowedValues: ['enabled', 'disabled', 'enhanced']
    Description: Enable or disable CloudWatch Container Insights for the cluster
```

```
Resources:
  # ECS Cluster
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ECSClusterName
      ClusterSettings:
        - Name: containerInsights
          Value: !Ref EnableContainerInsights
      Tags:
        - Key: Name
          Value: !Ref ECSClusterName

  # IAM Roles
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

  # CloudWatch Logs
  TaskLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      LogGroupName: !Ref LogGroupName
      RetentionInDays: 30

  # Task Definition
  VpcLatticeTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
```

```
            /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
            App</title> <style>body {margin-top: 40px; background-color:
            #333;} </style> </head><body> <div
            style=color:white;text-align:center> <h1>Amazon ECS Sample
            App</h1> <h2>Congratulations!</h2> <p>Your application is now
            running on a container in Amazon ECS.</p> </div></body></html>' >
            /usr/local/apache2/htdocs/index.html && httpd-foreground"
        EntryPoint:
          - sh
          - '-c'
        Essential: true
        Image: !Ref ContainerImage
        LogConfiguration:
          LogDriver: awslogs
          Options:
            mode: non-blocking
            max-buffer-size: 25m
            awslogs-create-group: 'true'
            awslogs-group: !Ref LogGroupName
            awslogs-region: !Ref 'AWS::Region'
            awslogs-stream-prefix: ecs
        Name: vpc-lattice-container
        PortMappings:
          - ContainerPort: 80
            HostPort: 80
            Protocol: tcp
            Name: vpc-lattice-port
    Cpu: !Ref TaskCpu
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
    Family: vpc-lattice-task-definition
    Memory: !Ref TaskMemory
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX

ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref VpcLatticeTaskDefinition
    LaunchType: FARGATE
    ServiceName: !Ref ECSServiceName
```

```
      SchedulingStrategy: REPLICA
      DesiredCount: 2
      AvailabilityZoneRebalancing: ENABLED
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups: !Ref SecurityGroupIDs
          Subnets: !Ref SubnetIDs
      PlatformVersion: LATEST
      VpcLatticeConfigurations:
        - RoleArn: arn:aws:iam::111122223333:role/ecsInfrastructureRole
          PortName: vpc-lattice-port
          TargetGroupArn: !Ref TargetGroup1
      DeploymentConfiguration:
        DeploymentCircuitBreaker:
          Enable: true
          Rollback: true
        MaximumPercent: 200
        MinimumHealthyPercent: 100
      DeploymentController:
        Type: ECS
      ServiceConnectConfiguration:
        Enabled: false
      Tags: []
      EnableECSManagedTags: true

  TargetGroup1:
    Type: AWS::VpcLattice::TargetGroup
    Properties:
      Type: IP
      Name: first-target-group
      Config:
        Port: 80
        Protocol: HTTP
        VpcIdentifier: !Ref VpcID
        HealthCheck:
          Enabled: true
          Path: /
      Tags:
        - Key: ecs-application-networking/ServiceName
          Value: !Ref ECSServiceName
        - Key: ecs-application-networking/ClusterName
          Value: !Ref ECSClusterName
        - Key: ecs-application-networking/TaskDefinition
```

```
          Value: !Ref VpcLatticeTaskDefinition
        - Key: ecs-application-networking/VpcId
          Value: !Ref VpcID

Outputs:
  ClusterName:
    Description: The cluster used to create the service.
    Value: !Ref ECSCluster
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  ECSService:
    Description: The created service.
    Value: !Ref ECSService
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref VpcLatticeTaskDefinition
```

Menyebarkan layanan dengan konfigurasi volume

Template berikut mencakup konfigurasi volume dalam definisi layanan. Amazon ECS mendukung konfigurasi volume data berikut dengan menggunakan konfigurasi volume saat peluncuran: Volume Amazon EBS. Untuk informasi lebih lanjut tentang peristiwa Amazon EBS, lihat Gunakan volume Amazon EBS dengan Amazon ECS.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "The template used to create an ECS Service that includes a
 volume configuration. The configuration is used to create Amazon EBS volumes for
 attachment to the tasks. One volume is attached per task.",
  "Parameters": {
    "ECSClusterName": {
      "Type": "String",
      "Default": "volume-config-cluster",
      "Description": "Name of the ECS cluster"
    },
    "SecurityGroupIDs": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>",
      "Description": "Security Group IDs for the ECS Service"
    },
```

```
    "SubnetIDs": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "Subnet IDs for the ECS Service"
    },
    "InfrastructureRoleArn": {
      "Type": "String",
      "Description": "ARN of the IAM role that ECS will use to manage EBS volumes"
    },
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/nginx/nginx:latest",
      "Description": "Container image to use for the task"
    },
    "TaskCpu": {
      "Type": "String",
      "Default": "2048",
      "Description": "CPU units for the task"
    },
    "TaskMemory": {
      "Type": "String",
      "Default": "4096",
      "Description": "Memory (in MiB) for the task"
    },
    "VolumeSize": {
      "Type": "String",
      "Default": "10",
      "Description": "Size of the EBS volume in GiB"
    },
    "VolumeType": {
      "Type": "String",
      "Default": "gp3",
      "AllowedValues": ["gp2", "gp3", "io1", "io2", "st1", "sc1", "standard"],
      "Description": "EBS volume type"
    },
    "VolumeIops": {
      "Type": "String",
      "Default": "3000",
      "Description": "IOPS for the EBS volume (required for io1, io2, and gp3)"
    },
    "VolumeThroughput": {
      "Type": "String",
      "Default": "125",
      "Description": "Throughput for the EBS volume (only for gp3)"
    },
```

```
    "FilesystemType": {
      "Type": "String",
      "Default": "xfs",
      "AllowedValues": ["xfs", "ext4"],
      "Description": "Filesystem type for the EBS volume"
    },
    "EnableContainerInsights": {
      "Type": "String",
      "Default": "enhanced",
      "AllowedValues": ["enabled", "disabled", "enhanced"],
      "Description": "Enable or disable CloudWatch Container Insights for the
cluster"
    }
  },
  "Resources": {
    "ECSCluster": {
      "Type": "AWS::ECS::Cluster",
      "Properties": {
        "ClusterName": {"Ref": "ECSClusterName"},
        "ClusterSettings": [
          {
            "Name": "containerInsights",
            "Value": {"Ref": "EnableContainerInsights"}
          }
        ],
        "Tags": [
          {
            "Key": "Name",
            "Value": {"Ref": "ECSClusterName"}
          }
        ]
      }
    },
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
              },
```

```
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
      ]
    }
  },
  "EBSTaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
      "Family": "ebs-task-attach-task-def",
      "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
      "NetworkMode": "awsvpc",
      "RequiresCompatibilities": [
        "EC2",
        "FARGATE"
      ],
      "Cpu": {"Ref": "TaskCpu"},
      "Memory": {"Ref": "TaskMemory"},
      "ContainerDefinitions": [
        {
          "Name": "nginx",
          "Image": {"Ref": "ContainerImage"},
          "Essential": true,
          "PortMappings": [
            {
              "Name": "nginx-80-tcp",
              "ContainerPort": 80,
              "HostPort": 80,
              "Protocol": "tcp",
              "AppProtocol": "http"
            }
          ],
          "MountPoints": [
            {
              "SourceVolume": "ebs-vol",
              "ContainerPath": "/foo-container-path",
              "ReadOnly": false
            }
          ]
        }
      ],
```

```
        "Volumes": [
          {
            "Name": "ebs-vol",
            "ConfiguredAtLaunch": true
          }
        ]
      }
    },
    "ECSService": {
      "Type": "AWS::ECS::Service",
      "Properties": {
        "Cluster": {"Ref": "ECSCluster"},
        "TaskDefinition": {"Ref": "EBSTaskDefinition"},
        "LaunchType": "FARGATE",
        "ServiceName": "ebs",
        "SchedulingStrategy": "REPLICA",
        "DesiredCount": 1,
        "NetworkConfiguration": {
          "AwsvpcConfiguration": {
            "AssignPublicIp": "ENABLED",
            "SecurityGroups": {"Ref": "SecurityGroupIDs"},
            "Subnets": {"Ref": "SubnetIDs"}
          }
        },
        "PlatformVersion": "LATEST",
        "DeploymentConfiguration": {
          "MaximumPercent": 200,
          "MinimumHealthyPercent": 100,
          "DeploymentCircuitBreaker": {
            "Enable": true,
            "Rollback": true
          }
        },
        "DeploymentController": {
          "Type": "ECS"
        },
        "Tags": [],
        "EnableECSManagedTags": true,
        "VolumeConfigurations": [
          {
            "Name": "ebs-vol",
            "ManagedEBSVolume": {
              "RoleArn": {"Ref": "InfrastructureRoleArn"},
              "VolumeType": {"Ref": "VolumeType"},
```

```
              "Iops": {"Ref": "VolumeIops"},
              "Throughput": {"Ref": "VolumeThroughput"},
              "SizeInGiB": {"Ref": "VolumeSize"},
              "FilesystemType": {"Ref": "FilesystemType"},
              "TagSpecifications": [
                {
                  "ResourceType": "volume",
                  "PropagateTags": "TASK_DEFINITION"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "ClusterName": {
      "Description": "The cluster used to create the service.",
      "Value": {"Ref": "ECSCluster"}
    },
    "ClusterArn": {
      "Description": "The ARN of the ECS cluster",
      "Value": {"Fn::GetAtt": ["ECSCluster", "Arn"]}
    },
    "ECSService": {
      "Description": "The created service.",
      "Value": {"Ref": "ECSService"}
    },
    "TaskDefinitionArn": {
      "Description": "The ARN of the task definition",
      "Value": {"Ref": "EBSTaskDefinition"}
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: The template used to create an ECS Service that includes a volume
 configuration. The configuration is used to create Amazon EBS volumes for
 attachment to the tasks. One volume is attached per task.
Parameters:
```

```
  ECSClusterName:
    Type: String
    Default: volume-config-cluster
    Description: Name of the ECS cluster

  SecurityGroupIDs:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service

  SubnetIDs:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service

  InfrastructureRoleArn:
    Type: String
    Description: ARN of the IAM role that ECS will use to manage EBS volumes

  ContainerImage:
    Type: String
    Default: public.ecr.aws/nginx/nginx:latest
    Description: Container image to use for the task

  TaskCpu:
    Type: String
    Default: "2048"
    Description: CPU units for the task

  TaskMemory:
    Type: String
    Default: "4096"
    Description: Memory (in MiB) for the task

  VolumeSize:
    Type: String
    Default: "10"
    Description: Size of the EBS volume in GiB

  VolumeType:
    Type: String
    Default: gp3
    AllowedValues: [gp2, gp3, io1, io2, st1, sc1, standard]
    Description: EBS volume type

  VolumeIops:
```

```
    Type: String
    Default: "3000"
    Description: IOPS for the EBS volume (required for io1, io2, and gp3)

  VolumeThroughput:
    Type: String
    Default: "125"
    Description: Throughput for the EBS volume (only for gp3)

  FilesystemType:
    Type: String
    Default: xfs
    AllowedValues: [xfs, ext4]
    Description: Filesystem type for the EBS volume

  EnableContainerInsights:
    Type: String
    Default: 'enhanced'
    AllowedValues: ['enabled', 'disabled', 'enhanced']
    Description: Enable or disable CloudWatch Container Insights for the cluster

Resources:
  # ECS Cluster
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ECSClusterName
      ClusterSettings:
        - Name: containerInsights
          Value: !Ref EnableContainerInsights
      Tags:
        - Key: Name
          Value: !Ref ECSClusterName

  # IAM Role for Task Execution
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
```

```
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

  # Task Definition
  EBSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: ebs-task-attach-task-def
      ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - EC2
        - FARGATE
      Cpu: !Ref TaskCpu
      Memory: !Ref TaskMemory
      ContainerDefinitions:
        - Name: nginx
          Image: !Ref ContainerImage
          Essential: true
          PortMappings:
            - Name: nginx-80-tcp
              ContainerPort: 80
              HostPort: 80
              Protocol: tcp
              AppProtocol: http
          MountPoints:
            - SourceVolume: ebs-vol
              ContainerPath: /foo-container-path
              ReadOnly: false
      Volumes:
        - Name: ebs-vol
          ConfiguredAtLaunch: true

  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster: !Ref ECSCluster
      TaskDefinition: !Ref EBSTaskDefinition
      LaunchType: FARGATE
      ServiceName: ebs
      SchedulingStrategy: REPLICA
      DesiredCount: 1
      NetworkConfiguration:
```

```
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups: !Ref SecurityGroupIDs
          Subnets: !Ref SubnetIDs
      PlatformVersion: LATEST
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 100
        DeploymentCircuitBreaker:
          Enable: true
          Rollback: true
      DeploymentController:
        Type: ECS
      Tags: []
      EnableECSManagedTags: true
      VolumeConfigurations:
        - Name: ebs-vol
          ManagedEBSVolume:
            RoleArn: !Ref InfrastructureRoleArn
            VolumeType: !Ref VolumeType
            Iops: !Ref VolumeIops
            Throughput: !Ref VolumeThroughput
            SizeInGiB: !Ref VolumeSize
            FilesystemType: !Ref FilesystemType
            TagSpecifications:
              - ResourceType: volume
                PropagateTags: TASK_DEFINITION

Outputs:
  ClusterName:
    Description: The cluster used to create the service.
    Value: !Ref ECSCluster
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  ECSService:
    Description: The created service.
    Value: !Ref ECSService
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref EBSTaskDefinition
```

Menyebarkan layanan dengan penyedia kapasitas

Template berikut mendefinisikan layanan yang menggunakan penyedia kapasitas untuk meminta kapasitas AL2 023 untuk dijalankan. Kontainer akan diluncurkan ke instans AL2 023 saat mereka online.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "An example service that deploys in AWS VPC networking mode on
 EC2 capacity. Service uses a capacity provider to request EC2 instances to run
 on. Service runs with networking in private subnets, but still accessible to the
 internet via a load balancer hosted in public subnets.",
  "Parameters": {
      "VpcId": {
          "Type": "String",
          "Description": "The VPC that the service is running inside of"
      },
      "PublicSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of public subnet ID's to put the load balancer in"
      },
      "PrivateSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of private subnet ID's that the AWS VPC tasks are in"
      },
      "ClusterName": {
          "Type": "String",
          "Description": "The name of the ECS cluster into which to launch
 capacity."
      },
      "ECSTaskExecutionRole": {
          "Type": "String",
          "Description": "The role used to start up an ECS task"
      },
      "CapacityProvider": {
          "Type": "String",
          "Description": "The cluster capacity provider that the service should use
 to request capacity when it wants to start up a task"
      },
      "ServiceName": {
          "Type": "String",
```

```
            "Default": "web",
            "Description": "A name for the service"
        },
        "ImageUrl": {
            "Type": "String",
            "Default": "public.ecr.aws/docker/library/nginx:latest",
            "Description": "The url of a docker image that contains the application
    process that will handle the traffic for this service"
        },
        "ContainerCpu": {
            "Type": "Number",
            "Default": 256,
            "Description": "How much CPU to give the container. 1024 is 1 CPU"
        },
        "ContainerMemory": {
            "Type": "Number",
            "Default": 512,
            "Description": "How much memory in megabytes to give the container"
        },
        "ContainerPort": {
            "Type": "Number",
            "Default": 80,
            "Description": "What port that the application expects traffic on"
        },
        "DesiredCount": {
            "Type": "Number",
            "Default": 2,
            "Description": "How many copies of the service task to run"
        }
    },
    "Resources": {
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Ref": "ServiceName"
                },
                "Cpu": {
                    "Ref": "ContainerCpu"
                },
                "Memory": {
                    "Ref": "ContainerMemory"
                },
                "NetworkMode": "awsvpc",
```

```
                "RequiresCompatibilities": [
                    "EC2"
                ],
                "ExecutionRoleArn": {
                    "Ref": "ECSTaskExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Ref": "ServiceName"
                        },
                        "Cpu": {
                            "Ref": "ContainerCpu"
                        },
                        "Memory": {
                            "Ref": "ContainerMemory"
                        },
                        "Image": {
                            "Ref": "ImageUrl"
                        },
                        "PortMappings": [
                            {
                                "ContainerPort": {
                                    "Ref": "ContainerPort"
                                },
                                "HostPort": {
                                    "Ref": "ContainerPort"
                                }
                            }
                        ],
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "mode": "non-blocking",
                                "max-buffer-size": "25m",
                                "awslogs-group": {
                                    "Ref": "LogGroup"
                                },
                                "awslogs-region": {
                                    "Ref": "AWS::Region"
                                },
                                "awslogs-stream-prefix": {
                                    "Ref": "ServiceName"
                                }
```

```
                        }
                    }
                }
            ]
        }
    },
    "Service": {
        "Type": "AWS::ECS::Service",
        "DependsOn": "PublicLoadBalancerListener",
        "Properties": {
            "ServiceName": {
                "Ref": "ServiceName"
            },
            "Cluster": {
                "Ref": "ClusterName"
            },
            "PlacementStrategies": [
                {
                    "Field": "attribute:ecs.availability-zone",
                    "Type": "spread"
                },
                {
                    "Field": "cpu",
                    "Type": "binpack"
                }
            ],
            "CapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ],
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "SecurityGroups": [
                        {
                            "Ref": "ServiceSecurityGroup"
                        }
                    ],
                    "Subnets": {
                        "Ref": "PrivateSubnetIds"
```

```
                    }
                }
            },
            "DeploymentConfiguration": {
                "MaximumPercent": 200,
                "MinimumHealthyPercent": 75
            },
            "DesiredCount": {
                "Ref": "DesiredCount"
            },
            "TaskDefinition": {
                "Ref": "TaskDefinition"
            },
            "LoadBalancers": [
                {
                    "ContainerName": {
                        "Ref": "ServiceName"
                    },
                    "ContainerPort": {
                        "Ref": "ContainerPort"
                    },
                    "TargetGroupArn": {
                        "Ref": "ServiceTargetGroup"
                    }
                }
            ]
        }
    },
    "ServiceSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Security group for service",
            "VpcId": {
                "Ref": "VpcId"
            }
        }
    },
    "ServiceTargetGroup": {
        "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
        "Properties": {
            "HealthCheckIntervalSeconds": 6,
            "HealthCheckPath": "/",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckTimeoutSeconds": 5,
```

```
                "HealthyThresholdCount": 2,
                "TargetType": "ip",
                "Port": {
                    "Ref": "ContainerPort"
                },
                "Protocol": "HTTP",
                "UnhealthyThresholdCount": 10,
                "VpcId": {
                    "Ref": "VpcId"
                },
                "TargetGroupAttributes": [
                    {
                        "Key": "deregistration_delay.timeout_seconds",
                        "Value": 0
                    }
                ]
            }
        },
        "PublicLoadBalancerSG": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Access to the public facing load balancer",
                "VpcId": {
                    "Ref": "VpcId"
                },
                "SecurityGroupIngress": [
                    {
                        "CidrIp": "0.0.0.0/0",
                        "IpProtocol": -1
                    }
                ]
            }
        },
        "PublicLoadBalancer": {
            "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
            "Properties": {
                "Scheme": "internet-facing",
                "LoadBalancerAttributes": [
                    {
                        "Key": "idle_timeout.timeout_seconds",
                        "Value": "30"
                    }
                ],
                "Subnets": {
```

```
                "Ref": "PublicSubnetIds"
            },
            "SecurityGroups": [
                {
                    "Ref": "PublicLoadBalancerSG"
                }
            ]
        }
    },
    "PublicLoadBalancerListener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "Properties": {
            "DefaultActions": [
                {
                    "Type": "forward",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn": {
                                    "Ref": "ServiceTargetGroup"
                                },
                                "Weight": 100
                            }
                        ]
                    }
                }
            ],
            "LoadBalancerArn": {
                "Ref": "PublicLoadBalancer"
            },
            "Port": 80,
            "Protocol": "HTTP"
        }
    },
    "ServiceIngressfromLoadBalancer": {
        "Type": "AWS::EC2::SecurityGroupIngress",
        "Properties": {
            "Description": "Ingress from the public ALB",
            "GroupId": {
                "Ref": "ServiceSecurityGroup"
            },
            "IpProtocol": -1,
            "SourceSecurityGroupId": {
                "Ref": "PublicLoadBalancerSG"
```

```
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service
  runs with networking in private subnets, but still accessible to the internet
  via a load balancer hosted in public subnets.
Parameters:
  VpcId:
    Type: String
    Description: The VPC that the service is running inside of
  PublicSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of public subnet ID's to put the load balancer in
  PrivateSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of private subnet ID's that the AWS VPC tasks are in
  ClusterName:
    Type: String
    Description: The name of the ECS cluster into which to launch capacity.
  ECSTaskExecutionRole:
    Type: String
    Description: The role used to start up an ECS task
  CapacityProvider:
    Type: String
    Description: >-
      The cluster capacity provider that the service should use to request
      capacity when it wants to start up a task
  ServiceName:
    Type: String
    Default: web
    Description: A name for the service
  ImageUrl:
```

```
    Type: String
    Default: 'public.ecr.aws/docker/library/nginx:latest'
    Description: >-
      The url of a docker image that contains the application process that will
      handle the traffic for this service
  ContainerCpu:
    Type: Number
    Default: 256
    Description: How much CPU to give the container. 1024 is 1 CPU
  ContainerMemory:
    Type: Number
    Default: 512
    Description: How much memory in megabytes to give the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: What port that the application expects traffic on
  DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
Resources:
  TaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      Family: !Ref ServiceName
      Cpu: !Ref ContainerCpu
      Memory: !Ref ContainerMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - EC2
      ExecutionRoleArn: !Ref ECSTaskExecutionRole
      ContainerDefinitions:
        - Name: !Ref ServiceName
          Cpu: !Ref ContainerCpu
          Memory: !Ref ContainerMemory
          Image: !Ref ImageUrl
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              HostPort: !Ref ContainerPort
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
```

```
            max-buffer-size: 25m
            awslogs-group: !Ref LogGroup
            awslogs-region: !Ref AWS::Region
            awslogs-stream-prefix: !Ref ServiceName
  Service:
    Type: AWS::ECS::Service
    DependsOn: PublicLoadBalancerListener
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ClusterName
      PlacementStrategies:
        - Field: 'attribute:ecs.availability-zone'
          Type: spread
        - Field: cpu
          Type: binpack
      CapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
      NetworkConfiguration:
        AwsvpcConfiguration:
          SecurityGroups:
            - !Ref ServiceSecurityGroup
          Subnets: !Ref PrivateSubnetIds
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: !Ref DesiredCount
      TaskDefinition: !Ref TaskDefinition
      LoadBalancers:
        - ContainerName: !Ref ServiceName
          ContainerPort: !Ref ContainerPort
          TargetGroupArn: !Ref ServiceTargetGroup
  ServiceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Security group for service
      VpcId: !Ref VpcId
  ServiceTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
```

```
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Port: !Ref ContainerPort
      Protocol: HTTP
      UnhealthyThresholdCount: 10
      VpcId: !Ref VpcId
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: 0
  PublicLoadBalancerSG:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: -1
  PublicLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '30'
      Subnets: !Ref PublicSubnetIds
      SecurityGroups:
        - !Ref PublicLoadBalancerSG
  PublicLoadBalancerListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ServiceTargetGroup
                Weight: 100
      LoadBalancerArn: !Ref PublicLoadBalancer
      Port: 80
      Protocol: HTTP
  ServiceIngressfromLoadBalancer:
    Type: 'AWS::EC2::SecurityGroupIngress'
    Properties:
      Description: Ingress from the public ALB
```

```
      GroupId: !Ref ServiceSecurityGroup
      IpProtocol: -1
      SourceSecurityGroupId: !Ref PublicLoadBalancerSG
  LogGroup:
    Type: 'AWS::Logs::LogGroup'
```

Peran IAM untuk Amazon ECS

Anda dapat menggunakan AWS CloudFormation template untuk membuat peran IAM untuk digunakan dengan Amazon ECS. Untuk informasi selengkapnya tentang peran IAM untuk Amazon ECS, lihat. Peran IAM untuk Amazon ECS

Peran eksekusi tugas Amazon ECS

Peran eksekusi tugas memberikan izin kepada kontainer Amazon ECS dan agen Fargate untuk melakukan panggilan AWS API atas nama Anda. Peran diperlukan tergantung pada persyaratan tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Template berikut dapat digunakan untuk membuat peran eksekusi tugas sederhana yang menggunakan kebijakan `AmazonECSTaskExecutionRolePolicy` terkelola.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation template for ECS Task Execution Role",
  "Resources": {
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": ["ecs-tasks.amazonaws.com"]
              },
              "Action": ["sts:AssumeRole"],
              "Condition": {
                "ArnLike": {
                  "aws:SourceArn": {
```

```
                  "Fn::Sub": "arn:aws:ecs:${AWS::Region}:${AWS::AccountId}:*"
                }
              },
              "StringEquals": {
                "aws:SourceAccount": {
                  "Ref": "AWS::AccountId"
                }
              }
            }
          }
        ]
      },
      "Path": "/",
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
      ]
    }
  }
},
"Outputs": {
  "ECSTaskExecutionRoleARN": {
    "Description": "ARN of the ECS Task Execution Role",
    "Value": {
      "Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECSTaskExecutionRoleARN"
      }
    }
  },
  "ECSTaskExecutionRoleName": {
    "Description": "Name of the ECS Task Execution Role",
    "Value": {
      "Ref": "ECSTaskExecutionRole"
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECSTaskExecutionRoleName"
      }
    }
  }
}
```

```
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'CloudFormation template for ECS Task Execution Role'
Resources:
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: [ecs-tasks.amazonaws.com]
            Action: ['sts:AssumeRole']
            Condition:
              ArnLike:
                aws:SourceArn: !Sub arn:aws:ecs:${AWS::Region}:${AWS::AccountId}:*
              StringEquals:
                aws:SourceAccount: !Ref AWS::AccountId
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ECSTaskExecutionRoleARN:
    Description: ARN of the ECS Task Execution Role
    Value: !GetAtt ECSTaskExecutionRole.Arn
    Export:
      Name: !Sub "${AWS::StackName}-ECSTaskExecutionRoleARN"
  ECSTaskExecutionRoleName:
    Description: Name of the ECS Task Execution Role
    Value: !Ref ECSTaskExecutionRole
    Export:
      Name: !Sub "${AWS::StackName}-ECSTaskExecutionRoleName"
```

Membuat sumber daya Amazon ECS menggunakan konsol AWS CloudFormation

Salah satu cara untuk menggunakan Amazon ECS AWS CloudFormation adalah melalui. AWS Management Console Di sini Anda dapat membuat AWS CloudFormation tumpukan untuk komponen

Amazon ECS seperti definisi tugas, kluster, dan layanan dan menerapkannya langsung dari konsol. Tutorial berikut menunjukkan bagaimana Anda dapat menggunakan AWS CloudFormation konsol untuk membuat layanan Amazon ECS, definisi tugas, dan cluster.

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah selesai.

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.

Langkah 1: Buat template tumpukan

Gunakan langkah-langkah berikut untuk membuat template AWS CloudFormation tumpukan untuk layanan Amazon ECS dan sumber daya terkait lainnya.

1. Menggunakan editor teks pilihan Anda, buat file bernama`ecs-tutorial-template.yaml`.
2. Dalam `ecs-tutorial-template.yaml` file, tempel template berikut dan simpan perubahannya.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A template that deploys an application that is built on an Apache
 web server Docker image by creating an Amazon ECS cluster, task definition, and
 service. The template also creates networking and logging resources, and an Amazon
 ECS task execution role.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  TaskFamily:
    Type: String
    Default: task-definition-cfn
    Description: Family name for the Task Definition
  ServiceName:
    Type: String
    Default: cfn-service
    Description: Name of the ECS Service
  ContainerImage:
```

```
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
  TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
  DesiredCount:
    Type: Number
    Default: 1
    Description: Desired number of tasks to run
  LogGroupName:
    Type: String
    Default: /ecs/fargate-task-definition
    Description: CloudWatch Log Group name
  VpcCidr:
    Type: String
    Default: 10.0.0.0/16
    Description: CIDR block for the VPC
  PublicSubnet1Cidr:
    Type: String
    Default: 10.0.0.0/24
    Description: CIDR block for public subnet 1
  PublicSubnet2Cidr:
    Type: String
    Default: 10.0.1.0/24
    Description: CIDR block for public subnet 2
Resources:
  # VPC and Networking Resources
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-VPC
```

```
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-IGW
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Ref PublicSubnet1Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet1
  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs '']
      CidrBlock: !Ref PublicSubnet2Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet2
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicRouteTable
  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
```

```
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
  # Security Group
  ECSSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ECS tasks
      VpcId: !Ref VPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
  # IAM Roles
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
  # CloudWatch Logs
  TaskLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
```

```
    Properties:
      LogGroupName: !Ref LogGroupName
      RetentionInDays: 30
  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ClusterName
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"s
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: !Ref ContainerImage
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-create-group: 'true'
              awslogs-group: !Ref LogGroupName
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: !Ref TaskCpu
      ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
      Family: !Ref TaskFamily
      Memory: !Ref TaskMemory
```

```
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: AWS::ECS::Service
    DependsOn:
      - PublicSubnet1RouteTableAssociation
      - PublicSubnet2RouteTableAssociation
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ECSCluster
      DesiredCount: !Ref DesiredCount
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - !Ref ECSSecurityGroup
          Subnets:
            - !Ref PublicSubnet1
            - !Ref PublicSubnet2
      TaskDefinition: !Ref ECSTaskDefinition
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref ECSTaskDefinition
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ECSService
  VpcId:
    Description: The ID of the VPC
    Value: !Ref VPC
  PublicSubnet1:
    Description: The ID of public subnet 1
    Value: !Ref PublicSubnet1
  PublicSubnet2:
    Description: The ID of public subnet 2
    Value: !Ref PublicSubnet2
  SecurityGroup:
```

```
    Description: The ID of the security group
    Value: !Ref ECSSecurityGroup
  ExecutionRoleArn:
    Description: The ARN of the task execution role
    Value: !GetAtt ECSTaskExecutionRole.Arn
```

Langkah 2: Buat tumpukan untuk sumber daya Amazon ECS

Setelah membuat file untuk template, Anda dapat mengikuti langkah-langkah ini untuk membuat tumpukan dengan template dengan menggunakan AWS CloudFormation konsol.

1. Masuk ke AWS Management Console dan buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pada halaman Stacks, di sudut kanan atas, pilih Buat tumpukan, lalu pilih dengan sumber daya baru (standar).
3. Pilih Pilih template yang sudah ada.
4. Pilih Unggah file templat lalu pilih Pilih file untuk memilih `ecs-tutorial-template` file.

   Setelah file diunggah ke bucket Amazon S3, Anda dapat memilih View in Infrastructure Composer untuk memvisualisasikan template di Infrastructure Composer. Untuk informasi selengkapnya tentang AWS CloudFormation template dan Infrastructure Composer, lihat Membuat template secara visual dengan Infrastructure Composer di AWS CloudFormation Panduan Pengguna.
5. Pilih Berikutnya.
6. Pada halaman Tentukan detail tumpukan, di bawah nama Stack, berikan nama berikut untuk tumpukan:`ecs-tutorial-stack`. Biarkan semua nilai untuk parameter di bawah Parameter sebagai default, lalu pilih Berikutnya.
7. Pada halaman Configure stack options, di bawah Capabilities, pilih kotak centang untuk mengakui AWS CloudFormation pembuatan resource IAM. Pengakuan ini diperlukan untuk membuat peran eksekusi tugas Amazon ECS seperti yang didefinisikan dalam template. Biarkan pengaturan lainnya sebagai default dan pilih Berikutnya.
8. Tinjau detail tumpukan pada halaman Tinjauan dan buat lalu pilih Kirim untuk memulai pembuatan tumpukan.

Langkah 3: Verifikasi

Gunakan langkah-langkah berikut untuk memverifikasi pembuatan sumber daya Amazon ECS menggunakan templat yang disediakan.

1. Masuk ke AWS Management Console dan buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pada halaman Stacks, pilih ecs-tutorial-stack.
3. Pilih tab Acara. Jika status acara mengatakan`CREATE_IN_PROGRESS`, tunggu sampai pembuatan selesai dan status berubah menjadi. `CREATE_COMPLETE`
4. Setelah status acara beralih ke`CREATE_COMPLETE`, pilih tab Sumber Daya. Anda akan melihat sumber daya dengan ID Logis`ECSCluster`,`ECSTaskDefinition`, dan `ECSService` masing-masing.
5. Untuk memverifikasi pembuatan klaster Amazon ECS, pilih ID Fisik yang terkait dengannya`ECSCluster`. Anda akan diarahkan ke konsol Amazon ECS di mana Anda dapat melihat cluster yang dibuat dipanggil. `CFNCluster`
6. Untuk memverifikasi pembuatan layanan Amazon ECS, pilih ID Fisik yang terkait dengannya`ECSService`. Anda akan diarahkan ke konsol Amazon ECS di mana Anda dapat melihat layanan yang disebut `cfn-service` yang dibuat di cluster. `cfnCluster`
7. Untuk memverifikasi pembuatan definisi tugas Amazon ECS, pilih ID Fisik yang terkait dengannya`ECSTaskDefinition`. Anda akan diarahkan ke konsol Amazon ECS di mana Anda dapat melihat revisi definisi tugas dengan nama. `task-definition-cfn`

Langkah 4: Bersihkan Sumber Daya

Untuk membersihkan sumber daya dan menghindari biaya lebih lanjut, ikuti langkah-langkah ini.

1. Masuk ke AWS Management Console dan buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pada halaman Stacks, pilih ecs-tutorial-stack.
3. Pilih Hapus.
4. Saat diminta konfirmasi, pilih Hapus lagi.
5. Pilih tab Acara. Status untuk `ecs-tutorial-stack` perubahan pada DELETE_IN_PROGRESS dan kemudian ke DELETE_COMPLETE setelah sumber daya dihapus atau dideregistrasi. Penghapusan memakan waktu beberapa menit.

6. Pilih tab Sumber Daya. Anda sekarang akan melihat daftar ID Logis dengan Status diperbarui ke DELETE_COMPLETE.

Membuat sumber daya Amazon ECS menggunakan AWS CLI perintah untuk AWS CloudFormation

Tutorial berikut menunjukkan bagaimana Anda dapat menggunakan AWS CLI untuk membuat sumber daya Amazon ECS dengan AWS CloudFormation template.

Prasyarat

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.

Langkah 1: Buat tumpukan

Untuk membuat tumpukan menggunakan AWS CLI, ikuti langkah-langkah ini.

1. Menggunakan editor teks pilihan Anda, buat file bernama`ecs-tutorial-template.yaml`.
2. Dalam `ecs-tutorial-template.yaml` file, tempel template berikut dan simpan perubahannya.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A template that deploys an application that is built on an Apache
 web server Docker image by creating an Amazon ECS cluster, task definition, and
 service. The template also creates networking and logging resources, and an Amazon
 ECS task execution role.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  TaskFamily:
    Type: String
    Default: task-definition-cfn
    Description: Family name for the Task Definition
  ServiceName:
```

```
    Type: String
    Default: cfn-service
    Description: Name of the ECS Service
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
  TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
  DesiredCount:
    Type: Number
    Default: 1
    Description: Desired number of tasks to run
  LogGroupName:
    Type: String
    Default: /ecs/fargate-task-definition
    Description: CloudWatch Log Group name
  VpcCidr:
    Type: String
    Default: 10.0.0.0/16
    Description: CIDR block for the VPC
  PublicSubnet1Cidr:
    Type: String
    Default: 10.0.0.0/24
    Description: CIDR block for public subnet 1
  PublicSubnet2Cidr:
    Type: String
    Default: 10.0.1.0/24
    Description: CIDR block for public subnet 2
Resources:
  # VPC and Networking Resources
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsSupport: true
```

```
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-VPC
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-IGW
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Ref PublicSubnet1Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet1
  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs '']
      CidrBlock: !Ref PublicSubnet2Cidr
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicSubnet2
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${AWS::StackName}-PublicRouteTable
  DefaultPublicRoute:
    Type: AWS::EC2::Route
```

```
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
  # Security Group
  ECSSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ECS tasks
      VpcId: !Ref VPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
  # IAM Roles
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
  # CloudWatch Logs
```

```
  TaskLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      LogGroupName: !Ref LogGroupName
      RetentionInDays: 30
  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ClusterName
  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"s
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: !Ref ContainerImage
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-create-group: 'true'
              awslogs-group: !Ref LogGroupName
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
```

```
      Cpu: !Ref TaskCpu
      ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
      Family: !Ref TaskFamily
      Memory: !Ref TaskMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: AWS::ECS::Service
    DependsOn:
      - PublicSubnet1RouteTableAssociation
      - PublicSubnet2RouteTableAssociation
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ECSCluster
      DesiredCount: !Ref DesiredCount
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - !Ref ECSSecurityGroup
          Subnets:
            - !Ref PublicSubnet1
            - !Ref PublicSubnet2
      TaskDefinition: !Ref ECSTaskDefinition
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref ECSTaskDefinition
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ECSService
  VpcId:
    Description: The ID of the VPC
    Value: !Ref VPC
  PublicSubnet1:
    Description: The ID of public subnet 1
    Value: !Ref PublicSubnet1
```

```
  PublicSubnet2:
    Description: The ID of public subnet 2
    Value: !Ref PublicSubnet2
  SecurityGroup:
    Description: The ID of the security group
    Value: !Ref ECSSecurityGroup
  ExecutionRoleArn:
    Description: The ARN of the task execution role
    Value: !GetAtt ECSTaskExecutionRole.Arn
```

3. Setelah membuat file template, gunakan perintah berikut untuk membuat tumpukan. `--capabilities`Bendera diperlukan untuk membuat peran eksekusi tugas Amazon ECS seperti yang ditentukan dalam template. Anda juga dapat menentukan `--parameters` bendera untuk menyesuaikan parameter template.

```
aws cloudformation create-stack \
      --stack-name ecs-stack \
      --template-body file://ecs-tutorial-template.yaml \
      --capabilities CAPABILITY_IAM
```

Langkah 2: Verifikasi pembuatan sumber daya

Untuk memastikan bahwa sumber daya dibuat dengan benar, ikuti langkah-langkah ini. Anda juga dapat memeriksa konsol Amazon ECS.

1. Jalankan perintah berikut untuk mencantumkan semua definisi tugas dalam an Wilayah AWS.

```
aws ecs list-task-definitions
```

   Perintah mengembalikan daftar definisi tugas Amazon Resource Name (ARN) s. ARN definisi tugas yang Anda buat menggunakan template akan ditampilkan dalam format berikut.

```
{
    "taskDefinitionArns": [
     .....
        "arn:aws:ecs:aws-region:111122223333:task-definition/task-definition-
cfn:1",
     .....
    ]
}
```

2. Jalankan perintah berikut untuk mencantumkan semua cluster dalam file Wilayah AWS.

```
aws ecs list-clusters
```

Perintah mengembalikan daftar cluster ARNs. ARN dari cluster yang Anda buat menggunakan template akan ditampilkan dalam format berikut.

```
{
    "clusterArns": [
        .....
        "arn:aws:ecs:aws-region:111122223333:cluster/CFNCluster",
        .....
    ]
}
```

3. Jalankan perintah berikut untuk daftar semua layanan di cluster`CFNCluster`.

```
aws ecs list-services \
      --cluster CFNCluster
```

Perintah mengembalikan daftar layanan ARNs. ARN layanan yang Anda buat menggunakan template akan ditampilkan dalam format berikut.

```
{
    "serviceArns": [
        "arn:aws:ecs:aws-region:111122223333:service/CFNCluster/cfn-service"
    ]
}
```

Langkah 3: Bersihkan

Untuk membersihkan sumber daya yang Anda buat, jalankan perintah berikut.

```
aws cloudformation delete-stack \
      --stack-name ecs-stack
```

`delete-stack`Perintah memulai penghapusan AWS CloudFormation tumpukan yang dibuat dalam tutorial ini, menghapus semua sumber daya di tumpukan. Untuk memverifikasi penghapusan, Anda dapat mengulangi prosedur di. Langkah 2: Verifikasi pembuatan sumber daya Daftar ARNs dalam

output tidak akan lagi menyertakan definisi tugas yang disebut `task-definition-cfn` atau cluster yang disebut`CFNCLuster`. `list-services`Panggilan akan gagal.

Praktik terbaik Amazon ECS

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional yang paling penting untuk jaringan Amazon ECS.

Ikhtisar praktik terbaik	Pelajari selengkapnya
Connect aplikasi ke internet	Hubungkan aplikasi Amazon ECS ke internet
Terima koneksi masuk ke Amazon ECS dari internet.	Praktik terbaik untuk menerima koneksi masuk ke Amazon ECS dari internet
Hubungkan Amazon ECS ke AWS layanan lain dalam VPC	Praktik terbaik untuk menghubungkan Amazon ECS ke AWS layanan dari dalam VPC Anda
Layanan jaringan di seluruh Akun AWS dan VPCs	Praktik terbaik untuk jaringan layanan Amazon ECS di seluruh Akun AWS dan VPCs
Memecahkan masalah jaringan	AWS layanan untuk pemecahan masalah jaringan Amazon ECS

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional terpenting untuk Fargate di Amazon ECS.

Ikhtisar praktik terbaik	Pelajari selengkapnya
Keamanan Fargate	Praktik terbaik keamanan Fargate di Amazon ECS
Pertimbangan keamanan Fargate	Pertimbangan keamanan Fargate untuk Amazon ECS

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Wadah Linux pada perilaku tarik gambar kontainer Fargate	Wadah Linux pada perilaku tarik gambar kontainer Fargate untuk Amazon ECS	
Wadah Windows pada perilaku tarik gambar kontainer Fargate	Wadah Windows pada perilaku tarik gambar kontainer Fargate untuk Amazon ECS	
Pensiun tugas Fargate	Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS	

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional yang paling penting untuk definisi tugas.

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Image kontainer	Praktik terbaik untuk gambar kontainer Amazon ECS	
Ukuran tugas	Praktik terbaik untuk ukuran tugas Amazon ECS	
Praktik terbaik volume	Opsi penyimpanan untuk tugas Amazon ECS	

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional yang paling penting untuk cluster dan kapasitas.

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Keamanan Fargate	Praktik terbaik keamanan Fargate di Amazon ECS	

Ikhtisar praktik terbaik	Pelajari selengkapnya	
EC2 pertimbangan keamanan contoh kontainer	Pertimbangan keamanan instans EC2 penampung Amazon untuk Amazon ECS	
Auto Scaling klaster	Optimalkan penskalaan otomatis cluster Amazon ECS	
Beroperasi pada skala	Mengoperasikan Amazon ECS dalam skala besar	
Penskalaan otomatis dan manajemen kapasitas	Praktik terbaik penskalaan dan manajemen kapasitas Amazon ECS Auto	

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional yang paling penting untuk tugas dan layanan.

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Optimalkan waktu peluncuran tugas	Optimalkan waktu peluncuran tugas Amazon ECS	
Parameter layanan	Praktik terbaik untuk parameter layanan Amazon ECS	
Optimalkan parameter pemeriksaan kesehatan penyeimbang beban	Optimalkan parameter pemeriksaan kesehatan penyeimbang beban untuk Amazon ECS	
Optimalkan parameter pengurasan koneksi penyeimbang beban	Optimalkan parameter pengurasan koneksi penyeimbang beban untuk Amazon ECS	

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Optimalkan penskalaan otomatis layanan	Mengoptimalkan penskalaan otomatis layanan Amazon ECS	

Anda dapat menggunakan salah satu halaman berikut untuk mempelajari praktik terbaik operasional yang paling penting untuk keamanan.

Ikhtisar praktik terbaik	Pelajari selengkapnya	
Keamanan jaringan	Praktik terbaik keamanan jaringan untuk Amazon ECS	
Keamanan tugas dan kontainer	Praktik terbaik tugas Amazon ECS dan keamanan kontainer	

AWS Fargateuntuk Amazon ECS

AWS Fargateadalah teknologi yang dapat Anda gunakan dengan Amazon ECS untuk menjalankan container tanpa harus mengelola server atau cluster instance Amazon EC2 . Dengan AWS Fargate, Anda tidak perlu lagi menyediakan, mengonfigurasi, atau menskalakan klaster mesin virtual untuk menjalankan kontainer. Hal ini menghilangkan kebutuhan untuk memilih tipe server, memutuskan waktu untuk menskalakan klaster Anda, atau mengoptimalkan klaster kemasan.

Ketika Anda menjalankan tugas dan layanan Anda dengan tipe peluncuran Fargate, Anda mengemas aplikasi Anda dalam wadah, menentukan persyaratan CPU dan memori, menentukan kebijakan jaringan dan IAM, dan meluncurkan aplikasi. Setiap tugas Fargate memiliki batas isolasi sendiri dan tidak berbagi kernel yang mendasarinya, sumber daya CPU, sumber daya memori, atau elastic network interface dengan tugas lain. Anda mengonfigurasi definisi tugas untuk Fargate dengan menyetel parameter definisi `requiresCompatibilities` tugas ke. `FARGATE` Untuk informasi selengkapnya, lihat Jenis peluncuran.

Fargate menawarkan versi platform untuk Amazon Linux 2 (versi platform 1.3.0), sistem operasi Bottlerocket (platform versi 1.4.0), dan Microsoft Windows 2019 Server edisi Lengkap dan Inti. Kecuali ditentukan lain, informasi di halaman ini berlaku untuk semua platform Fargate.

Topik ini menjelaskan berbagai komponen tugas dan layanan Fargate, dan memanggil pertimbangan khusus untuk menggunakan Fargate dengan Amazon ECS.

Untuk informasi tentang Wilayah yang mendukung kontainer Linux di Fargate, lihat. the section called “Kontainer Linux di AWS Fargate”

Untuk informasi tentang Wilayah yang mendukung kontainer Windows di Fargate, lihat. the section called “Wadah Windows di AWS Fargate”

Panduan

Untuk informasi tentang cara memulai menggunakan konsol, lihat:

- Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate
- Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate

Untuk informasi tentang cara mulai menggunakan AWS CLI, lihat:

- Membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate dengan AWS CLI
- Membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate dengan AWS CLI

Penyedia kapasitas

Penyedia kapasitas berikut tersedia:

- Fargate
- Fargate Spot - Jalankan tugas Amazon ECS toleran interupsi dengan tarif diskon dibandingkan dengan harga. AWS Fargate Spot Fargate menjalankan tugas pada kapasitas komputasi cadangan. Ketika AWS membutuhkan kapasitas kembali, tugas Anda akan terganggu dengan peringatan dua menit. Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk Fargate.

Ketentuan tugas

Tugas yang menggunakan tipe peluncuran Fargate tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda untuk tugas Fargate. Untuk informasi selengkapnya, lihat CPU dan memori tugas.

Versi platform

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Ini adalah kombinasi dari kernel dan versi runtime container. Anda memilih versi platform ketika Anda menjalankan tugas atau ketika Anda membuat layanan untuk mempertahankan sejumlah tugas yang identik.

Revisi baru versi platform dirilis saat lingkungan runtime berkembang, misalnya, jika ada pembaruan kernel atau sistem operasi, fitur baru, perbaikan bug, atau pembaruan keamanan. Versi platform Fargate diperbarui dengan membuat revisi versi platform baru. Setiap tugas berjalan pada satu revisi versi platform selama siklus hidupnya. Jika Anda ingin menggunakan revisi versi platform terbaru, maka Anda harus memulai tugas baru. Tugas baru yang berjalan di Fargate selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.

Jika ditemukan masalah keamanan yang memengaruhi versi platform yang ada, AWS buat revisi baru yang ditambal dari versi platform dan menghentikan tugas yang berjalan pada revisi yang

rentan. Dalam beberapa kasus, Anda mungkin diberi tahu bahwa tugas Anda di Fargate telah dijadwalkan untuk pensiun. Untuk informasi selengkapnya, lihat Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS .

Untuk mengetahui informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.

Penyeimbangan beban layanan

Layanan Amazon ECS Anda secara opsional AWS Fargate dapat dikonfigurasi untuk menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas dalam layanan Anda.

Layanan Amazon ECS AWS Fargate mendukung Application Load Balancer, Network Load Balancer, dan tipe load balancer. Aplikasi Load Balancer digunakan untuk merutekan HTTP/HTTPS (atau lapisan 7) lalu lintas. Network Load Balancer digunakan untuk merutekan lalu lintas TCP atau UDP (atau layer 4). Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Ketika Anda membuat grup target untuk layanan ini, Anda harus memilih `ip` sebagai tipe target, bukan `instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Menggunakan Network Load Balancer untuk merutekan lalu lintas UDP ke Amazon ECS pada AWS Fargate tugas hanya didukung saat menggunakan platform versi 1.4 atau yang lebih baru.

Metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke akun Anda penggunaan sumber daya. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

AWS Fargate metrik penggunaan sesuai dengan kuota AWS layanan. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota AWS Fargate layanan, titik akhir Amazon ECS dan kuota di.. Referensi Umum Amazon Web

Untuk informasi selengkapnya tentang metrik AWS Fargate penggunaan, lihat metrik AWS Fargate penggunaan.

Pertimbangan keamanan Amazon ECS kapan harus menggunakan jenis peluncuran Fargate

Kami menyarankan agar pelanggan yang mencari isolasi yang kuat untuk tugas mereka menggunakan Fargate. Fargate menjalankan setiap tugas dalam lingkungan virtualisasi perangkat keras. Ini memastikan bahwa beban kerja kontainer ini tidak berbagi antarmuka jaringan, penyimpanan sementara Fargate, CPU, atau memori dengan tugas lain. Untuk informasi selengkapnya, lihat Gambaran Umum Keamanan AWS Fargate.

Praktik terbaik keamanan Fargate di Amazon ECS

Kami menyarankan Anda mempertimbangkan praktik terbaik berikut saat Anda menggunakannya AWS Fargate. Untuk panduan tambahan, lihat Ikhtisar keamanan AWS Fargate.

Gunakan AWS KMS untuk mengenkripsi penyimpanan sementara untuk Fargate

Anda harus memiliki penyimpanan sementara Anda dienkripsi oleh salah satu AWS KMS atau kunci yang dikelola pelanggan Anda sendiri. Untuk tugas yang di-host di Fargate menggunakan versi platform `1.4.0` atau yang lebih baru, setiap tugas menerima 20 GiB penyimpanan sementara. Untuk informasi selengkapnya, lihat kunci terkelola pelanggan (CMK). Anda dapat meningkatkan jumlah total penyimpanan sementara, hingga maksimum 200 GiB, dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda. Untuk tugas-tugas seperti itu yang diluncurkan pada 28 Mei 2020 atau lebih baru, penyimpanan sementara dienkripsi dengan algoritma enkripsi AES-256 menggunakan kunci enkripsi yang dikelola oleh Fargate.

Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.

Contoh: Meluncurkan tugas di platform Fargate versi 1.4.0 dengan enkripsi penyimpanan singkat

Perintah berikut akan meluncurkan tugas pada platform Fargate versi 1.4. Karena tugas ini diluncurkan sebagai bagian dari cluster, ia menggunakan 20 GiB penyimpanan sementara yang secara otomatis dienkripsi.

```
aws ecs run-task --cluster clustername \
  --task-definition taskdefinition:version \
  --count 1
```

```
  --launch-type "FARGATE" \
  --platform-version 1.4.0 \
  --network-configuration
"awsvpcConfiguration={subnets=[subnetid],securityGroups=[securitygroupid]}" \
  --region region
```

Kemampuan SYS_PTRACE untuk penelusuran syscall kernel dengan Fargate

Konfigurasi default kemampuan Linux yang ditambahkan atau dihapus dari container Anda disediakan oleh Docker.

Tugas yang diluncurkan di Fargate hanya mendukung penambahan kemampuan `SYS_PTRACE` kernel.

Video berikut menunjukkan cara menggunakan fitur ini melalui proyek Sysdig Falco.

ContainersFromTheCouch - Memecahkan Masalah Tugas Fargate Anda menggunakan kemampuan SYS_PTRACE

Kode yang dibahas dalam video sebelumnya dapat ditemukan di GitHub sini.

Gunakan Amazon GuardDuty dengan Fargate Runtime Monitoring

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Runtime Monitoring in GuardDuty melindungi beban kerja yang berjalan di Fargate dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensyal yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDuty Runtime Monitoring di Panduan GuardDuty Pengguna.

Pertimbangan keamanan Fargate untuk Amazon ECS

Setiap tugas memiliki kapasitas infrastruktur khusus karena Fargate menjalankan setiap beban kerja pada lingkungan virtual yang terisolasi. Beban kerja yang berjalan di Fargate tidak berbagi antarmuka jaringan, penyimpanan sementara, CPU, atau memori dengan tugas lain. Anda dapat menjalankan beberapa kontainer dalam tugas termasuk wadah aplikasi dan kontainer sespan, atau hanya sespan. Sidecar adalah wadah yang berjalan bersama wadah aplikasi dalam tugas Amazon ECS. Sementara wadah aplikasi menjalankan kode aplikasi inti, proses yang berjalan di sidecars dapat menambah aplikasi. Sidecars membantu Anda memisahkan fungsi aplikasi ke dalam wadah khusus, sehingga memudahkan Anda memperbarui bagian aplikasi Anda.

Kontainer yang merupakan bagian dari tugas berbagi sumber daya yang sama untuk jenis peluncuran Fargate karena kontainer ini akan selalu berjalan di host yang sama dan berbagi sumber daya komputasi. Wadah ini juga berbagi penyimpanan singkat yang disediakan oleh Fargate. Kontainer Linux dalam ruang nama jaringan berbagi tugas, termasuk alamat IP dan port jaringan. Di dalam tugas, kontainer yang termasuk dalam tugas dapat saling berkomunikasi melalui localhost.

Lingkungan runtime di Fargate mencegah Anda menggunakan fitur pengontrol tertentu yang didukung EC2 pada instance. Pertimbangkan hal berikut ketika Anda merancang beban kerja yang berjalan di Fargate:

- Tidak ada wadah atau akses istimewa - Fitur seperti wadah istimewa atau akses saat ini tidak tersedia di Fargate. Ini akan memengaruhi kasus penggunaan seperti menjalankan Docker di Docker.
- Akses terbatas ke kemampuan Linux - Lingkungan di mana kontainer berjalan di Fargate dikunci. Kemampuan Linux tambahan, seperti CAP_SYS_ADMIN dan CAP_NET_ADMIN, dibatasi untuk mencegah eskalasi hak istimewa. Fargate mendukung penambahan kapabilitas CAP_SYS_PTRACE Linux ke tugas-tugas untuk memungkinkan observabilitas dan alat keamanan yang digunakan dalam tugas untuk memantau aplikasi container.
- Tidak ada akses ke host yang mendasarinya - Baik pelanggan maupun AWS operator tidak dapat terhubung ke host yang menjalankan beban kerja pelanggan. Anda dapat menggunakan ECS exec untuk menjalankan perintah atau mendapatkan shell ke wadah yang berjalan di Fargate. Anda dapat menggunakan ECS exec untuk membantu mengumpulkan informasi diagnostik untuk debugging. Fargate juga mencegah kontainer mengakses sumber daya host yang mendasarinya, seperti sistem file, perangkat, jaringan, dan runtime kontainer.

- Jaringan - Anda dapat menggunakan grup keamanan dan jaringan ACLs untuk mengontrol lalu lintas masuk dan keluar. Tugas Fargate menerima alamat IP dari subnet yang dikonfigurasi di VPC Anda.

Versi platform Fargate untuk Amazon ECS

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Ini adalah kombinasi dari kernel dan versi runtime container. Anda memilih versi platform ketika Anda menjalankan tugas atau ketika Anda membuat layanan untuk mempertahankan sejumlah tugas yang identik.

Revisi baru versi platform dirilis saat lingkungan runtime berkembang, misalnya, jika ada pembaruan kernel atau sistem operasi, fitur baru, perbaikan bug, atau pembaruan keamanan. Versi platform Fargate diperbarui dengan membuat revisi versi platform baru. Setiap tugas berjalan pada satu revisi versi platform selama siklus hidupnya. Jika Anda ingin menggunakan revisi versi platform terbaru, maka Anda harus memulai tugas baru. Tugas baru yang berjalan di Fargate selalu berjalan pada revisi terbaru dari versi platform, memastikan bahwa tugas selalu dimulai pada infrastruktur yang aman dan ditambal.

Jika ditemukan masalah keamanan yang memengaruhi versi platform yang ada, AWS buat revisi baru yang ditambal dari versi platform dan menghentikan tugas yang berjalan pada revisi yang rentan. Dalam beberapa kasus, Anda mungkin diberi tahu bahwa tugas Anda di Fargate telah dijadwalkan untuk pensiun. Untuk informasi selengkapnya, lihat Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS .

Anda menentukan versi platform saat menjalankan tugas, atau menerapkan layanan.

Pertimbangkan hal berikut saat menentukan versi platform:

- Anda dapat menentukan nomor versi tertentu, misalnya`1.4.0`, atau`LATEST`.

  Versi platform Linux terbaru adalah`1.4.0`.

  Versi platform Windows TERBARU adalah`1.0.0`.
- Jika Anda ingin memperbarui versi platform untuk suatu layanan, buat penerapan. Misalnya, asumsikan bahwa Anda memiliki layanan yang menjalankan tugas pada versi platform Linux`1.3.0`. Untuk mengubah layanan untuk menjalankan tugas pada versi platform Linux`1.4.0`, Anda memperbarui layanan Anda dan menentukan versi platform baru. Tugas Anda dipindahkan

dengan versi platform terbaru dan revisi versi platform terbaru. Untuk informasi lebih lanjut tentang deployment, lihat Layanan-layanan Amazon ECS.

- Jika layanan Anda ditingkatkan tanpa memperbarui versi platform, tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Misalnya, asumsikan bahwa Anda memiliki layanan yang menjalankan tugas pada versi platform Linux`1.3.0`. Jika Anda meningkatkan jumlah layanan yang diinginkan, penjadwal layanan memulai tugas baru menggunakan revisi versi platform terbaru dari versi platform. `1.3.0`
- Tugas baru selalu berjalan pada revisi terbaru dari versi platform. Ini memastikan tugas selalu pada infrastruktur yang aman dan ditambal.
- Nomor versi platform untuk wadah Linux dan wadah Windows di Fargate bersifat independen. Misalnya, perilaku, fitur, dan perangkat lunak yang digunakan dalam versi platform `1.0.0` untuk wadah Windows di Fargate tidak sebanding dengan versi platform `1.0.0` untuk wadah Linux di Fargate.
- Berikut ini berlaku untuk versi platform Fargate Windows.

  Gambar kontainer Microsoft Windows Server harus dibuat dari versi Windows Server tertentu. Anda harus memilih versi Windows Server yang sama `platformFamily` ketika Anda menjalankan tugas atau membuat layanan yang cocok dengan gambar wadah Windows Server. Selain itu, Anda dapat memberikan pencocokan `operatingSystemFamily` dalam definisi tugas untuk mencegah tugas dijalankan pada versi Windows yang salah. Untuk informasi selengkapnya, lihat Mencocokkan versi host kontainer dengan versi gambar kontainer di situs web Microsoft Learn.

Migrasi ke platform Linux versi 1.4.0 di Amazon ECS

Pertimbangkan hal berikut saat memigrasikan Amazon ECS Anda pada tugas Fargate dari versi platform`1.0.0`, `1.1.0`1.2.0, atau `1.3.0` ke versi platform. `1.4.0` Ini adalah praktik terbaik untuk mengonfirmasi tugas Anda berfungsi dengan baik pada versi platform `1.4.0` sebelum Anda memigrasikan tugas.

- Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4.0, semua Amazon ECS pada tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua arus lalu lintas jaringan melalui ENI tersebut dalam VPC Anda. Lalu lintas dapat dilihat oleh Anda melalui log aliran VPC Anda. Untuk mengetahui informasi selengkapnya, lihat Opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate.
- Jika Anda menggunakan titik akhir VPC antarmuka, pertimbangkan hal berikut.

- Untuk gambar kontainer yang dihosting dengan Amazon ECR, Anda memerlukan titik akhir berikut. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - com.amazonaws. ***region***.ecr.dkr Titik akhir Amazon ECR VPC
  - com.amazonaws. ***region***.ecr.api Titik akhir Amazon ECR VPC
  - Titik akhir gerbang Amazon S3
- Ketika definisi tugas Anda mereferensikan rahasia Secrets Manager untuk mengambil data sensitif untuk container Anda, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .
- Ketika definisi tugas Anda mereferensikan parameter Systems Manager Parameter Store untuk mengambil data sensitif untuk container Anda, Anda harus membuat titik akhir VPC antarmuka untuk Systems Manager. Untuk informasi selengkapnya, lihat Meningkatkan keamanan EC2 instans dengan menggunakan titik akhir VPC untuk Systems Manager di AWS Systems Manager Panduan Pengguna.
- Grup keamanan untuk Elastic Network Interface (ENI) yang terkait dengan tugas Anda memerlukan aturan grup keamanan untuk mengizinkan lalu lintas antara tugas dan titik akhir VPC.

Log perubahan versi platform Fargate Linux

Berikut ini adalah versi platform Linux yang tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

1.4.0

Berikut ini adalah changelog untuk versi platform. `1.4.0`

- Mulai 5 November 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan dapat menggunakan fitur-fitur berikut:
  - Saat menggunakan Secrets Manager untuk menyimpan data sensitif, Anda dapat menyuntikkan kunci JSON tertentu atau versi rahasia tertentu sebagai variabel lingkungan atau dalam konfigurasi log. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

  - Tentukan variabel lingkungan dalam jumlah besar menggunakan parameter ketentuan kontainer `environmentFiles`. Untuk informasi selengkapnya, lihat Meneruskan variabel lingkungan individual ke wadah Amazon ECS.
  - Tugas yang dijalankan di VPC dan subnet diaktifkan untuk IPv6 akan diberikan IPv4 alamat pribadi dan alamat. IPv6 Untuk informasi selengkapnya, lihat opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate.
  - Titik akhir tugas metadata versi 4 menyediakan metadata tambahan tentang tugas dan kontainer termasuk tipe peluncuran tugas, Amazon Resource Name (ARN) kontainer, serta driver log dan opsi log driver yang digunakan. Saat melakukan kueri terhadap titik akhir `/stats` Anda juga menerima statistik tingkat jaringan untuk kontainer Anda. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas versi 4.
- Mulai 30 Juli 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan dapat merutekan lalu lintas UDP menggunakan Network Load Balancer ke Amazon ECS mereka pada tugas Fargate. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.
- Mulai 28 Mei 2020, setiap tugas Amazon ECS baru yang diluncurkan di Fargate menggunakan `1.4.0` versi platform akan memiliki penyimpanan fana yang dienkripsi dengan algoritme enkripsi AES-256 menggunakan kunci enkripsi yang dimiliki. AWS Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate untuk Amazon ECS dan Opsi penyimpanan untuk tugas Amazon ECS.
- Menambahkan dukungan untuk menggunakan volume sistem file Amazon EFS untuk penyimpanan tugas persisten. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.
- Penyimpanan tugas sementara telah ditingkatkan menjadi minimal 20 GB untuk setiap tugas. Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate untuk Amazon ECS.
- Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4.0, semua tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda dan akan terlihat oleh Anda melalui log aliran VPC Anda. Untuk informasi selengkapnya tentang jaringan untuk jenis EC2 peluncuran Amazon, lihat opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat. Opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate
- Tugas ENIs menambahkan dukungan untuk bingkai jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu

frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Dengan mendukung jumbo frame akan dapat mengurangi overhead ketika jalur jaringan antara tugas dengan tujuan Anda mendukung jumbo frame, seperti semua lalu lintas yang tetap berada dalam VPC Anda.

- CloudWatch Container Insights akan menyertakan metrik kinerja jaringan untuk tugas Fargate. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.
- Menambahkan dukungan untuk titik akhir metadata tugas versi 4 yang memberikan informasi tambahan untuk tugas Fargate Anda, termasuk statistik jaringan untuk tugas dan Zona Ketersediaan mana tugas sedang berjalan. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4 dan Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate.
- Menambahkan dukungan untuk parameter Linux `SYS_PTRACE` dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Parameter Linux.
- Agen kontainer Fargate menggantikan penggunaan agen kontainer Amazon ECS untuk semua tugas Fargate. Biasanya, perubahan ini tidak berpengaruh pada bagaimana tugas Anda berjalan.
- Waktu aktif kontainer kini menggunakan Containerd bukan Docker. Kemungkinan besar, perubahan ini tidak berpengaruh pada bagaimana tugas Anda berjalan. Anda akan melihat bahwa beberapa pesan kesalahan yang berasal dari runtime container berubah dari menyebutkan Docker menjadi kesalahan yang lebih umum. Untuk informasi selengkapnya, lihat Amazon ECS menghentikan pesan kesalahan tugas.
- Berdasarkan Amazon Linux 2.

1.3.0

Berikut ini adalah changelog untuk versi platform. `1.3.0`

- Mulai 30 September 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver log`awsfirelens`. FireLens Konfigurasikan Amazon ECS untuk menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan Jaringan AWS Mitra (APN) untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.
- Menambahkan daur ulang tugas untuk tugas Fargate, yang merupakan proses menyegarkan tugas yang merupakan bagian dari layanan Amazon ECS. Untuk informasi lebih lanjut, Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS.

- Dimulai pada 27 Maret 2019, tugas Fargate baru apa pun yang diluncurkan dapat menggunakan parameter definisi tugas tambahan yang Anda gunakan untuk menentukan konfigurasi proxy, dependensi untuk startup dan shutdown kontainer serta nilai batas waktu mulai dan berhenti per kontainer. Lihat informasi selengkapnya di Konfigurasi proxy, Dependensi kontainer, dan Waktu habis kontainer.
- Dimulai pada 2 April 2019, tugas Fargate baru apa pun yang diluncurkan mendukung penyuntikan data sensitif ke dalam wadah Anda dengan menyimpan data sensitif Anda baik dalam AWS Secrets Manager rahasia atau AWS Systems Manager parameter Parameter Store dan kemudian merujuknya dalam definisi penampung Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.
- Mulai 1 Mei 2019, tugas Fargate baru apa pun yang diluncurkan mendukung referensi data sensitif dalam konfigurasi log wadah menggunakan parameter definisi `secretOptions` wadah. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.
- Mulai 1 Mei 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver `splunk` log selain driver `awslogs` log. Untuk informasi selengkapnya, lihat Penyimpanan dan pencatatan.
- Mulai 9 Juli 2019, setiap tugas Fargate baru yang diluncurkan mendukung CloudWatch Wawasan Kontainer. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.
- Mulai 3 Desember 2019, penyedia kapasitas Fargate Spot didukung. Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk Fargate.
- Berdasarkan Amazon Linux 2.

AWS Penghentian versi platform Fargate Linux

Halaman ini mencantumkan versi platform Linux yang AWS Fargate tidak digunakan lagi atau telah dijadwalkan untuk dihentikan. Versi platform ini tetap tersedia hingga tanggal penghentian yang dipublikasikan.

Tanggal pembaruan paksa disediakan untuk setiap versi platform yang dijadwalkan untuk penghentian. Pada tanggal pembaruan paksa, layanan apa pun yang menggunakan versi `LATEST` platform yang diarahkan ke versi platform yang dijadwalkan untuk penghentian akan diperbarui menggunakan opsi force new deployment. Ketika layanan diperbarui menggunakan opsi force new deployment, semua tugas yang berjalan pada versi platform yang dijadwalkan untuk penghentian akan dihentikan dan tugas baru diluncurkan menggunakan versi platform yang ditunjuk `LATEST` tag

pada saat itu. Tugas atau layanan mandiri dengan set versi platform eksplisit tidak terpengaruh oleh tanggal pembaruan paksa.

Kami menyarankan untuk memperbarui tugas mandiri layanan Anda untuk menggunakan versi platform terbaru. Untuk informasi selengkapnya tentang migrasi ke versi platform terbaru, lihatMigrasi ke platform Linux versi 1.4.0 di Amazon ECS.

Setelah versi platform mencapai tanggal penghentian, versi platform tidak akan lagi tersedia untuk tugas atau layanan baru. Setiap tugas atau layanan mandiri yang secara eksplisit menggunakan versi platform yang tidak digunakan lagi akan terus menggunakan versi platform tersebut hingga tugas dihentikan. Setelah tanggal penghentian, versi platform yang tidak digunakan lagi tidak akan lagi menerima pembaruan keamanan atau perbaikan bug.

Versi platform	Tanggal pembaruan paksa	Tanggal pengusangan
1.0.0	26 Oktober 2020	14 Desember 2020
1.1.0	26 Oktober 2020	14 Desember 2020
1.2.0	26 Oktober 2020	14 Desember 2020

Untuk informasi tentang versi platform saat ini, lihat Versi platform Fargate untuk Amazon ECS.

Log perubahan versi Fargate Linux yang tidak digunakan lagi

1.2.0

Berikut ini adalah changelog untuk versi platform. `1.2.0`

> **Note**
>
> Versi platform `1.2.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Menambahkan dukungan untuk otentikasi registri pribadi menggunakan AWS Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

1.1.0

Berikut ini adalah changelog untuk versi platform. `1.1.0`

> **Note**
>
> Versi platform `1.1.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Menambahkan dukungan untuk titik akhir metadata tugas Amazon ECS. Untuk informasi selengkapnya, lihat Metadata tugas Amazon ECS tersedia untuk tugas di Fargate.
- Menambahkan dukungan untuk pemeriksaan kondisi Docker dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.
- Menambahkan dukungan untuk penemuan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.

1.0.0

Berikut ini adalah changelog untuk versi platform. `1.0.0`

> **Note**
>
> Versi platform `1.0.0` tidak lagi tersedia. Untuk informasi tentang penghentian versi platform, lihat. AWS Penghentian versi platform Fargate Linux

- Berdasarkan Amazon Linux 2017.09.
- Pelepasan awal.

Log perubahan versi platform Windows Fargate

Berikut ini adalah versi platform yang tersedia untuk wadah Windows.

1.0.0

Berikut ini adalah changelog untuk versi platform. `1.0.0`

- Rilis awal untuk dukungan pada sistem operasi Microsoft Windows Server berikut:
  - Windows Server 2019 Full
  - Windows Server 2019 Core
  - Windows Server 2022 Lengkap
  - Windows Server 2022 Inti

Wadah Windows tentang pertimbangan Fargate untuk Amazon ECS

Berikut ini adalah perbedaan dan pertimbangan untuk mengetahui kapan Anda menjalankan wadah Windows di AWS Fargate.

Jika Anda perlu menjalankan tugas di wadah Linux dan Windows, maka Anda perlu membuat definisi tugas terpisah untuk setiap sistem operasi.

AWS menangani manajemen lisensi sistem operasi, sehingga Anda tidak memerlukan lisensi Microsoft Windows Server tambahan.

Kontainer Windows di AWS Fargate mendukung sistem operasi berikut:

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Lengkap
- Windows Server 2022 Inti

Kontainer Windows di AWS Fargate mendukung driver awslogs. Untuk informasi selengkapnya, lihat the section called “Kirim log ke CloudWatch ”.

Fitur-fitur berikut tidak didukung pada wadah Windows di Fargate:

- Amazon FSx
- ENI trunking
- g MSAs untuk Windows Container
- Layanan App Mesh dan integrasi proxy untuk tugas
- Integrasi router log Firelens untuk tugas

- Volume EFS
- Volume EBS
- Parameter definisi tugas berikut:
  - `maxSwap`
  - `swappiness`
  - `environmentFiles`
- Penyedia kapasitas Fargate Spot
- Volume gambar

  `volume`Opsi Dockerfile diabaikan. Sebagai gantinya, gunakan bind mount dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Gunakan bind mount dengan Amazon ECS.
- Parameter CPU dan memori tingkat tugas diabaikan untuk wadah Windows. Kami merekomendasikan penentuan sumber daya tingkat kontainer untuk Windows kontainer.
- memori untuk tugas
- MermoryReservasi pada kontainer
- Mulai ulang kebijakan tentang container

Penyimpanan sementara tugas Fargate untuk Amazon ECS

Saat disediakan, setiap tugas Amazon ECS yang dihosting di container Linux AWS Fargate menerima penyimpanan singkat berikut untuk pemasangan bind. Ini dapat dipasang dan dibagikan di antara kontainer yang menggunakan `volumesmountPoints`,, dan `volumesFrom` parameter dalam definisi tugas. Ini tidak didukung untuk wadah Windows aktif AWS Fargate.

Versi platform kontainer Fargate Linux

Versi 1.4.0 atau yang lebih baru

Secara default, tugas Amazon ECS yang di-host di Fargate menggunakan `1.4.0` versi platform atau yang lebih baru menerima minimal 20 GiB penyimpanan sementara. Jumlah total penyimpanan fana dapat ditingkatkan, hingga maksimum 200 GiB. Anda dapat melakukan ini dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.

Citra kontainer yang ditarik, dikompresi, dan tidak dikompresi untuk tugas disimpan di penyimpanan sementara. Untuk menentukan jumlah total penyimpanan sementara yang harus digunakan tugas

Anda, Anda harus mengurangi jumlah penyimpanan yang digunakan citra kontainer Anda dari jumlah total penyimpanan sementara yang dialokasikan tugas Anda.

Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru yang diluncurkan pada 28 Mei 2020 atau lebih baru, penyimpanan sementara dienkripsi dengan algoritme enkripsi AES-256. Algoritma ini menggunakan kunci enkripsi yang AWS dimiliki, atau Anda dapat membuat kunci yang dikelola pelanggan Anda sendiri. Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan AWS Fargate untuk penyimpanan sementara.

Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru yang diluncurkan pada 18 November 2022 atau lebih baru, penggunaan penyimpanan sementara dilaporkan melalui titik akhir metadata tugas. Aplikasi Anda dalam tugas Anda dapat menanyakan titik akhir metadata tugas versi 4 untuk mendapatkan ukuran cadangan penyimpanan sementara dan jumlah yang digunakan.

Selain itu, ukuran cadangan penyimpanan sementara dan jumlah yang digunakan akan dikirim ke Amazon CloudWatch Wawasan Kontainer jika Anda mengaktifkan Wawasan Kontainer.

> **Note**
>
> Fargate mencadangkan ruang pada disk. Ruang ini hanya digunakan oleh Fargate. Anda tidak akan dikenai biaya untuk ruang tersebut. Ruang ini tidak akan ditampilkan dalam metrik ini. Namun demikian, Anda dapat melihat penyimpanan tambahan ini di alat lain seperti `df`.

Versi 1.3.0 atau sebelumnya

Untuk Amazon ECS pada tugas Fargate yang menggunakan `1.3.0` versi platform atau versi sebelumnya, setiap tugas menerima penyimpanan singkat berikut.

- Penyimpanan lapisan Docker 10 GB

  > **Note**
  >
  > Jumlah ini mencakup artifact citra kontainer terkompresi dan tidak terkompresi.

- Tambahan 4 GB untuk pemasangan volume. Ini dapat dipasang dan dibagikan di antara kontainer yang menggunakan `volumesmountPoints`,, dan `volumesFrom` parameter dalam definisi tugas.

Fargate versi platform kontainer Windows

Versi 1.0.0 atau yang lebih baru

Secara default, tugas Amazon ECS yang di-host di Fargate menggunakan `1.0.0` versi platform atau yang lebih baru menerima minimal 20 GiB penyimpanan sementara. Jumlah total penyimpanan fana dapat ditingkatkan, hingga maksimum 200 GiB. Anda dapat melakukan ini dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.

Citra kontainer yang ditarik, dikompresi, dan tidak dikompresi untuk tugas disimpan di penyimpanan sementara. Untuk menentukan jumlah total penyimpanan sementara yang harus digunakan tugas Anda, Anda harus mengurangi jumlah penyimpanan yang digunakan gambar kontainer Anda dari jumlah total penyimpanan sementara yang dialokasikan tugas Anda.

Untuk informasi selengkapnya, lihat Gunakan bind mount dengan Amazon ECS.

Kunci yang dikelola pelanggan untuk penyimpanan AWS Fargate singkat untuk Amazon ECS

AWS Fargate mendukung kunci terkelola pelanggan untuk mengenkripsi data untuk tugas Amazon ECS yang disimpan dalam penyimpanan sementara untuk membantu pelanggan yang sensitif terhadap peraturan memenuhi kebijakan keamanan internal mereka. Pelanggan masih mendapatkan manfaat tanpa server dari Fargate, sambil memberikan visibilitas yang ditingkatkan pada enkripsi penyimpanan yang dikelola sendiri kepada auditor kepatuhan. Sementara Fargate memiliki enkripsi penyimpanan sementara yang dikelola Fargate secara default, pelanggan juga dapat menggunakan kunci yang dikelola sendiri saat mengenkripsi data sensitif seperti informasi terkait keuangan atau kesehatan.

Anda dapat mengimpor kunci Anda sendiri ke dalam AWS KMS atau membuat kunci di AWS KMS. Kunci yang dikelola sendiri ini disimpan AWS KMS dan melakukan tindakan AWS KMS siklus hidup standar seperti memutar, menonaktifkan, dan menghapus. Anda dapat mengaudit akses kunci dan penggunaan dalam CloudTrail log.

Secara default, kunci KMS mendukung 50.000 hibah per kunci. Fargate menggunakan AWS KMS hibah tunggal per tugas kunci yang dikelola pelanggan, sehingga mendukung hingga 50.000 tugas bersamaan untuk sebuah kunci. Jika Anda ingin menambah jumlah ini, Anda dapat meminta kenaikan batas, yang disetujui atas case-by-case dasar.

Fargate tidak mengenakan biaya tambahan apa pun untuk menggunakan kunci yang dikelola pelanggan. Anda hanya dikenakan harga standar untuk menggunakan AWS KMS kunci untuk penyimpanan dan permintaan API.

Topik

- Buat kunci enkripsi untuk penyimpanan sementara Fargate untuk Amazon ECS
- Mengelola AWS KMS kunci untuk penyimpanan fana Fargate untuk Amazon ECS

Buat kunci enkripsi untuk penyimpanan sementara Fargate untuk Amazon ECS

Buat kunci yang dikelola pelanggan untuk mengenkripsi data yang disimpan di penyimpanan sementara Fargate.

> **Note**
>
> Enkripsi penyimpanan fana Fargate dengan kunci yang dikelola pelanggan tidak tersedia untuk cluster tugas Windows.
> Enkripsi penyimpanan fana Fargate dengan kunci yang dikelola pelanggan tidak tersedia lebih awal dari sebelumnya. `platformVersions` `1.4.0`
> Fargate menyediakan ruang pada penyimpanan sementara yang hanya digunakan oleh Fargate, dan Anda tidak ditagih untuk ruang tersebut. Alokasi mungkin berbeda dari tugas-tugas utama yang dikelola non-pelanggan, tetapi total ruang tetap sama. Anda dapat melihat perubahan ini di alat seperti`df`.
> Kunci Multi-Wilayah tidak didukung untuk penyimpanan sementara Fargate.
> Alias kunci KMS tidak didukung untuk penyimpanan sementara Fargate.

Untuk membuat kunci terkelola pelanggan (CMK) untuk mengenkripsi penyimpanan sementara untuk Fargate, ikuti langkah-langkah ini. AWS KMS

1. Arahkan https://console.aws.amazon.comke/kms.
2. Ikuti petunjuk untuk Membuat Kunci di Panduan AWS Key Management Service Pengembang.
3. Saat membuat AWS KMS kunci Anda, pastikan untuk memberikan izin AWS KMS operasi yang relevan dengan layanan Fargate dalam kebijakan utama. Operasi API berikut harus diizinkan dalam kebijakan untuk menggunakan kunci terkelola pelanggan Anda dengan sumber daya klaster Amazon ECS Anda.

- `kms:GenerateDataKeyWithoutPlainText`- Panggilan `GenerateDataKeyWithoutPlainText` untuk menghasilkan kunci data terenkripsi dari kunci yang disediakan AWS KMS .
- `kms:CreateGrant`- Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses kontrol ke AWS KMS kunci tertentu, yang memungkinkan akses untuk memberikan operasi yang diperlukan Amazon ECS Fargate. Untuk informasi selengkapnya tentang Menggunakan Hibah, lihat Panduan AWS Key Management Service Pengembang. Ini memungkinkan Amazon ECS Fargate untuk melakukan hal berikut:
  - Panggil `Decrypt` AWS KMS untuk mendapatkan kunci enkripsi untuk mendekripsi data penyimpanan sementara.
  - Siapkan kepala sekolah yang pensiun untuk memungkinkan layanan. `RetireGrant`
- `kms:DescribeKey`- Menyediakan detail kunci yang dikelola pelanggan untuk memungkinkan Amazon ECS memvalidasi kunci jika simetris dan diaktifkan.

Contoh berikut menunjukkan kebijakan AWS KMS kunci yang akan Anda terapkan pada kunci target untuk enkripsi. Untuk menggunakan contoh pernyataan kebijakan, ganti *`user input placeholders`* dengan informasi Anda sendiri. Seperti biasa, hanya konfigurasikan izin yang Anda butuhkan, tetapi Anda harus memberikan AWS KMS izin kepada setidaknya satu pengguna untuk menghindari kesalahan.

```
{
      "Sid": "Allow generate data key access for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        }
      },
      "Resource": "*"
```

```
    },
    {
      "Sid": "Allow grant creation permission for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:CreateGrant"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        },
       "ForAllValues:StringEquals": {
          "kms:GrantOperations": [
             "Decrypt"
          ]
       }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow describe key permission for cluster operator - CreateCluster
 and UpdateCluster.",
      "Effect": "Allow",
      "Principal": { "AWS":"arn:aws:iam::customerAccountId:role/customer-chosen-
role" },
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
```

Tugas Fargate menggunakan kunci konteks `aws:ecs:clusterAccount` dan `aws:ecs:clusterName` enkripsi untuk operasi kriptografi dengan kunci. Pelanggan harus menambahkan izin ini untuk membatasi akses ke klaster akun and/or tertentu. Gunakan nama cluster dan bukan ARN saat Anda menentukan cluster.

Untuk informasi lebih lanjut, lihat Konteks enkripsi di Panduan Developer AWS KMS.

Saat membuat atau memperbarui cluster, Anda memiliki opsi untuk menggunakan kunci kondisi`fargateEphemeralStorageKmsKeyId`. Kunci kondisi ini memungkinkan pelanggan untuk memiliki kontrol yang lebih terperinci terhadap kebijakan IAM. Pembaruan `fargateEphemeralStorageKmsKeyId` konfigurasi hanya berlaku pada penerapan layanan baru.

Berikut ini adalah contoh mengizinkan pelanggan untuk memberikan izin hanya untuk satu set AWS KMS kunci tertentu yang disetujui.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ecs:fargate-ephemeral-storage-kms-key": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
      }
    }
  ]
}
```

Berikutnya adalah contoh untuk menolak upaya untuk menghapus AWS KMS kunci yang sudah dikaitkan dengan cluster.

JSON

```
{
  "Version": "2012-10-17",
```

```
  "Statement": {
    "Effect": "Deny",
    "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
    ],
    "Resource": "*",
    "Condition": {
      "Null": {
        "ecs:fargate-ephemeral-storage-kms-key": "true"
      }
    }
  }
}
```

Pelanggan dapat melihat apakah tugas atau tugas layanan mereka yang tidak dikelola dienkripsi menggunakan kunci dengan menggunakan AWS CLI `describe-tasks`,`describe-cluster`, atau perintah. `describe-services`

Untuk informasi selengkapnya, lihat Kunci kondisi untuk AWS KMS di Panduan AWS KMS Pengembang.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pilih Cluster di navigasi kiri dan Buat cluster di kanan atas, atau pilih cluster yang ada. Untuk klaster yang ada, pilih Perbarui klaster di kanan atas.
3. Di bawah bagian Enkripsi alur kerja, Anda akan memiliki opsi untuk memilih AWS KMS kunci Anda di bawah Penyimpanan terkelola dan penyimpanan sementara Fargate. Anda juga dapat memilih untuk membuat AWS KMS kunci dari sini.
4. Pilih Buat setelah Anda selesai membuat klaster baru atau Perbarui, jika Anda memperbarui yang sudah ada.

AWS CLI

Berikut ini adalah contoh membuat cluster dan mengonfigurasi penyimpanan sementara Fargate Anda menggunakan (ganti *red* nilainya dengan AWS CLI milik Anda sendiri):

```
aws ecs create-cluster --cluster clusterName \
--configuration '{"managedStorageConfiguration":
{"fargateEphemeralStorageKmsKeyId":"arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}}'
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:012345678901:cluster/clusterName",
        "clusterName": "clusterName",
        "configuration": {
            "managedStorageConfiguration": {
                "fargateEphemeralStorageKmsKeyId": "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": []
    },
    "clusterCount": 5
}
```

AWS CloudFormation

Berikut ini adalah contoh template untuk membuat cluster dan mengonfigurasi penyimpanan sementara Fargate Anda menggunakan (ganti *red* nilainya dengan AWS CloudFormation milik Anda sendiri):

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: "clusterName"
      Configuration:
        ManagedStorageConfiguration:
```

```
        FargateEphemeralStorageKmsKeyId: "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Mengelola AWS KMS kunci untuk penyimpanan fana Fargate untuk Amazon ECS

Setelah membuat atau mengimpor AWS KMS kunci Anda untuk mengenkripsi penyimpanan sementara Fargate Anda, Anda mengelolanya dengan cara yang sama seperti yang Anda lakukan pada kunci lainnya. AWS KMS

Rotasi AWS KMS tombol otomatis

Anda dapat mengaktifkan rotasi tombol otomatis atau memutarnya secara manual. Rotasi kunci otomatis memutar kunci untuk Anda setiap tahun dengan menghasilkan materi kriptografi baru untuk kunci tersebut. AWS KMS juga menyimpan semua versi sebelumnya dari materi kriptografi, sehingga Anda akan dapat mendekripsi data apa pun yang menggunakan versi kunci sebelumnya. Materi yang diputar tidak akan dihapus AWS KMS sampai Anda menghapus kunci.

Rotasi tombol otomatis bersifat opsional dan dapat diaktifkan atau dinonaktifkan kapan saja.

Menonaktifkan atau mencabut kunci AWS KMS

Jika Anda menonaktifkan kunci yang dikelola pelanggan AWS KMS, itu tidak berdampak pada menjalankan tugas, dan mereka terus berfungsi melalui siklus hidupnya. Jika tugas baru menggunakan kunci yang dinonaktifkan atau dicabut, tugas gagal karena tidak dapat mengakses kunci. Anda harus mengatur CloudWatch alarm atau serupa untuk memastikan kunci yang dinonaktifkan tidak pernah diperlukan untuk mendekripsi data yang sudah dienkripsi.

Menghapus kunci AWS KMS

Menghapus kunci harus selalu menjadi pilihan terakhir dan hanya boleh dilakukan jika Anda yakin kunci yang dihapus tidak pernah diperlukan lagi. Tugas baru yang mencoba menggunakan kunci yang dihapus akan gagal karena mereka tidak dapat mengaksesnya. AWS KMS menyarankan untuk menonaktifkan kunci alih-alih menghapusnya. Jika Anda merasa perlu untuk menghapus kunci, kami sarankan menonaktifkannya terlebih dahulu dan mengatur CloudWatch alarm untuk memastikan itu tidak diperlukan. Jika Anda menghapus kunci, AWS KMS sediakan setidaknya tujuh hari untuk mengubah pikiran Anda.

Mengaudit akses AWS KMS kunci

Anda dapat menggunakan CloudTrail log untuk mengaudit akses ke AWS KMS kunci Anda. Anda dapat memeriksa AWS KMS operasi`CreateGrant`,`GenerateDataKeyWithoutPlaintext`,

danDecrypt. Operasi ini juga menunjukkan aws:ecs:clusterAccount dan aws:ecs:clusterName sebagai bagian dari EncryptionContext login CloudTrail.

Berikut ini adalah contoh CloudTrail peristiwa untukGenerateDataKeyWithoutPlaintext,GenerateDataKeyWithoutPlaintext (DryRun),CreateGrant,CreateGrant (DryRun), dan RetireGrant (ganti *red* nilai dengan milik Anda sendiri).

GenerateDataKeyWithoutPlaintext

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 64,
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ebs:id": "vol-xxxxxxx",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

GenerateDataKeyWithoutPlaintext (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "dryRun": true,
        "numberOfBytes": 64,
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ebs:id": "vol-xxxx",
                "aws:ecs:clusterName": "cluster-name"
            }
        },
        "retiringPrincipal": "ec2.us-west-2.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
```

```
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "dryRun": true,
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ecs:clusterName": "cluster-name"
            }
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

RetireGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2024-04-20T18:37:38Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "additionalEventData": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS

AWS bertanggung jawab untuk menjaga infrastruktur yang mendasari AWS Fargate. AWS menentukan kapan revisi versi platform perlu diganti dengan revisi baru untuk infrastruktur. Ini dikenal sebagai pensiun tugas. AWS mengirimkan pemberitahuan pensiun tugas ketika revisi versi platform dihentikan. Kami secara rutin memperbarui versi platform kami yang didukung untuk memperkenalkan revisi baru yang berisi pembaruan pada perangkat lunak runtime Fargate dan dependensi yang mendasarinya seperti sistem operasi dan runtime kontainer. Setelah revisi yang lebih baru tersedia, kami menghentikan revisi lama untuk memastikan semua beban kerja pelanggan

berjalan pada revisi terbaru dari versi platform Fargate. Ketika revisi dihentikan, semua tugas yang berjalan pada revisi itu dihentikan.

Tugas Amazon ECS dapat dikategorikan sebagai tugas layanan atau tugas mandiri. Tugas layanan digunakan sebagai bagian dari layanan dan dikendalikan oleh jadwal Amazon ECS. Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS. Tugas mandiri adalah tugas yang dimulai oleh Amazon ECS `RunTask` API, baik secara langsung atau oleh penjadwal eksternal seperti tugas terjadwal (yang dimulai oleh Amazon EventBridge), AWS Batch atau. AWS Step Functions Anda tidak perlu mengambil tindakan apa pun dalam menanggapi pensiun tugas untuk tugas layanan Anda karena penjadwal Amazon ECS secara otomatis menggantikan tugas.

Untuk tugas mandiri, Anda mungkin perlu melakukan penanganan tambahan dalam menanggapi pensiun tugas. Untuk informasi selengkapnya, lihat Bisakah Amazon ECS secara otomatis menangani tugas mandiri?.

Untuk tugas layanan, Anda tidak perlu mengambil tindakan apa pun untuk pensiun tugas kecuali Anda ingin mengganti tugas-tugas ini sebelum AWS melakukannya. Ketika penjadwal Amazon ECS menghentikan tugas, ia menggunakan `maximumPercent` dan meluncurkan tugas baru dalam upaya untuk mempertahankan jumlah yang diinginkan untuk layanan. Ikuti praktik terbaik untuk meminimalkan dampak pensiun tugas. `maximumPercent`Nilai default untuk layanan yang menggunakan penjadwal layanan REPLICA adalah 200%. Oleh karena itu, ketika AWS Fargate mulai pensiun tugas, Amazon ECS pertama-tama menjadwalkan tugas baru, dan kemudian menunggu sampai tugas itu berjalan, sebelum menghentikan tugas lama. Saat Anda menetapkan `maximumPercent` nilainya menjadi 100%, Amazon ECS menghentikan tugas terlebih dahulu, lalu menggantinya.

Untuk pensiun tugas mandiri, AWS hentikan tugas pada atau setelah tanggal pensiun tugas. Amazon ECS tidak meluncurkan tugas pengganti saat tugas dihentikan. Jika Anda memerlukan tugas-tugas ini untuk terus berjalan, Anda harus menghentikan tugas yang sedang berjalan dan meluncurkan tugas pengganti sebelum waktu yang ditunjukkan dalam pemberitahuan. Oleh karena itu, kami menyarankan agar pelanggan memantau keadaan tugas mandiri dan jika diperlukan, terapkan logika untuk mengganti tugas yang dihentikan.

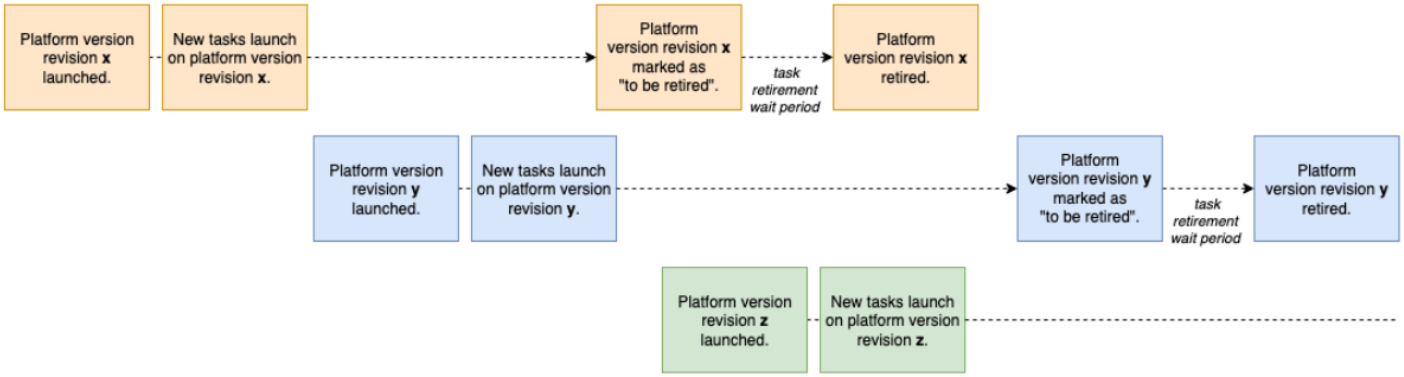
Ketika tugas dihentikan di salah satu skenario, Anda dapat menjalankannya`describe-tasks`. Yang `stoppedReason` dalam tanggapannya adalah`ECS is performing maintenance on the underlying infrastructure hosting the task`.

Pemeliharaan tugas berlaku ketika ada revisi versi platform baru perlu diganti dengan revisi baru. Jika ada masalah dengan host Fargate yang mendasarinya, Amazon ECS menggantikan host tanpa pemberitahuan pensiun tugas.

Ikhtisar pemberitahuan pensiun tugas

Saat AWS menandai revisi versi platform sebagai perlu dihentikan, kami mengidentifikasi semua tugas yang berjalan pada revisi versi platform tersebut di semua Wilayah. Kami kemudian mengirimkan satu pemberitahuan per akun per Wilayah, menyoroti tugas atau layanan yang terpengaruh dan tanggal kapan pensiun akan mulai berlangsung.

Ilustrasi berikut menunjukkan siklus hidup revisi versi platform Fargate dari peluncuran revisi baru ke pensiun revisi platform.



Informasi berikut memberikan rincian.

- Setelah revisi versi platform baru diluncurkan, semua tugas baru dijadwalkan pada revisi ini.
- Tugas yang ada yang telah dijadwalkan dan berjalan tetap pada revisi yang awalnya ditempatkan selama durasi tugas dan tidak dimigrasikan ke revisi baru.
- Tugas baru, misalnya sebagai bagian dari pembaruan layanan atau pensiun tugas Fargate, ditempatkan pada revisi versi platform terbaru yang tersedia pada saat peluncuran.

Pemberitahuan pensiun tugas dikirim melalui AWS Health Dasbor serta melalui email ke alamat email yang terdaftar dan mencakup informasi berikut:

- Tanggal pensiun tugas - Tugas dihentikan pada atau setelah tanggal ini.
- Untuk tugas mandiri, IDs tugas.
- Untuk tugas layanan, ID cluster tempat layanan berjalan dan layanan. IDs

- Langkah selanjutnya yang perlu Anda ambil.

Biasanya, kami mengirim satu pemberitahuan masing-masing untuk layanan dan tugas mandiri di masing-masing Wilayah AWS. Namun, dalam kasus tertentu Anda mungkin menerima lebih dari satu peristiwa untuk setiap jenis tugas, misalnya ketika ada terlalu banyak tugas yang harus dihentikan yang akan melampaui batas dalam mekanisme pemberitahuan kami.

Anda dapat mengidentifikasi tugas yang dijadwalkan untuk pensiun dengan cara berikut:

- The AWS Health Dashboard

  AWS Health pemberitahuan dapat dikirim melalui Amazon EventBridge ke penyimpanan arsip seperti Amazon Simple Storage Service, mengambil tindakan otomatis seperti menjalankan AWS Lambda fungsi, atau sistem notifikasi lainnya seperti Amazon Simple Notification Service. Untuk informasi selengkapnya, lihat Memantau AWS Health peristiwa dengan Amazon EventBridge. Untuk konfigurasi sampel untuk mengirim notifikasi ke Amazon Chime, Slack, atau Microsoft Teams, lihat repositori AWS Health Aware aktif. GitHub

  Berikut ini adalah contoh EventBridge acara.

```
{
    "version": "0",
    "id": "3c268027-f43c-0171-7425-1d799EXAMPLE",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-08-16T23:18:51Z",
    "region": "us-east-1",
    "resources": [
        "cluster|service",
        "cluster|service"
    ],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/ECS/
AWS_ECS_TASK_PATCHING_RETIREMENT/AWS_ECS_TASK_PATCHING_RETIREMENT_test1",
        "service": "ECS",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
        "communicationId":
 "7988399e2e6fb0b905ddc88e0e2de1fd17e4c9fa60349577446d95a18EXAMPLE",
        "lastUpdatedTime": "Wed, 16 Aug 2023 23:18:52 GMT",
        "eventRegion": "us-east-1",
```

```
        "eventTypeCode": "AWS_ECS_TASK_PATCHING_RETIREMENT",
        "eventTypeCategory": "scheduledChange",
        "startTime": "Wed, 16 Aug 2023 23:18:51 GMT",
        "endTime": "Fri, 18 Aug 2023 23:18:51 GMT",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "\\nA software update has been deployed to
 Fargate which includes CVE patches or other critical patches. No action is required
 on your part. All new tasks launched automatically uses the latest software
 version. For existing tasks, your tasks need to be restarted in order for these
 updates to apply. Your tasks running as part of the following ECS Services will
 be automatically updated beginning Wed, 16 Aug 2023 23:18:51 GMT.\\n\\nAfter Wed,
 16 Aug 2023 23:18:51 GMT, the ECS scheduler will gradually replace these tasks,
 respecting the deployment settings for your service. Typically, services should
 see little to no interruption during the update and no action is required. When AWS
 stops tasks, AWS uses the minimum healthy percent (1) and launches a new task in
 an attempt to maintain the desired count for the service. By default, the minimum
 healthy percent of a service is 100 percent, so a new task is started first before
 a task is stopped. Service tasks are routinely replaced in the same way when
 you scale the service or deploy configuration changes or deploy task definition
 revisions. If you would like to control the timing of this restart you can update
 the service before Wed, 16 Aug 2023 23:18:51 GMT, by running the update-service
 command from the ECS command-line interface specifying force-new-deployment for
 services using Rolling update deployment type. For example:\\n\\n$ aws ecs update-
service -service service_name \\\n--cluster cluster_name -force-new-deployment\
\n\\nFor services using Blue/Green deployment type with AWS CodeDeploy:\\nPlease
 refer to create-deployment document (2) and create new deployment using same task
 definition revision.\\n\\nFor further details on ECS deployment types, please
 refer to ECS Deployment Developer Guide (1).\\nFor further details on Fargate's
 update process, please refer to the AWS Fargate User Guide (3).\\nIf you have
 any questions or concerns, please contact AWS Support (4).\\n\\n(1) https://
docs.aws.amazon.com/AmazonECS/latest/developerguide/deployment-types.html\\n(2)
 https://docs.aws.amazon.com/cli/latest/reference/deploy/create-deployment.html\\n(3)
 https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-maintenance.html\\n(4)
 https://aws.amazon.com/support\\n\\nA list of your affected resources(s) can be
 found in the 'Affected resources' tab in the 'Cluster/ Service' format in the AWS
 Health Dashboard. \\n\\n"
            }
        ],
      "affectedEntities": [
                {
                    "entityValue": "arn:aws:ecs:eu-west-1:111222333444:task/
examplecluster/00805ce1d81940b5a37398e5a2c23333"
```

```
                },
                {
                    "entityValue": "arn:aws:ecs:eu-west-1:111222333444:task/
examplecluster/00805ce1d81940b5a37398e5a2c25555"
                }
            }
        ]
    }
}
```

- Email

  Email dikirim ke email terdaftar untuk Akun AWS ID.

Untuk informasi tentang cara mempersiapkan pensiun tugas, lihatBersiaplah untuk pensiun tugas AWS Fargate di Amazon ECS .

Bisakah saya memilih keluar dari pensiun tugas?

Tidak. Sebagai bagian dari model tanggung jawab AWS bersama, AWS bertanggung jawab untuk mengelola dan memelihara infrastruktur yang mendasarinya AWS Fargate. Ini termasuk melakukan pembaruan platform berkala untuk memastikan keamanan dan stabilitas. Pembaruan ini diterapkan secara otomatis oleh AWS dan bukan sesuatu yang dapat dipilih pelanggan. Ini adalah manfaat utama menggunakan AWS Fargate dibandingkan dengan menjalankan beban kerja Anda pada EC2 instans, tanggung jawab untuk mempertahankan platform yang mendasarinya ditangani oleh. AWS Model ini memungkinkan Anda untuk fokus pada aplikasi Anda daripada pemeliharaan infrastruktur. Dengan menerapkan pembaruan platform ini secara otomatis, AWS dapat menjaga lingkungan Fargate up-to-date dan aman, tanpa tindakan apa pun yang diperlukan dari Anda sebagai pelanggan. Ini membantu menyediakan lingkungan kontainer yang andal dan aman untuk menjalankan beban kerja Anda di Fargate.

Bisakah saya mendapatkan pemberitahuan pensiun tugas melalui AWS layanan lain?

AWS mengirimkan pemberitahuan pensiun tugas ke AWS Health Dashboard dan ke kontak email utama di Akun AWS. AWS Health Dashboard Ini menyediakan sejumlah integrasi ke AWS layanan lain, termasuk EventBridge. Anda dapat menggunakan EventBridge untuk mengotomatiskan visibilitas pemberitahuan (Misalnya. meneruskan pesan ke alat). ChatOps Untuk informasi selengkapnya, lihat Ringkasan solusi: Menangkap pemberitahuan pensiun tugas.

Bisakah saya mengubah pensiun tugas setelah dijadwalkan?

Tidak. Jadwal didasarkan pada tugas waktu tunggu pensiun yang memiliki default 7 hari. Jika Anda membutuhkan lebih banyak waktu, Anda dapat memilih untuk mengkonfigurasi periode tunggu hingga 14 hari. Untuk informasi selengkapnya, lihat Langkah 2: Tangkap pemberitahuan pensiun tugas untuk memperingatkan tim dan mengambil tindakan. Perubahan dalam konfigurasi ini berlaku untuk pensiun yang akan dijadwalkan di masa depan. Pensiun yang dijadwalkan saat ini tidak terpengaruh. Jika Anda memiliki masalah lebih lanjut, hubungi Dukungan.

Bagaimana Amazon ECS menangani tugas yang merupakan bagian dari layanan?

Untuk tugas layanan, Anda tidak perlu mengambil tindakan apa pun dalam menanggapi pensiun tugas kecuali Anda ingin mengganti tugas-tugas ini sebelum AWS melakukannya. Ketika penjadwal Amazon ECS menghentikan tugas, ia menggunakan persentase sehat minimum dan meluncurkan tugas baru dalam upaya untuk mempertahankan jumlah yang diinginkan untuk layanan. Untuk meminimalkan dampak pensiun tugas Fargate, beban kerja harus diterapkan mengikuti praktik terbaik Amazon ECS. Misalnya, saat menerapkan aplikasi stateless sebagai layanan Amazon ECS, seperti server web atau API, pelanggan harus menerapkan beberapa replika tugas dan menyetel ke 100%. minimumHealthyPercent Secara default, persentase kesehatan minimum dari suatu layanan adalah 100 persen. Oleh karena itu, ketika Fargate mulai pensiun tugas, Amazon ECS pertama-tama menjadwalkan tugas baru dan menunggu tugas itu berjalan, sebelum menghentikan tugas lama. Tugas layanan secara rutin diganti sebagai bagian dari pensiun tugas dengan cara yang sama saat Anda menskalakan layanan, menerapkan perubahan konfigurasi, atau menerapkan revisi definisi tugas. Untuk mempersiapkan proses pensiun tugas, lihatBersiaplah untuk pensiun tugas AWS Fargate di Amazon ECS .

Bisakah Amazon ECS secara otomatis menangani tugas mandiri?

Tidak. AWS tidak dapat membuat tugas pengganti untuk tugas mandiri yang dimulai oleh`RunTask`, tugas terjadwal (misalnya melalui EventBridge Scheduler) AWS Batch, atau. AWS Step Functions Amazon ECS hanya mengelola tugas yang merupakan bagian dari layanan.

Bersiaplah untuk pensiun tugas AWS Fargate di Amazon ECS

Untuk mempersiapkan pensiun tugas, lakukan operasi berikut:

1. Atur masa tunggu pensiun tugas.

2. Tangkap pemberitahuan pensiun tugas untuk memberi tahu anggota tim.
3. Anda tidak dapat mengontrol waktu yang tepat dari pensiun tugas, namun, Anda dapat mengontrol penggantian tugas dengan memperbarui layanan dengan opsi penerapan paksa.

Langkah 1: Atur waktu tunggu tugas

Anda dapat mengkonfigurasi waktu Fargate memulai tugas pensiun. Untuk beban kerja yang memerlukan aplikasi pembaruan segera, pilih pengaturan langsung (`0`). Ketika Anda membutuhkan kontrol lebih, misalnya, ketika tugas hanya dapat dihentikan selama jendela tertentu, konfigurasikan opsi 7 hari (7), atau 14 hari (14).

Kami menyarankan Anda memilih masa tunggu yang lebih pendek untuk mengambil revisi versi platform yang lebih baru lebih cepat.

Konfigurasikan periode tunggu dengan menjalankan `put-account-setting-default` atau `put-account-setting` sebagai pengguna root atau pengguna administratif. Gunakan `fargateTaskRetirementWaitPeriod` opsi untuk `name` dan `value` opsi yang disetel ke salah satu nilai berikut:

- `0`- AWS mengirim pemberitahuan, dan segera mulai pensiun tugas yang terpengaruh.
- 7- AWS mengirim pemberitahuan, dan menunggu 7 hari kalender sebelum mulai pensiun tugas yang terpengaruh.
- 14- AWS mengirim pemberitahuan, dan menunggu 14 hari kalender sebelum mulai pensiun tugas yang terpengaruh.

Defaultnya adalah 7 hari.

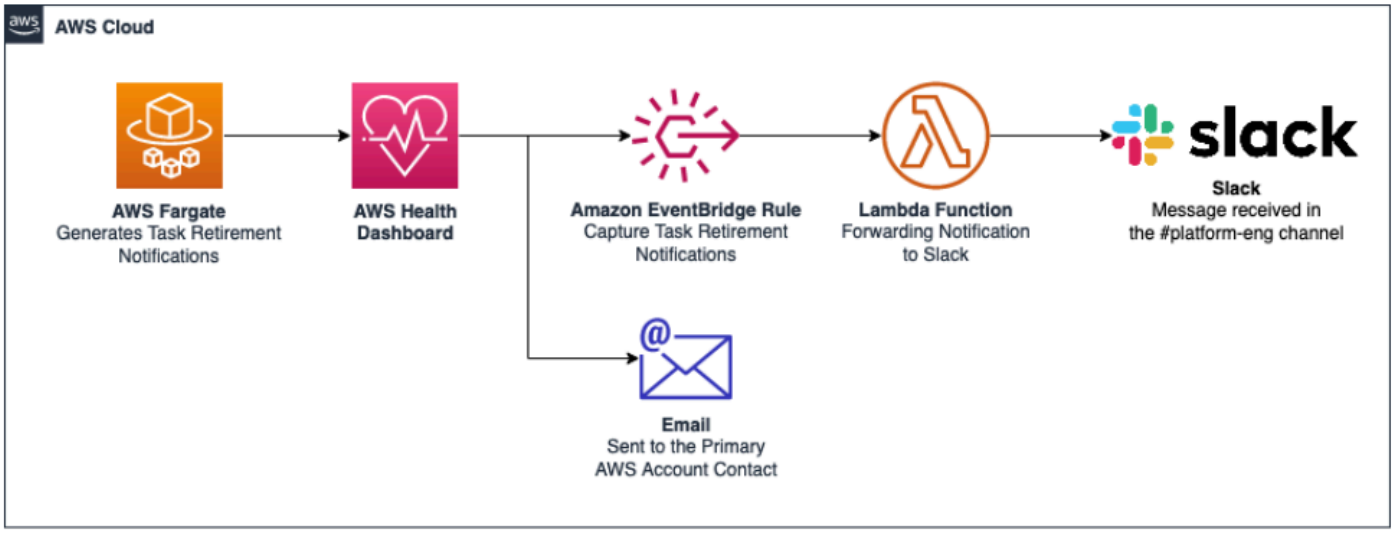
Untuk informasi selengkapnya, lihat, put-account-setting-defaultdan put-account-settingdi Referensi API Amazon Elastic Container Service.

Langkah 2: Tangkap pemberitahuan pensiun tugas untuk memperingatkan tim dan mengambil tindakan

Ketika ada pensiun tugas yang akan datang, AWS kirimkan pemberitahuan pensiun tugas ke AWS Health Dasbor, dan ke kontak email utama di Akun AWS. AWS Health Dasbor menyediakan sejumlah integrasi ke AWS layanan lain, termasuk Amazon EventBridge. Anda dapat menggunakan EventBridge untuk membangun otomatisasi dari pemberitahuan pensiun tugas, seperti meningkatkan visibilitas pensiun yang akan datang dengan meneruskan pesan ke alat. ChatOps AWS Health Aware

adalah sumber daya yang menunjukkan kekuatan AWS Health Dasbor dan bagaimana notifikasi dapat didistribusikan ke seluruh organisasi. Anda dapat meneruskan pemberitahuan pensiun tugas ke aplikasi obrolan, seperti Slack.

Ilustrasi berikut menunjukkan ikhtisar solusi.



Informasi berikut memberikan rincian.

- Fargate mengirimkan pemberitahuan pensiun tugas ke Dasbor. AWS Health
- AWS Health Dasbor mengirim email ke kontak email utama di Akun AWS, dan memberi tahu EventBridge.
- EventBridge memiliki aturan yang menangkap pemberitahuan pensiun.

  Aturan mencari acara dengan Jenis Detail Acara: `"AWS Health Event" and the Event Detail Type Code: "AWS_ECS_TASK_PATCHING_RETIREMENT"`
- Aturan tersebut memicu fungsi Lambda yang meneruskan informasi ke Slack menggunakan Slack Incoming Webhook. Untuk informasi selengkapnya, lihat Webhook Masuk.

Untuk contoh kode, lihat Menangkap Pemberitahuan Pensiun AWS Fargate Tugas di Github.

Langkah 3: Kontrol penggantian tugas

Anda tidak dapat mengontrol waktu yang tepat dari pensiun tugas, namun, Anda dapat menentukan waktu tunggu. Jika Anda ingin mengontrol penggantian tugas sesuai jadwal Anda sendiri, Anda dapat menangkap pemberitahuan pensiun tugas untuk terlebih dahulu memahami tanggal pensiun tugas.

Anda kemudian dapat menerapkan kembali layanan Anda untuk meluncurkan tugas pengganti, dan juga mengganti tugas mandiri apa pun. Untuk layanan yang menggunakan penerapan bergulir, Anda memperbarui layanan menggunakan opsi sebelum waktu mulai `update-service` pensiun. `force-deployment`

`update-service`Contoh berikut menggunakan `force-deployment` opsi.

```
aws ecs update-service —-service service_name \
    --cluster cluster_name \
     --force-new-deployment
```

Untuk layanan yang menggunakan blue/green penerapan, Anda perlu membuat penerapan baru di. AWS CodeDeployUntuk informasi tentang cara membuat penerapan, lihat create-deployment di Referensi.AWS Command Line Interface

Wilayah yang Didukung untuk Amazon ECS di Fargate AWS

Anda dapat menggunakan tabel berikut untuk memverifikasi dukungan Wilayah untuk kontainer Linux di AWS Fargate dan kontainer Windows di AWS Fargate.

Kontainer Linux di AWS Fargate

Kontainer Amazon ECS Linux aktif AWS Fargate didukung dalam hal berikut Wilayah AWS ini. Availability Zone IDs yang didukung dicatat bila berlaku.

Nama Wilayah	Wilayah
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
AS Barat (California Utara)	us-west-1 (& hanya`usw1-az1`) `usw1-az3`
AS Barat (Oregon)	us-west-2
Kanada (Pusat)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Meksiko (Tengah)	mx-pusat-1

Nama Wilayah	Wilayah
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Tokyo)	`apne1-az1` ap-northeast-1 (,, & hanya) `apne1-az2` `apne1-az4`
Asia Pasifik (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pasifik (Singapura)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Thailand)	ap-tenggara 7
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pacific (Melbourne)	ap-southeast-4
Asia Pasifik (Malaysia)	ap-southeast-5
Kanada (Pusat)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1 (& hanya) `cnn1-az1` `cnn1-az2`
Tiongkok (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1

Nama Wilayah	Wilayah
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1
Eropa (Spanyol)	eu-south-2
Eropa (Stockholm)	eu-north-1
South America (Sao Paulo)	sa-east-1
Israel (Tel Aviv)	il-central-1
Timur Tengah (Bahrain)	me-south-1
Timur Tengah (UEA)	me-central-1
AWS GovCloud (AS-Timur)	us-gov-east-1
AWS GovCloud (AS-Barat)	us-gov-west-1

Wadah Windows di AWS Fargate

Amazon ECS Windows container AWS Fargate aktif didukung sebagai berikut Wilayah AWS. Availability Zone IDs yang didukung dicatat bila berlaku.

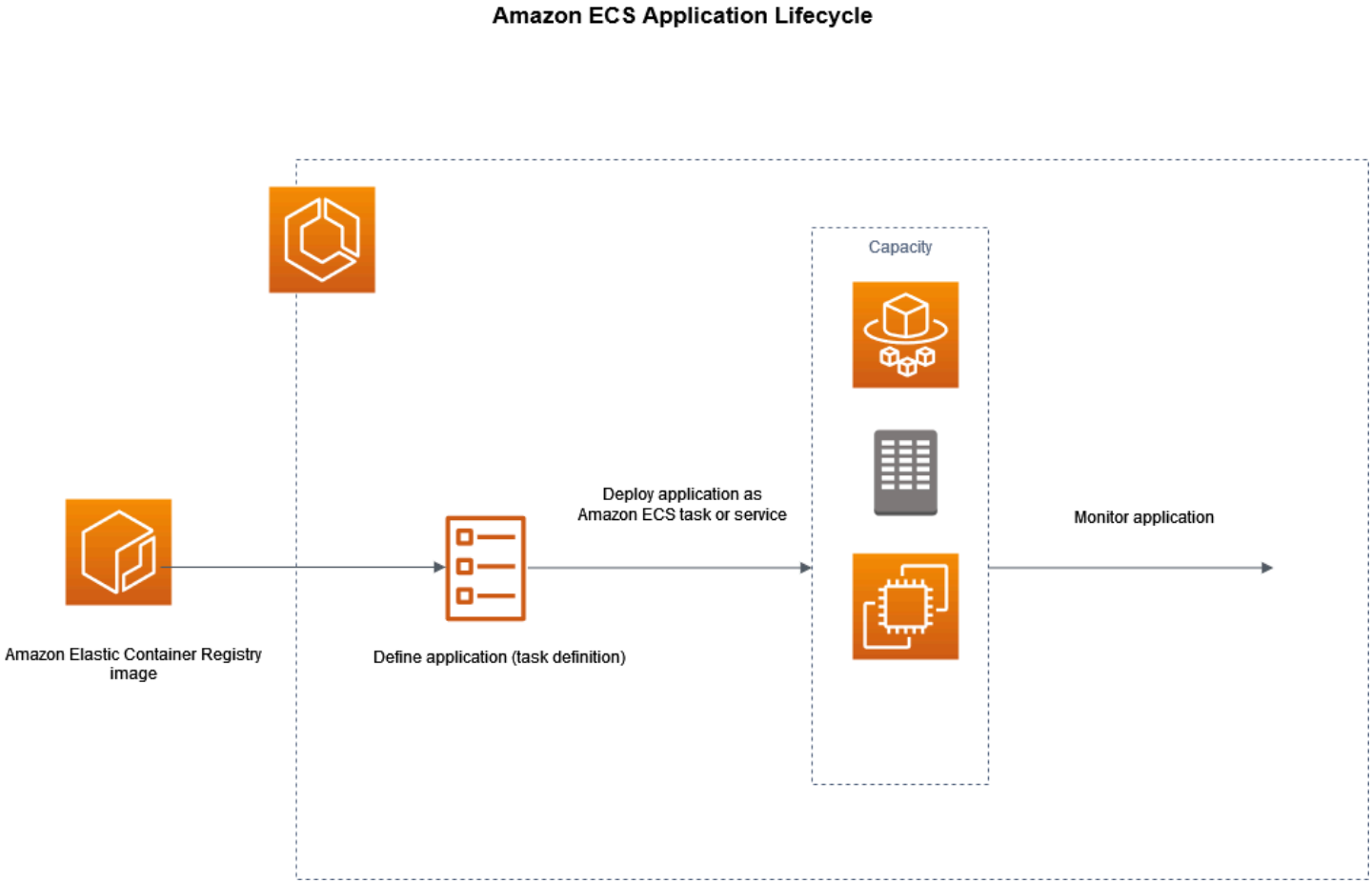
Nama Wilayah	Wilayah
US East (Ohio)	us-east-2
AS Timur (Virginia Utara)	us-east-1 `use1-az1` (,,`use1-az2`,, `use1-az4` & hanya) `use1-az5` `use1-az6`
AS Barat (California Utara)	us-west-1 (& hanya`usw1-az1`) `usw1-az3`
AS Barat (Oregon)	as-barat-2

Nama Wilayah	Wilayah
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Melbourne)	ap-southeast-4
Asia Pasifik (Malaysia)	ap-southeast-5
Asia Pasifik (Tokyo)	`apne1-az1` ap-northeast-1 (,, & hanya) `apne1-az2` `apne1-az4`
Kanada (Pusat)	ca-central-1 (& hanya) `cac1-az1` `cac1-az2`
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1 (& hanya) `cnn1-az1` `cnn1-az2`
Tiongkok (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3

Nama Wilayah	Wilayah
Eropa (Milan)	eu-south-1
Eropa (Spanyol)	eu-south-2
Eropa (Stockholm)	eu-north-1
South America (Sao Paulo)	sa-east-1
Israel (Tel Aviv)	il-central-1
Timur Tengah (UEA)	me-central-1
Timur Tengah (Bahrain)	me-south-1

Arsitek solusi Anda untuk Amazon ECS

Diagram berikut menunjukkan siklus hidup aplikasi dan cara kerjanya dengan komponen Amazon ECS.



Anda harus merancang aplikasi Anda sehingga mereka dapat berjalan pada kontainer. Container adalah unit standar pengembangan perangkat lunak yang menyimpan semua yang dibutuhkan aplikasi perangkat lunak Anda untuk dijalankan. Ini termasuk kode yang relevan, runtime, alat sistem, dan pustaka sistem. Container dibuat dari template read-only yang disebut gambar. Gambar biasanya dibuat dari Dockerfile. Dockerfile adalah file plaintext yang berisi instruksi untuk membangun wadah. Setelah dibuat, gambar-gambar ini disimpan dalam registri seperti Amazon ECR tempat mereka dapat diunduh.

Setelah Anda membuat dan menyimpan gambar Anda, Anda membuat definisi tugas Amazon ECS. Definisi tugas adalah cetak biru untuk aplikasi Anda. Ini adalah file teks dalam format JSON yang menjelaskan parameter dan satu atau lebih wadah yang membentuk aplikasi Anda. Misalnya, Anda dapat menggunakannya untuk menentukan gambar dan parameter untuk sistem operasi, wadah

mana yang akan digunakan, port mana yang akan dibuka untuk aplikasi Anda, dan volume data apa yang akan digunakan dengan wadah dalam tugas. Parameter khusus telah tersedia untuk ketentuan tugas Anda tergantung pada kebutuhan aplikasi spesifik milik Anda.

Setelah menentukan definisi tugas, Anda menerapkannya sebagai layanan atau tugas di klaster. Cluster adalah pengelompokan logis tugas atau layanan yang berjalan pada infrastruktur kapasitas yang terdaftar ke cluster.

tugas adalah penunjukan hal abstrak dengan instans yang konkret dari ketentuan tugas dalam sebuah klaster. Anda dapat menjalankan tugas mandiri, atau Anda dapat menjalankan tugas sebagai bagian dari layanan. Anda dapat menggunakan layanan Amazon ECS untuk menjalankan dan mempertahankan jumlah tugas yang Anda inginkan secara bersamaan di kluster Amazon ECS. Cara kerjanya adalah, jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan instance lain berdasarkan definisi tugas Anda. Ia melakukan ini untuk menggantikannya dan dengan demikian mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Agen kontainer berjalan pada setiap instance kontainer dalam klaster Amazon ECS. Agen mengirimkan informasi tentang tugas yang sedang berjalan dan pemanfaatan sumber daya kontainer Anda ke Amazon ECS. Ini memulai dan menghentikan tugas setiap kali menerima permintaan dari Amazon ECS.

Setelah menerapkan tugas atau layanan, Anda dapat menggunakan salah satu alat berikut untuk memantau penerapan dan aplikasi Anda:

- CloudWatch
- Pemantauan Runtime

Kapasitas

Kapasitasnya adalah infrastruktur tempat kontainer Anda berjalan. Berikut ini adalah opsinya:

- EC2 Contoh Amazon
- Tanpa Server ()AWS Fargate
- Mesin virtual lokal (VM) atau server

Anda menentukan infrastruktur saat membuat klaster. Anda juga menentukan jenis infrastruktur saat mendaftarkan definisi tugas. Definisi tugas mengacu pada infrastruktur sebagai “tipe peluncuran”.

Anda juga menggunakan jenis peluncuran saat menjalankan tugas mandiri atau menerapkan layanan. Untuk informasi tentang opsi jenis peluncuran, lihatJenis peluncuran Amazon ECS.

Titik akhir layanan

Titik akhir layanan adalah URL titik masuk Amazon ECS yang Anda gunakan untuk terhubung ke layanan secara terprogram menggunakan Internet Protocol versi 4 (IPv4) atau Internet Protocol versi 6 (). IPv6 Secara default, permintaan yang Anda buat untuk terhubung ke Amazon ECS secara terprogram menggunakan titik akhir layanan yang hanya mendukung permintaan. IPv4 Untuk terhubung secara terprogram dengan Amazon ECS menggunakan salah satu IPv4 atau IPv6 permintaan, Anda dapat menggunakan titik akhir dual-stack. Untuk informasi tentang menggunakan titik akhir dual-stack, lihat. Menggunakan titik akhir tumpukan ganda Amazon ECS

Jaringan

AWS sumber daya dibuat dalam subnet. Saat Anda menggunakan EC2 instance, Amazon ECS meluncurkan instance di subnet yang Anda tentukan saat membuat klaster. Tugas Anda berjalan di subnet instance. Untuk Fargate atau mesin virtual lokal, Anda menentukan subnet saat menjalankan tugas atau membuat layanan.

Tergantung pada aplikasi Anda, subnet dapat berupa subnet pribadi atau publik dan subnet dapat berada di salah satu sumber daya berikut: AWS

- Zona Ketersediaan
- Zona Lokal
- Wavelength Zones
- Wilayah AWS
- AWS Outposts

Untuk informasi selengkapnya, lihat Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones atau Layanan Kontainer Elastis Amazon aktif AWS Outposts.

Anda dapat membuat aplikasi Anda terhubung ke internet dengan menggunakan salah satu metode berikut:

- Subnet publik dengan gateway internet

Gunakan subnet publik ketika Anda memiliki aplikasi publik yang membutuhkan bandwidth dalam jumlah besar atau latensi minimal. Skenario yang berlaku termasuk streaming video dan layanan game.

- Subnet pribadi dengan gateway NAT

  Gunakan subnet pribadi saat Anda ingin melindungi wadah Anda dari akses eksternal langsung. Skenario yang berlaku termasuk sistem pemrosesan pembayaran atau wadah yang menyimpan data pengguna dan kata sandi.

- AWS PrivateLink

  Gunakan AWS PrivateLink untuk memiliki konektivitas pribadi antara VPCs, AWS layanan, dan jaringan lokal Anda tanpa mengekspos lalu lintas Anda ke internet publik.

Akses fitur

Anda dapat menggunakan pengaturan akun Amazon ECS untuk mengakses fitur-fitur berikut:

- Wawasan Kontainer

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Metrik tersebut mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan.

- `awsvpc`trunking

  Untuk jenis EC2 instance tertentu, Anda dapat memiliki antarmuka jaringan tambahan (ENIs) yang tersedia pada instance kontainer yang baru diluncurkan.

- Otorisasi penandaan

  Pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, seperti`ecsCreateCluster`. Jika tag ditentukan dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan pada `ecs:TagResource` tindakan untuk memverifikasi apakah pengguna atau peran memiliki izin untuk membuat tag.

- Kepatuhan Fargate FIPS-140

  Fargate mendukung Federal Information Processing Standard (FIPS-140) yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Ini adalah standar pemerintah Amerika Serikat dan Kanada saat ini, dan berlaku untuk sistem yang harus

mematuhi Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) atau Program Manajemen Risiko dan Otorisasi Federal (FedRAMP).

- Perubahan waktu pensiun tugas Fargate

  Anda dapat mengonfigurasi periode tunggu sebelum tugas Fargate dihentikan untuk ditambal.

- VPC tumpukan ganda

  Izinkan tugas untuk berkomunikasi IPv4, IPv6, atau keduanya.

- Format Amazon Resource Name (ARN)

  Fitur tertentu, seperti otorisasi penandaan, memerlukan format Amazon Resource Name (ARN) baru.

Untuk informasi selengkapnya, lihat Akses fitur Amazon ECS dengan pengaturan akun.

Peran IAM

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Di Amazon ECS, Anda dapat membuat peran untuk memberikan izin ke sumber daya Amazon ECS seperti kontainer atau layanan.

Beberapa fitur Amazon ECS memerlukan peran. Untuk informasi selengkapnya, lihat Peran IAM untuk Amazon ECS.

Pencatatan log

Pencatatan dan pemantauan adalah aspek penting dalam menjaga keandalan, ketersediaan, dan kinerja beban kerja Amazon ECS. Pilihan berikut tersedia:

- Amazon CloudWatch log - rute log ke Amazon CloudWatch
- FireLensuntuk Amazon ECS - rute log ke AWS layanan atau AWS Partner Network tujuan untuk penyimpanan dan analisis log. AWS Partner Network Ini adalah komunitas mitra global yang memanfaatkan program, keahlian, dan sumber daya untuk membangun, memasarkan, dan menjual penawaran pelanggan.

Jenis peluncuran Amazon ECS

Jenis peluncuran definisi tugas mendefinisikan kapasitas apa yang dapat dijalankan tugas, misalnya AWS Fargate.

Setelah Anda memilih jenis peluncuran, Amazon ECS memverifikasi bahwa parameter definisi tugas yang Anda konfigurasikan berfungsi dengan jenis peluncuran.

Tabel berikut menjelaskan jenis peluncuran yang tersedia.

Jenis peluncuran		Pelajari selengkapnya
Fargate	Fargate adalah mesin pay-as-you-go komputasi tanpa server yang memungkinkan Anda fokus membangun aplikasi tanpa mengelola server.	Jenis peluncuran Fargate untuk Amazon ECS
EC2	Anda memilih jenis instans, jumlah instance, dan mengelola kapasitas.	EC2 jenis peluncuran untuk Amazon ECS
Mesin virtual lokal (VM) atau server	Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke klaster Anda.	Jenis peluncuran eksternal (Amazon ECS Anywhere) untuk Amazon ECS

Jenis peluncuran Fargate untuk Amazon ECS

Fargate adalah mesin pay-as-you-go komputasi tanpa server yang memungkinkan Anda fokus membangun aplikasi tanpa mengelola server. Ketika Anda memilih Fargate, Anda tidak perlu mengelola infrastruktur. EC2 Yang perlu Anda lakukan adalah membangun image container Anda dan menentukan cluster mana Anda ingin menjalankan aplikasi Anda. Fargate memiliki integrasi asli dengan AWS layanan termasuk:

- Amazon VPC
- Auto Scaling
- Penyeimbang Beban Elastis
- IAM
- Secrets Manager

Anda memiliki kontrol lebih dengan Fargate daripada EC2 karena Anda memilih CPU dan memori yang tepat yang dibutuhkan aplikasi Anda. Fargate menangani penskalaan kapasitas Anda, jadi Anda tidak perlu khawatir tentang lonjakan lalu lintas. Ini berarti bahwa ada sedikit upaya operasional dengan Fargate.

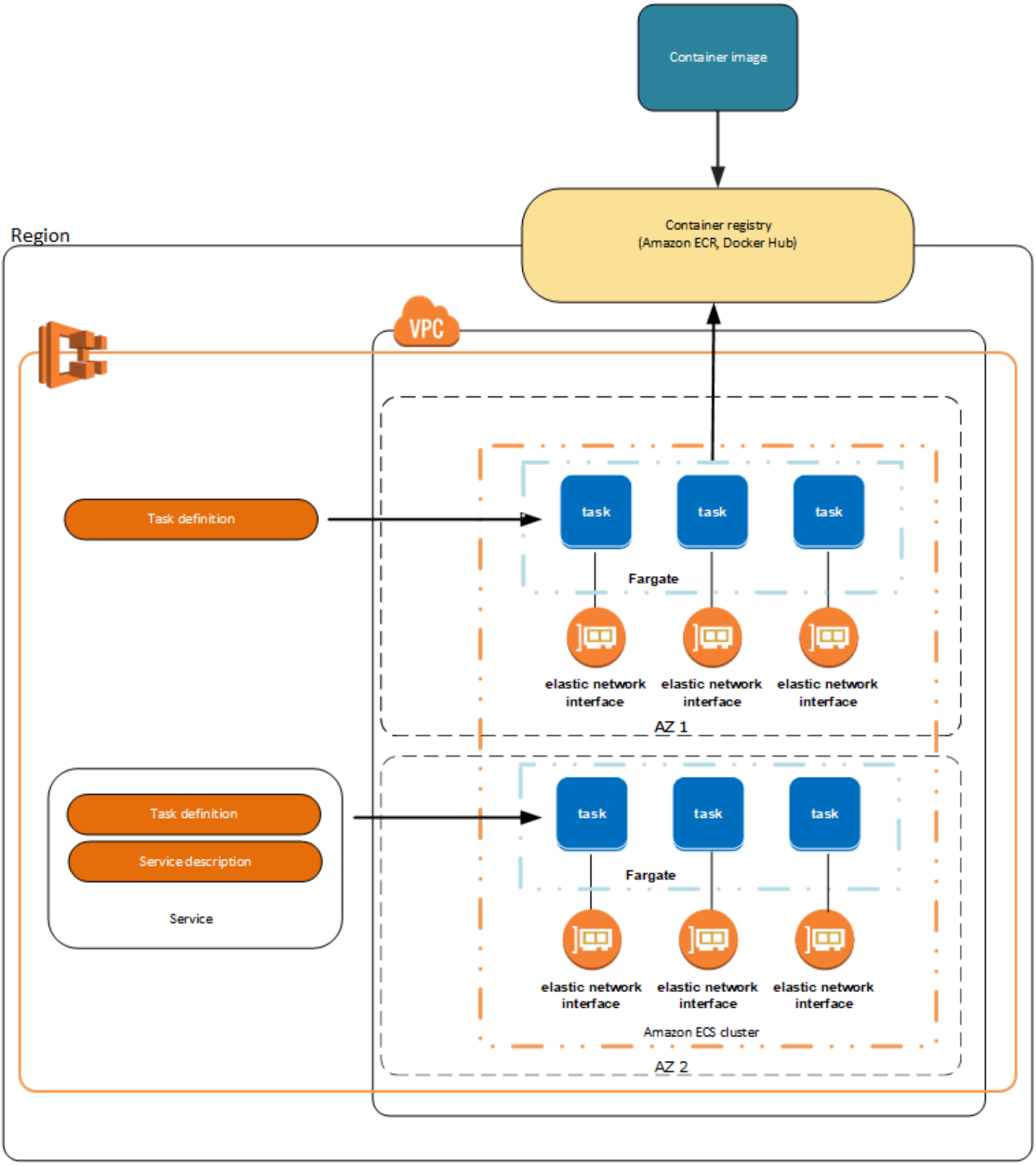
Fargate memenuhi standar untuk program kepatuhan termasuk PCI, FIPS 140-3, FedRAMP, dan HIPAA. Untuk informasi selengkapnya lihat AWS Layanan dalam Lingkup oleh Program Kepatuhan.

Fargate cocok untuk beban kerja berikut:

- Beban kerja besar yang membutuhkan overhead operasional rendah
- Beban kerja kecil yang sesekali meledak
- Beban kerja kecil
- Beban kerja batch

Untuk informasi tentang Daerah yang mendukung Fargate, lihat. the section called “AWS Wilayah Fargate”

Diagram berikut menunjukkan arsitektur umum.



Untuk informasi selengkapnya tentang Amazon ECS di Fargate, lihat. [AWS Fargateuntuk Amazon ECS]

Wadah Linux pada perilaku tarik gambar kontainer Fargate untuk Amazon ECS

Setiap tugas Fargate berjalan pada penggunaan tunggal sendiri, instance penyewa tunggal. Saat Anda menjalankan kontainer Linux di Fargate, gambar kontainer atau lapisan gambar kontainer tidak di-cache pada instance. Oleh karena itu, untuk setiap gambar kontainer yang ditentukan dalam tugas, seluruh gambar kontainer perlu ditarik dari registri gambar kontainer untuk setiap tugas Fargate. Waktu yang dibutuhkan untuk menarik gambar berkorelasi langsung dengan waktu yang dibutuhkan untuk memulai tugas Fargate.

Pertimbangkan hal berikut untuk mengoptimalkan waktu tarik gambar.

Kedekatan gambar kontainer

Untuk mengurangi waktu yang dibutuhkan untuk mengunduh gambar kontainer, cari data sedekat mungkin dengan komputasi. Menarik gambar kontainer melalui internet atau melintasi Wilayah AWS dapat memengaruhi waktu pengunduhan. Kami menyarankan Anda menyimpan gambar kontainer di Wilayah yang sama tempat tugas akan berjalan. Jika Anda menyimpan gambar kontainer di Amazon ECR, gunakan titik akhir antarmuka VPC untuk mengurangi waktu tarik gambar lebih lanjut. Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECR AWS PrivateLink() di Panduan Pengguna Amazon ECR.

Pengurangan ukuran gambar kontainer

Ukuran gambar kontainer secara langsung memengaruhi waktu pengunduhan. Mengurangi ukuran gambar kontainer atau jumlah lapisan gambar kontainer, dapat mengurangi waktu yang dibutuhkan gambar untuk diunduh. Gambar dasar yang ringan (seperti gambar kontainer Amazon Linux 2023 minimal) dapat secara signifikan lebih kecil daripada yang didasarkan pada gambar dasar sistem operasi tradisional. Untuk informasi selengkapnya tentang gambar minimal, lihat AL2023 Gambar kontainer minimal di Panduan Pengguna Amazon Linux 2023.

Algoritma kompresi alternatif

Lapisan gambar kontainer sering dikompresi saat didorong ke registri gambar kontainer. Mengompresi lapisan gambar kontainer mengurangi jumlah data yang harus ditransfer ke seluruh jaringan dan disimpan dalam registri gambar kontainer. Setelah layer image container diunduh ke instance oleh runtime container, layer tersebut didekompresi. Algoritma kompresi yang digunakan dan jumlah v yang CPUs tersedia untuk runtime memengaruhi waktu yang diperlukan untuk mendekompresi gambar kontainer. Di Fargate, Anda dapat meningkatkan ukuran tugas atau memanfaatkan algoritma kompresi zstd yang lebih berkinerja untuk mengurangi waktu yang dibutuhkan untuk dekompresi. Untuk informasi lebih lanjut, lihat zstd di. GitHub Untuk informasi

tentang cara mengimplementasikan gambar untuk Fargate, lihat Mengurangi Waktu AWS Fargate Startup dengan Gambar Kontainer Terkompresi zstd.

Lazy Loading gambar kontainer

Untuk gambar kontainer besar (> 250mb), mungkin optimal untuk malas memuat gambar kontainer daripada mengunduh semua gambar kontainer. Di Fargate, Anda dapat menggunakan Seekable OCI (SOCI) untuk malas memuat gambar kontainer dari registri gambar kontainer. Untuk informasi selengkapnya, lihat soci-snapshotter on GitHub dan Lazy loading container images menggunakan Seekable OCI (SOCI).

Wadah Windows pada perilaku tarik gambar kontainer Fargate untuk Amazon ECS

Fargate Windows cache bulan terakhir, dan bulan sebelumnya, gambar dasar servercore yang disediakan oleh Microsoft. Gambar-gambar ini cocok dengan patch KB/Build nomor yang diperbarui setiap Patch Tuesday. Misalnya, pada 9 April 2024, Microsoft merilis KB5 036896 (17763.5696) untuk Windows Server 2019. Bulan sebelumnya KB pada 12 Maret 2024 adalah KB5 035849 (17763.5576). Jadi untuk platform `WINDOWS_SERVER_2019_CORE` dan `WINDOWS_SERVER_2019_FULL` gambar kontainer berikut di-cache::

- `mcr.microsoft.com/windows/servercore:ltsc2019`
- `mcr.microsoft.com/windows/servercore:10.0.17763.5696`
- `mcr.microsoft.com/windows/servercore:10.0.17763.5576`

Selain itu, pada 9 April 2024, Microsoft merilis KB5 036909 (20348.2402) untuk Windows Server 2022. Bulan sebelumnya KB pada 12 Maret 2024 adalah KB5 035857 (20348.2340). Jadi untuk platform `WINDOWS_SERVER_2022_CORE` dan `WINDOWS_SERVER_2022_FULL` gambar kontainer berikut di-cache:

- `mcr.microsoft.com/windows/servercore:ltsc2022`
- `mcr.microsoft.com/windows/servercore:10.0.20348.2402`
- `mcr.microsoft.com/windows/servercore:10.0.20348.2340`

EC2 jenis peluncuran untuk Amazon ECS

Jenis EC2 peluncuran cocok untuk beban kerja besar yang harus dioptimalkan harga.

Saat mempertimbangkan cara memodelkan definisi tugas dan layanan menggunakan jenis EC2 peluncuran, kami sarankan Anda mempertimbangkan proses apa yang harus dijalankan bersama dan bagaimana Anda dapat melakukan penskalaan setiap komponen.

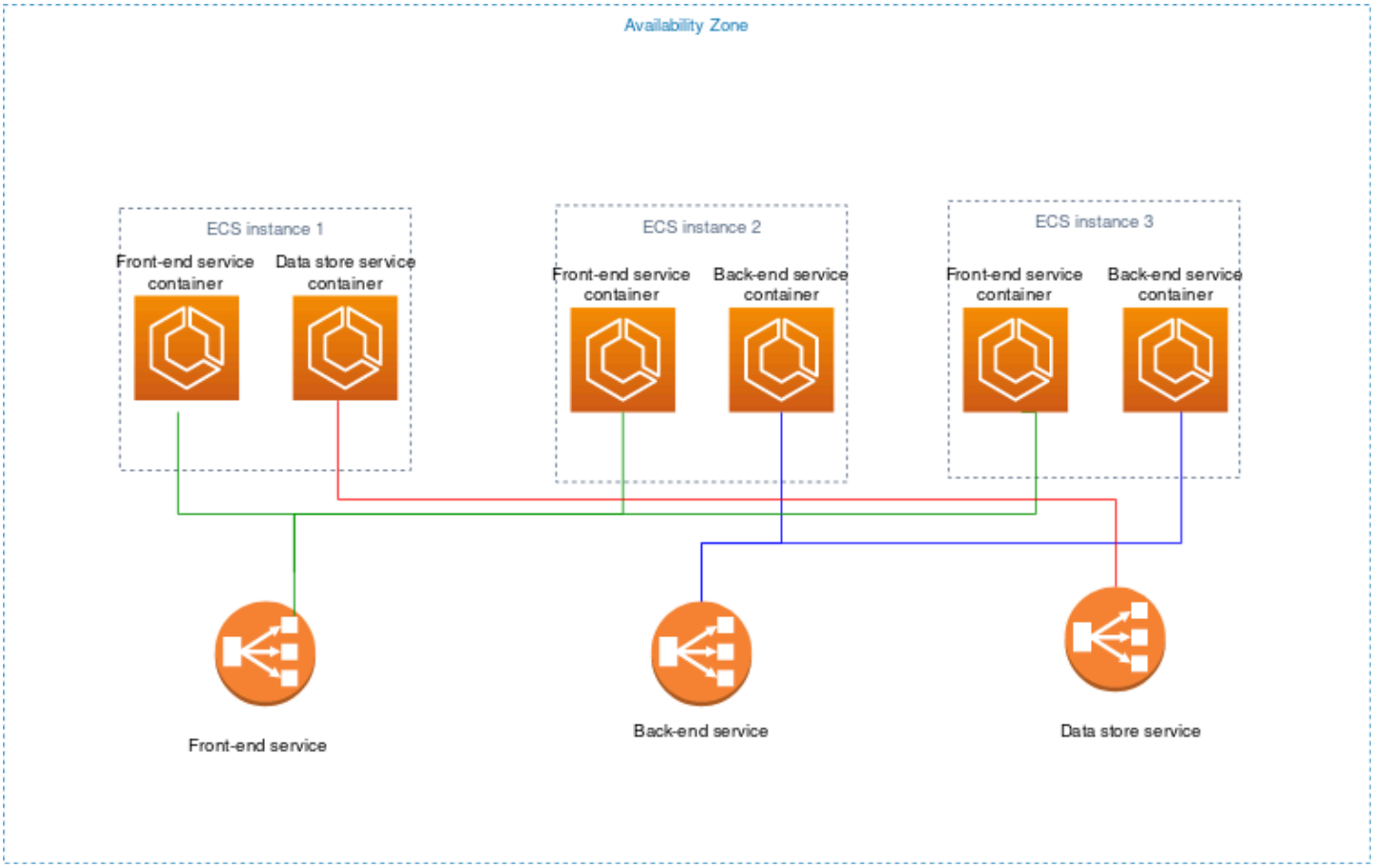
Sebagai contoh, misalkan aplikasi terdiri dari komponen-komponen berikut:

- Sebuah layanan frontend yang menampilkan informasi pada halaman web
- Layanan backend yang menyediakan layanan APIs frontend
- Penyimpanan data

Untuk contoh ini, buat definisi tugas yang mengelompokkan wadah yang digunakan untuk tujuan bersama. Pisahkan komponen yang berbeda menjadi beberapa definisi tugas yang terpisah. Contoh cluster berikut memiliki tiga instance kontainer yang menjalankan tiga kontainer layanan front-end, dua kontainer layanan backend, dan satu kontainer layanan penyimpanan data.

Anda dapat mengelompokkan kontainer terkait dalam ketentuan tugas, seperti kontainer terkait yang harus dijalankan bersama-sama. Misalnya, tambahkan wadah streaming log ke layanan front-end Anda dan sertakan dalam definisi tugas yang sama.

Setelah Anda memiliki ketentuan tugas, Anda dapat membuat layanan dari mereka untuk menjaga ketersediaan tugas yang Anda inginkan. Untuk informasi selengkapnya, lihat Membuat penyebaran pembaruan bergulir Amazon ECS. Dalam layanan Anda, Anda dapat mengaitkan kontainer dengan penyeimbang beban Elastic Load Balancing. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS. Ketika persyaratan aplikasi Anda berubah, Anda dapat memperbarui layanan Anda untuk meningkatkan atau menurunkan jumlah tugas yang diinginkan. Atau, Anda dapat memperbarui layanan untuk menerapkan versi kontainer yang lebih baru dalam tugas Anda. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.



Perilaku tarik gambar kontainer untuk jenis peluncuran EC2 dan eksternal untuk Amazon ECS

Waktu yang dibutuhkan wadah untuk memulai bervariasi, berdasarkan gambar kontainer yang mendasarinya. Misalnya, gambar yang lebih gemuk (versi lengkap Debian, Ubuntu, dan Amazon1/2) mungkin membutuhkan waktu lebih lama untuk memulai karena ada lebih banyak layanan yang berjalan di wadah dibandingkan dengan versi ramping masing-masing (Debian-slim, Ubuntu-slim, dan Amazon-slim) atau gambar dasar yang lebih kecil (Alpine).

Ketika agen Amazon ECS memulai tugas, ia menarik gambar Docker dari registri jarak jauhnya, dan kemudian menyimpan salinan lokal. Bila Anda menggunakan tag gambar baru untuk setiap rilis aplikasi Anda, perilaku ini tidak diperlukan.

Parameter `ECS_IMAGE_PULL_BEHAVIOR` agen menentukan perilaku tarik gambar. Pilihan berikut tersedia:

- `ECS_IMAGE_PULL_BEHAVIOR`: `default`

Gambar akan ditarik dari jarak jauh. Jika tarikan gagal, gambar cache dalam instance digunakan.

- `ECS_IMAGE_PULL_BEHAVIOR`: `always`

  Gambar akan ditarik dari jarak jauh. Jika tarikan gagal, tugas gagal.

Untuk mempercepat penerapan, setel parameter agen Amazon ECS ke salah satu nilai berikut:

- `ECS_IMAGE_PULL_BEHAVIOR`: `once`

  Gambar ditarik dari jarak jauh hanya jika tidak ditarik oleh tugas sebelumnya pada instance kontainer yang sama atau jika gambar yang di-cache dihapus oleh proses pembersihan gambar otomatis. Jika tidak, citra cache pada instans digunakan. Hal ini memastikan bahwa tidak ada percobaan penarikan citra yang tidak perlu.

- `ECS_IMAGE_PULL_BEHAVIOR`: `prefer-cached`

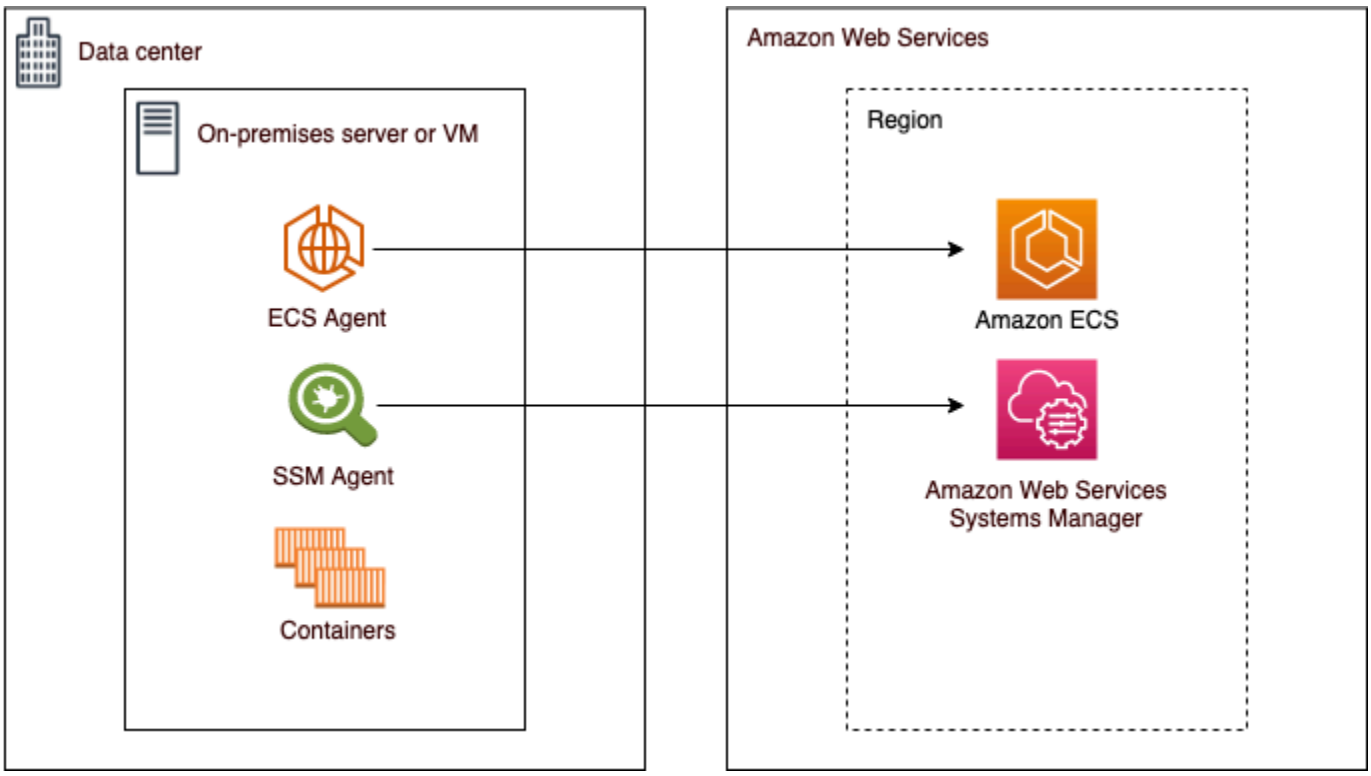
  Gambar ditarik dari jarak jauh jika tidak ada gambar yang di-cache. Jika tidak, citra cache pada instans digunakan. Pembersihan gambar otomatis dimatikan untuk wadah untuk memastikan bahwa gambar yang di-cache tidak dihapus.

Menyetel `ECS_IMAGE_PULL_BEHAVIOR` parameter ke salah satu nilai sebelumnya dapat menghemat waktu karena agen Amazon ECS menggunakan gambar yang diunduh yang ada. Untuk gambar Docker yang lebih besar, waktu pengunduhan mungkin memakan waktu 10-20 detik untuk melewati jaringan.

Jenis peluncuran eksternal (Amazon ECS Anywhere) untuk Amazon ECS

Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke kluster Amazon ECS Anda. Instans eksternal dioptimalkan guna menjalankan aplikasi yang menghasilkan lalu lintas keluar atau data proses. Jika aplikasi Anda memerlukan lalu lintas masuk, kurangnya dukungan Elastic Load Balancing membuat menjalankan beban kerja ini menjadi kurang efisien. Amazon ECS menambahkan jenis `EXTERNAL` peluncuran baru yang dapat Anda gunakan untuk membuat layanan atau menjalankan tugas pada instans eksternal Anda.

Berikut ini memberikan ikhtisar arsitektur sistem tingkat tinggi Amazon ECS Anywhere. Server lokal Anda telah menginstal agen Amazon ECS dan agen SSM.



Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk jenis peluncuran eksternal.

Menggunakan titik akhir tumpukan ganda Amazon ECS

Titik akhir dual-stack Amazon ECS mendukung permintaan ke Amazon ECS melalui Internet Protocol versi 4 (IPv4) dan Internet Protocol versi 6 (). IPv6 Untuk daftar semua titik akhir Amazon ECS, lihat titik akhir dan kuota Amazon ECS di. Referensi Umum AWS

Saat menggunakan REST API, Anda langsung mengakses titik akhir Amazon ECS dengan menggunakan nama titik akhir (URI). Amazon ECS hanya mendukung nama titik akhir dual-stack regional, yang berarti Anda harus menentukan wilayah sebagai bagian dari nama.

Gunakan konvensi penamaan berikut untuk nama titik akhir dual-stack:. `ecs.region.api.aws`

Saat menggunakan AWS Command Line Interface (AWS CLI) dan AWS SDKs, Anda dapat menggunakan parameter atau flag untuk mengubah ke titik akhir dual-stack. Anda juga dapat menentukan titik akhir dual-stack secara langsung sebagai penggantian titik akhir Amazon ECS dalam file konfigurasi.

Bagian berikut menjelaskan cara menggunakan titik akhir dual-stack dari AWS CLI, the AWS SDKs, dan REST API.

Topik

- Menggunakan titik akhir tumpukan ganda dari AWS CLI
- Menggunakan titik akhir tumpukan ganda dari AWS SDKs
- Menggunakan titik akhir tumpukan ganda dari API REST

Menggunakan titik akhir tumpukan ganda dari AWS CLI

Bagian ini memberikan contoh AWS CLI perintah yang digunakan untuk membuat permintaan ke endpoint dual-stack. Untuk informasi selengkapnya tentang menginstal AWS CLI atau memperbarui ke versi terbaru, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI dalam Panduan AWS Command Line Interface Pengguna untuk Versi 2.

Untuk menggunakan titik akhir dual-stack, Anda dapat menyetel nilai konfigurasi `use_dualstack_endpoint` ke `true` dalam `config` file agar dapat mengarahkan semua permintaan Amazon ECS yang dibuat oleh `ecs` AWS CLI perintah ke titik akhir tumpukan ganda untuk wilayah yang ditentukan. AWS CLI Anda dapat menentukan wilayah dalam `config` file atau dalam perintah dengan menggunakan `--region` opsi. Untuk informasi selengkapnya tentang file konfigurasi AWS CLI, lihat Pengaturan file konfigurasi dan kredensyal AWS CLI di Panduan AWS Command Line Interface Pengguna untuk Versi 2.

Jika Anda ingin menggunakan endpoint dual-stack untuk AWS CLI perintah tertentu, Anda dapat menggunakan salah satu metode berikut:

- Anda dapat menggunakan titik akhir dual-stack per perintah dengan mengatur `--endpoint-url` parameter ke `https://ecs.`*`aws-region`*`.api.aws` atau `http://ecs.`*`aws-region`*`.api.aws` untuk perintah apa pun. `ecs`

  Contoh perintah berikut mencantumkan semua cluster yang tersedia dan menggunakan endpoint dual-stack untuk permintaan tersebut.

  ```
  $ aws ecs list-clusters --endpoint-url https://ecs.aws-region.api.aws
  ```

- Anda dapat mengatur profil terpisah di AWS Config file Anda. Sebagai contoh, buat satu profil yang menetapkan `use_dualstack_endpoint` ke `true` dan profil yang tidak mengatur `use_dualstack_endpoint`. Ketika Anda menjalankan perintah, tentukan profil mana yang ingin Anda gunakan, tergantung pada apakah Anda ingin menggunakan titik akhir tumpukan ganda atau tidak.

Menggunakan titik akhir tumpukan ganda dari AWS SDKs

Bagian ini memberikan contoh cara mengakses endpoint dual-stack dengan menggunakan. AWS SDKs

AWS SDK for Java 2.x

Contoh berikut menunjukkan cara menentukan titik akhir dual-stack untuk `us-east-1` Wilayah menggunakan. AWS SDK for Java 2.x

```
Region region = Region.US_EAST_1
EcsClient client =
 EcsClient.builder().region(region).dualstackEnabled(true).build();
```

AWS SDK untuk Go

Contoh berikut menunjukkan cara menentukan titik akhir dual-stack untuk `us-east-1` Wilayah menggunakan. AWS SDK untuk Go

```
sess := session.Must(session.NewSession())
svc := ecs.New(sess, &aws.Config{
    Region: aws.String(endpoints.UsEast1RegionID),
    Endpoint: aws.String("https://ecs.us-east-1.api.aws")
})
```

Untuk informasi selengkapnya, lihat Dual-stack dan titik akhir FIPS di Panduan Referensi Alat AWS SDKs dan Alat.

Menggunakan titik akhir tumpukan ganda dari API REST

Saat menggunakan REST API, Anda dapat langsung mengakses titik akhir dual-stack dengan menentukannya dalam permintaan Anda. Contoh berikut menggunakan endpoint dual-stack untuk mencantumkan semua cluster Amazon ECS di Wilayah. `us-east-1`

```
POST / HTTP/1.1
Host: ecs.us-east-1.api.aws
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: AmazonEC2ContainerServiceV20141113.ListClusters
```

```
X-Amz-Date: 20150429T170621Z
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{}
```

Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones

Amazon ECS mendukung beban kerja yang menggunakan Local Zones, Wavelength Zones, dan AWS Outposts untuk saat latensi rendah atau pemrosesan data lokal merupakan persyaratan.

- Anda dapat menggunakan Local Zones sebagai perpanjangan dari sebuah Wilayah AWS untuk menempatkan sumber daya di beberapa lokasi yang lebih dekat dengan pengguna akhir Anda.
- Anda dapat menggunakan Wavelength Zones untuk membangun aplikasi yang memberikan latensi ultra-rendah ke perangkat 5G dan pengguna akhir. Wavelength menyebarkan layanan komputasi dan penyimpanan AWS standar ke tepi jaringan 5G operator telekomunikasi.
- AWS Outposts membawa model asli Layanan AWS, infrastruktur, dan operasi ke hampir semua pusat data, ruang co-lokasi, atau fasilitas lokal.

> **Important**
>
> Amazon ECS pada AWS Fargate beban kerja tidak didukung di Local Zones, Wavelength Zone, atau aktif saat ini. AWS Outposts

Untuk informasi tentang perbedaan antara Local Zones, Wavelength Zones, AWS Outposts dan, lihat Bagaimana saya harus memikirkan kapan harus menggunakan AWS Wavelength, AWS Local Zones, atau untuk aplikasi yang membutuhkan latensi rendah atau pemrosesan data lokal di. AWS Outposts AWS Wavelength FAQs

Subnet bersama

Anda dapat menggunakan berbagi VPC untuk berbagi subnet dengan AWS akun lain dalam hal yang sama. AWS Organizations

Anda dapat menggunakan shared VPCs untuk jenis EC2 peluncuran dengan pertimbangan berikut:

- Pemilik subnet VPC harus berbagi subnet dengan akun peserta sebelum akun tersebut dapat menggunakannya untuk sumber daya Amazon ECS.
- Anda tidak dapat menggunakan grup keamanan default VPC untuk instance container Anda karena milik pemiliknya. Selain itu, peserta tidak dapat meluncurkan instance menggunakan grup keamanan yang dimiliki oleh peserta lain atau pemilik.
- Dalam subnet bersama, peserta dan pemilik secara terpisah mengontrol grup keamanan dalam setiap akun masing-masing. Pemilik subnet dapat melihat grup keamanan yang dibuat oleh peserta tetapi tidak dapat melakukan tindakan apa pun pada mereka. Jika pemilik subnet ingin menghapus atau memodifikasi grup keamanan ini, peserta yang membuat grup keamanan harus mengambil tindakan.
- Pemilik VPC bersama tidak dapat melihat, memperbarui, atau menghapus klaster yang dibuat peserta di subnet bersama. Ini merupakan tambahan dari sumber daya VPC yang setiap akun memiliki akses berbeda. Untuk informasi selengkapnya, lihat Tanggung jawab dan izin untuk pemilik dan peserta di Panduan Pengguna Amazon VPC.

Anda dapat menggunakan shared VPCs untuk tipe peluncuran Fargate dengan pertimbangan berikut:

- Pemilik subnet VPC harus berbagi subnet dengan akun peserta sebelum akun tersebut dapat menggunakannya untuk sumber daya Amazon ECS.
- Anda tidak dapat membuat layanan atau menjalankan tugas menggunakan grup keamanan default untuk VPC karena milik pemilik. Selain itu, peserta tidak dapat membuat layanan atau menjalankan tugas menggunakan grup keamanan yang dimiliki oleh peserta lain atau pemilik.
- Dalam subnet bersama, peserta dan pemilik secara terpisah mengontrol grup keamanan dalam setiap akun masing-masing. Pemilik subnet dapat melihat grup keamanan yang dibuat oleh peserta tetapi tidak dapat melakukan tindakan apa pun pada mereka. Jika pemilik subnet ingin menghapus atau memodifikasi grup keamanan ini, peserta yang membuat grup keamanan harus mengambil tindakan.
- Pemilik VPC bersama tidak dapat melihat, memperbarui, atau menghapus klaster yang dibuat peserta di subnet bersama. Ini merupakan tambahan dari sumber daya VPC yang setiap akun memiliki akses berbeda. Untuk informasi selengkapnya, lihat Tanggung jawab dan izin untuk pemilik dan peserta di Panduan Pengguna Amazon VPC.

Untuk informasi selengkapnya tentang berbagi subnet VPC, lihat Bagikan VPC Anda dengan akun lain di Panduan Pengguna Amazon VPC.

Zona Lokal

Zona Lokal adalah perpanjangan Wilayah AWS dari jarak geografis yang dekat dengan pengguna Anda. Zona Lokal memiliki koneksinya sendiri ke internet dan mendukung AWS Direct Connect. Sumber daya yang dibuat di Zona Lokal dapat melayani pengguna lokal dengan komunikasi latensi rendah. Untuk informasi selengkapnya, lihat Zona Lokal AWS.

Zona Lokal diwakili oleh kode Wilayah diikuti oleh pengidentifikasi yang menunjukkan lokasi (misalnya,`us-west-2-lax-1a`).

Untuk menggunakan Zona Lokal, Anda harus ikut serta dalam zona tersebut. Setelah Anda ikut serta, Anda harus membuat VPC Amazon dan subnet di Zona Lokal.

Anda dapat meluncurkan EC2 instans Amazon, server FSx file Amazon, dan Application Load Balancer untuk digunakan untuk klaster dan tugas Amazon ECS Anda.

Untuk informasi selengkapnya, lihat Apa itu AWS Local Zones? di Panduan Pengguna AWS Local Zones.

Wavelength Zones

Anda dapat menggunakannya AWS Wavelengthuntuk membangun aplikasi yang memberikan latensi sangat rendah ke perangkat seluler dan pengguna akhir. Wavelength menyebarkan layanan komputasi dan penyimpanan AWS standar ke tepi jaringan 5G operator telekomunikasi. Anda dapat memperluas Amazon Virtual Private Cloud ke satu atau beberapa Wavelength Zone. Kemudian, Anda dapat menggunakan AWS sumber daya seperti EC2 instans Amazon untuk menjalankan aplikasi yang memerlukan latensi sangat rendah dan koneksi ke Layanan AWS di Wilayah.

Wavelength Zone adalah Zona terisolasi di lokasi pembawa tempat infrastruktur Wavelength dikerahkan. Zona Wavelength terikat pada sebuah. Wilayah AWS Zona Panjang Gelombang adalah perpanjangan logis dari Wilayah, dan dikelola oleh bidang kontrol di Wilayah.

Zona Wavelength diwakili oleh kode Wilayah diikuti oleh pengidentifikasi yang menunjukkan Zona Wavelength (misalnya,). `us-east-1-wl1-bos-wlz-1`

Untuk menggunakan Wavelength Zone, Anda harus ikut serta dalam Zona. Setelah Anda ikut serta, Anda harus membuat Amazon VPC dan subnet di Wavelength Zone. Kemudian, Anda dapat meluncurkan EC2 instans Amazon di Zona yang akan digunakan untuk klaster dan tugas Amazon ECS Anda.

Untuk informasi selengkapnya, lihat Memulai dengan AWS Wavelength di Panduan Developer AWS Wavelength .

Wavelength Zones tidak tersedia di semua. Wilayah AWS Untuk informasi tentang Wilayah yang mendukung Wavelength Zones, lihat Wavelength Zones yang Tersedia di Panduan Developer AWS Wavelength .

Layanan Kontainer Elastis Amazon aktif AWS Outposts

AWS Outposts memungkinkan AWS layanan asli, infrastruktur, dan model operasi di fasilitas lokal. Di AWS Outposts lingkungan, Anda dapat menggunakan alat AWS APIs, dan infrastruktur yang sama yang Anda gunakan di AWS Cloud.

Amazon ECS aktif AWS Outposts sangat ideal untuk beban kerja latensi rendah yang perlu dijalankan di dekat data dan aplikasi lokal.

Untuk informasi selengkapnya AWS Outposts, lihat Panduan AWS Outposts Pengguna.

Pertimbangan

Berikut ini adalah pertimbangan penggunaan Amazon ECS pada: AWS Outposts

- Amazon Elastic Container Registry AWS Identity and Access Management,, dan Network Load Balancer berjalan di AWS Wilayah, bukan di. AWS Outposts Hal ini akan meningkatkan latensi antara layanan ini dan kontainer.
- AWS Fargate tidak tersedia di AWS Outposts.

Berikut ini adalah pertimbangan konektivitas jaringan untuk AWS Outposts:

- Jika konektivitas jaringan antara Anda AWS Outposts dan AWS Wilayahnya hilang, cluster Anda akan terus berjalan. Namun, Anda tidak dapat membuat klaster baru atau mengambil tindakan baru pada klaster yang ada sampai konektivitas dipulihkan. Dalam kasus kegagalan instans, instans tidak akan diganti secara otomatis. Agen CloudWatch Log tidak akan dapat memperbarui log dan data peristiwa.
- Kami menyarankan Anda menyediakan konektivitas yang andal, sangat tersedia, dan latensi rendah antara Anda AWS Outposts dan AWS Wilayahnya.

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Amazon ECS pada: AWS Outposts

- Anda harus menginstal dan mengonfigurasi Outpost di pusat data On-Premise Anda.
- Anda harus memiliki koneksi jaringan yang dapat diandalkan antara Outpost Anda dan Wilayah AWS .

Ikhtisar pembuatan cluster pada AWS Outposts

Berikut ini adalah ikhtisar konfigurasi:

1. Buat peran dan kebijakan dengan hak pada AWS Outposts.
2. Buat profil instans IAM dengan hak pada AWS Outposts.
3. Buat VPC, atau gunakan yang sudah ada di Wilayah yang sama dengan Anda. AWS Outposts
4. Buat subnet atau gunakan yang sudah ada yang terkait dengan. AWS Outposts

   Ini adalah subnet tempat instance kontainer berjalan.
5. Buat grup keamanan untuk instance kontainer di klaster Anda.
6. Buat cluster Amazon ECS.
7. Tentukan variabel lingkungan agen kontainer Amazon ECS untuk meluncurkan instance ke dalam cluster.
8. Jalankan wadah.

Untuk informasi terperinci tentang cara mengintegrasikan Amazon ECS AWS Outposts, lihat Memperluas Amazon ECS di dua AWS Outposts rak.

Contoh berikut membuat cluster Amazon ECS pada file AWS Outposts.

1. Buat peran dan kebijakan dengan hak pada AWS Outposts.

   `role-policy.json`File adalah dokumen kebijakan yang berisi efek dan tindakan untuk sumber daya. Untuk informasi tentang format file, lihat PutRolePolicydi Referensi API IAM

   ```
   aws iam create-role --role-name ecsRole \
       --assume-role-policy-document file://ecs-policy.json
   ```

```
aws iam put-role-policy --role-name ecsRole --policy-name ecsRolePolicy \
    --policy-document file://role-policy.json
```

2. Buat profil instans IAM dengan hak pada AWS Outposts.

```
aws iam create-instance-profile --instance-profile-name outpost
aws iam add-role-to-instance-profile --instance-profile-name outpost \
    --role-name ecsRole
```

3. Buat sebuah VPC.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

4. Buat subnet yang terkait dengan Anda AWS Outposts.

```
aws ec2 create-subnet \
    --cidr-block 10.0.3.0/24 \
    --vpc-id vpc-xxxxxxxx \
    --outpost-arn arn:aws:outposts:us-west-2:123456789012:outpost/op-
xxxxxxxxxxxxxxxx \
    --availability-zone-id usw2-az1
```

5. Membuat grup keamanan untuk instans kontainer, menentukan kisaran CIDR yang tepat untuk AWS Outposts. (Langkah ini berbeda untuk AWS Outposts.)

```
aws ec2 create-security-group --group-name MyOutpostSG
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
    --port 22 --cidr 10.0.3.0/24
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
    --port 80 --cidr 10.0.3.0/24
```

6. Buat Klaster.

7. Tentukan variabel lingkungan agen penampung Amazon ECS untuk meluncurkan instance ke dalam klaster yang dibuat pada langkah sebelumnya dan tentukan tag apa pun yang ingin Anda tambahkan untuk membantu mengidentifikasi klaster (misalnya, `Outpost` untuk menunjukkan bahwa klaster tersebut untuk Outpost).

```
#! /bin/bash
cat << 'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_IMAGE_PULL_BEHAVIOR=prefer-cached
ECS_CONTAINER_INSTANCE_TAGS={"environment": "Outpost"}
```

```
EOF
```

> **Note**
>
> Untuk menghindari penundaan yang disebabkan oleh penarikan gambar kontainer dari Amazon ECR di Wilayah, gunakan cache gambar. Untuk melakukan ini, setiap kali tugas dijalankan, konfigurasikan agen Amazon ECS ke default untuk menggunakan gambar yang di-cache pada instance itu sendiri dengan menyetel `ECS_IMAGE_PULL_BEHAVIOR` ke. `prefer-cached`

8. Membuat instans kontainer, menentukan VPC dan subnet untuk AWS Outposts tempat instans ini harus dijalankan dan tipe instans yang tersedia di AWS Outposts. (Langkah ini berbeda untuk AWS Outposts.)

   `userdata.txt`File berisi data pengguna yang dapat digunakan instance untuk melakukan tugas konfigurasi otomatis umum dan bahkan menjalankan skrip setelah instance dimulai. Untuk informasi tentang file untuk panggilan API, lihat Menjalankan perintah pada instance Linux Anda saat diluncurkan di Panduan EC2 Pengguna Amazon.

   ```
   aws ec2 run-instances --count 1 --image-id ami-xxxxxxxx --instance-type c5.large \
       --key-name aws-outpost-key --subnet-id subnet-xxxxxxxxxxxxxxxxx \
       --iam-instance-profile Name outpost --security-group-id sg-xxxxxx \
       --associate-public-ip-address --user-data file://userdata.txt
   ```

   > **Note**
   >
   > Perintah ini juga digunakan saat menambahkan instans tambahan ke klaster. Setiap kontainer di-deploy di klaster yang akan ditempatkan pada AWS Outposts yang spesifik.

Praktik terbaik jaringan Amazon ECS

Aplikasi modern biasanya dibangun dari beberapa komponen terdistribusi yang berkomunikasi satu sama lain. Misalnya, aplikasi seluler atau web mungkin berkomunikasi dengan titik akhir API, dan API mungkin didukung oleh beberapa layanan mikro yang berkomunikasi melalui internet.

Untuk informasi tentang praktik terbaik untuk aplikasi koneksi ke internet, lihatHubungkan aplikasi Amazon ECS ke internet.

Untuk informasi tentang praktik terbaik untuk menerima koneksi masuk ke Amazon ECS dari internet, lihat. Praktik terbaik untuk menerima koneksi masuk ke Amazon ECS dari internet

Untuk informasi tentang praktik terbaik untuk menghubungkan Amazon ECS ke AWS layanan lain, lihatPraktik terbaik untuk menghubungkan Amazon ECS ke AWS layanan dari dalam VPC Anda.

Untuk informasi tentang praktik terbaik untuk menghubungkan layanan dalam VPC, lihat. Praktik terbaik untuk menghubungkan layanan Amazon ECS di VPC

Untuk informasi tentang praktik terbaik untuk layanan jaringan di seluruh Akun AWS dan VPCs, lihatPraktik terbaik untuk jaringan layanan Amazon ECS di seluruh Akun AWS dan VPCs.

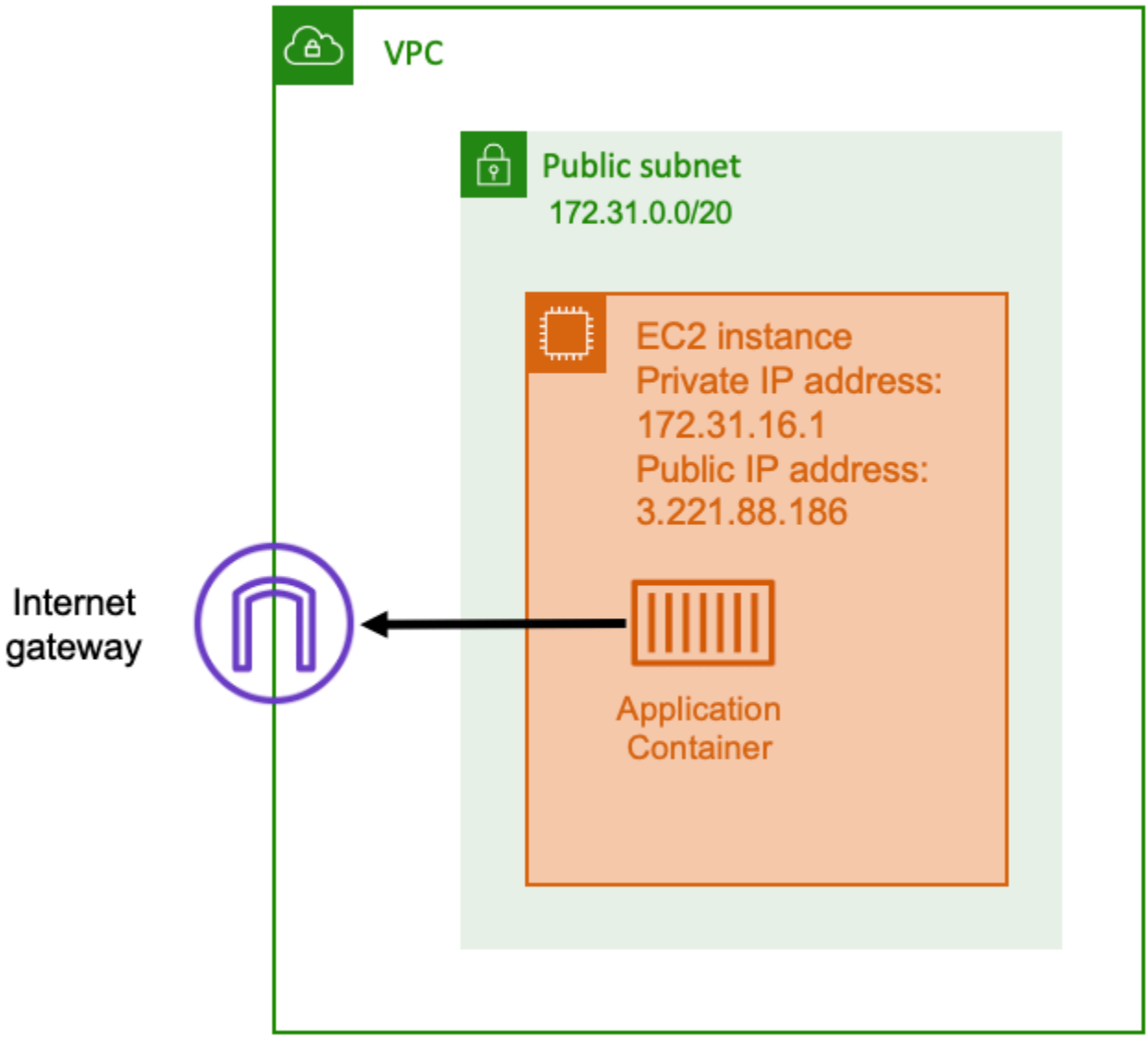
Untuk informasi tentang praktik terbaik layanan untuk mengatasi masalah jaringan, lihat. AWS layanan untuk pemecahan masalah jaringan Amazon ECS

Hubungkan aplikasi Amazon ECS ke internet

Sebagian besar aplikasi kontainer memiliki setidaknya beberapa komponen yang memerlukan akses keluar ke internet. Misalnya, backend untuk aplikasi seluler memerlukan akses keluar ke pemberitahuan push.

Amazon Virtual Private Cloud memiliki dua metode utama untuk memfasilitasi komunikasi antara VPC Anda dan internet.

Subnet publik dan gateway internet



Bila Anda menggunakan subnet publik yang memiliki rute ke gateway internet, aplikasi containerized Anda dapat berjalan pada host di dalam VPC pada subnet publik. Host yang menjalankan container Anda diberi alamat IP publik. Alamat IP publik ini dapat dirutekan dari internet. Untuk informasi lebih lanjut, lihat Gateway internet di Panduan Pengguna Amazon VPC.

Arsitektur jaringan ini memfasilitasi komunikasi langsung antara host yang menjalankan aplikasi Anda dan host lain di internet. Komunikasi bersifat bi-directional. Ini berarti bahwa Anda tidak hanya dapat membuat koneksi keluar ke host lain di internet, tetapi host lain di internet mungkin juga mencoba untuk terhubung ke host Anda. Karena itu, Anda harus memperhatikan grup keamanan dan aturan

firewall Anda. Ini memastikan bahwa host lain di internet tidak dapat membuka koneksi apa pun yang tidak ingin Anda buka.

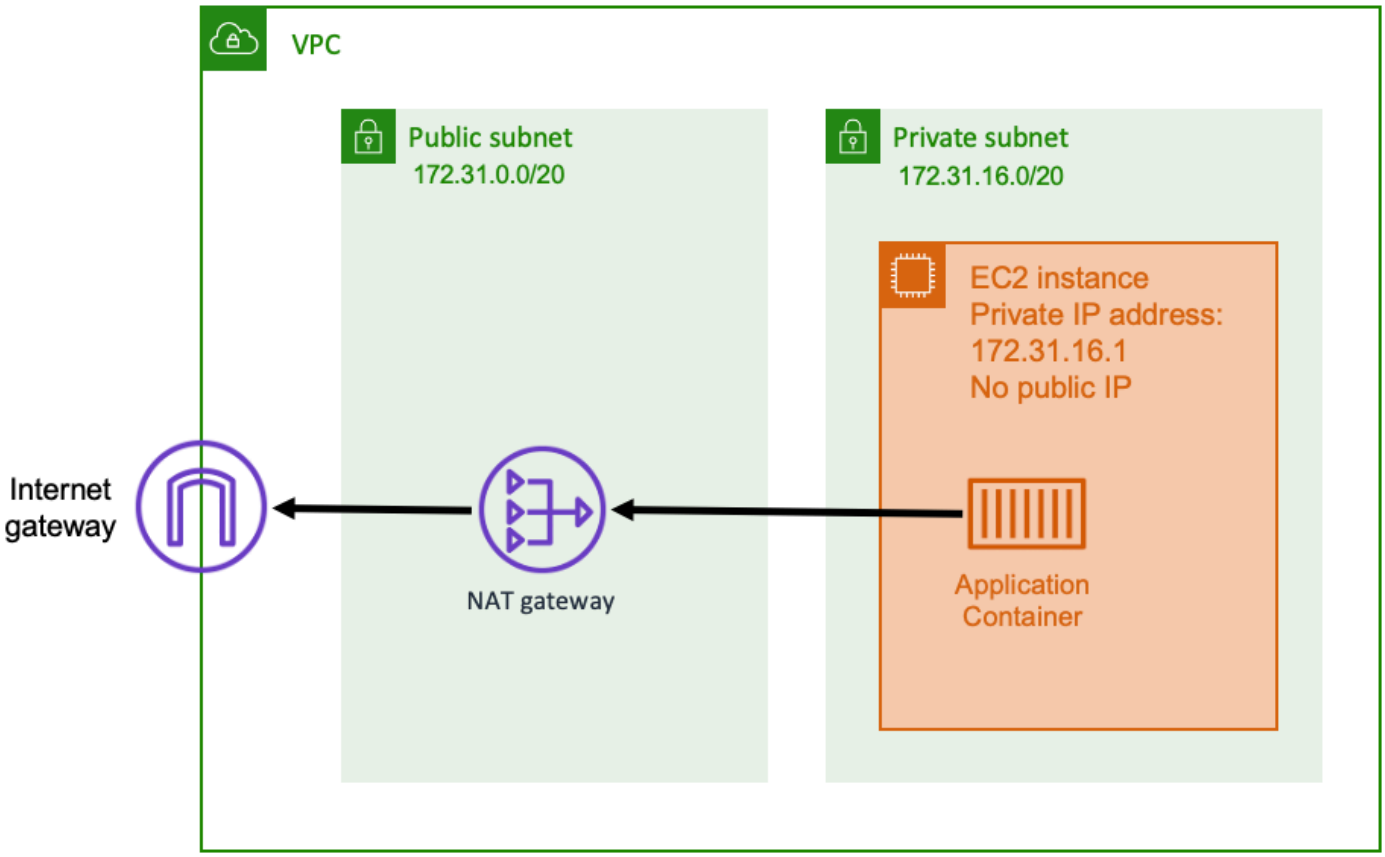
Misalnya, jika aplikasi Anda berjalan di Amazon EC2, pastikan port 22 untuk akses SSH tidak terbuka. Jika tidak, instans Anda dapat menerima upaya koneksi SSH konstan dari bot berbahaya di internet. Bot ini menjaring melalui alamat IP publik. Setelah mereka menemukan port SSH terbuka, mereka mencoba untuk memaksa kata sandi untuk mencoba mengakses instans Anda. Karena itu, banyak organisasi membatasi penggunaan subnet publik dan lebih memilih untuk memiliki sebagian besar, jika tidak semua, sumber daya mereka di dalam subnet pribadi.

Menggunakan subnet publik untuk jaringan cocok untuk aplikasi publik yang membutuhkan bandwidth dalam jumlah besar atau latensi minimal. Kasus penggunaan yang berlaku termasuk streaming video dan layanan game.

Pendekatan jaringan ini didukung baik saat Anda menggunakan Amazon ECS di Amazon EC2 dan saat Anda menggunakannya. AWS Fargate

- Amazon EC2 — Anda dapat meluncurkan EC2 instance di subnet publik. Amazon ECS menggunakan EC2 instans ini sebagai kapasitas cluster, dan setiap kontainer yang berjalan pada instans dapat menggunakan alamat IP publik yang mendasari host untuk jaringan keluar. Ini berlaku untuk mode `host` dan `bridge` jaringan. Namun, mode `awsvpc` jaringan tidak menyediakan tugas ENIs dengan alamat IP publik. Oleh karena itu, mereka tidak dapat menggunakan gateway internet secara langsung.
- Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet publik untuk konfigurasi jaringan layanan Anda, dan gunakan opsi Tetapkan alamat IP publik. Setiap tugas Fargate berjejaring di subnet publik, dan memiliki alamat IP publik sendiri untuk komunikasi langsung dengan internet.

Subnet pribadi dan gateway NAT



Saat Anda menggunakan subnet pribadi dan gateway NAT, Anda dapat menjalankan aplikasi kontainer Anda di host yang ada di subnet pribadi. Dengan demikian, host ini memiliki alamat IP pribadi yang dapat dirutekan di dalam VPC Anda, tetapi tidak dapat dirutekan dari internet. Ini berarti bahwa host lain di dalam VPC dapat terhubung ke host menggunakan alamat IP pribadinya, tetapi host lain di internet tidak dapat melakukan komunikasi masuk ke host.

Dengan subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT) untuk memungkinkan host di dalam subnet pribadi terhubung ke internet. Host di internet menerima koneksi masuk yang tampaknya berasal dari alamat IP publik gateway NAT yang ada di dalam subnet publik. Gateway NAT bertanggung jawab untuk berfungsi sebagai jembatan antara internet dan subnet pribadi. Konfigurasi ini sering disukai untuk alasan keamanan karena itu berarti bahwa VPC Anda dilindungi dari akses langsung oleh penyerang di internet. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC.

Pendekatan jaringan pribadi ini cocok untuk skenario di mana Anda ingin melindungi wadah Anda dari akses eksternal langsung. Skenario yang berlaku termasuk sistem pemrosesan pembayaran

atau wadah yang menyimpan data pengguna dan kata sandi. Anda dikenakan biaya untuk membuat dan menggunakan gateway NAT di akun Anda. Penggunaan NAT gateway per jam dan tarif pemrosesan data juga berlaku. Untuk tujuan redundansi, Anda harus memiliki gateway NAT di setiap Availability Zone. Dengan cara ini, hilangnya ketersediaan satu Availability Zone tidak mengganggu konektivitas keluar Anda. Karena itu, jika Anda memiliki beban kerja yang kecil, mungkin lebih hemat biaya untuk menggunakan subnet pribadi dan gateway NAT.

Pendekatan jaringan ini didukung baik saat menggunakan Amazon ECS di Amazon EC2 dan saat menggunakannya. AWS Fargate

- Amazon EC2 — Anda dapat meluncurkan EC2 instance di subnet pribadi. Kontainer yang berjalan di EC2 host ini menggunakan jaringan host yang mendasarinya, dan permintaan keluar melalui gateway NAT.
- Fargate - Saat Anda membuat layanan Amazon ECS, tentukan subnet pribadi untuk konfigurasi jaringan layanan Anda, dan jangan gunakan opsi Tetapkan alamat IP publik. Setiap tugas Fargate di-host di subnet pribadi. Lalu lintas keluarnya diarahkan melalui gateway NAT apa pun yang telah Anda kaitkan dengan subnet pribadi itu.

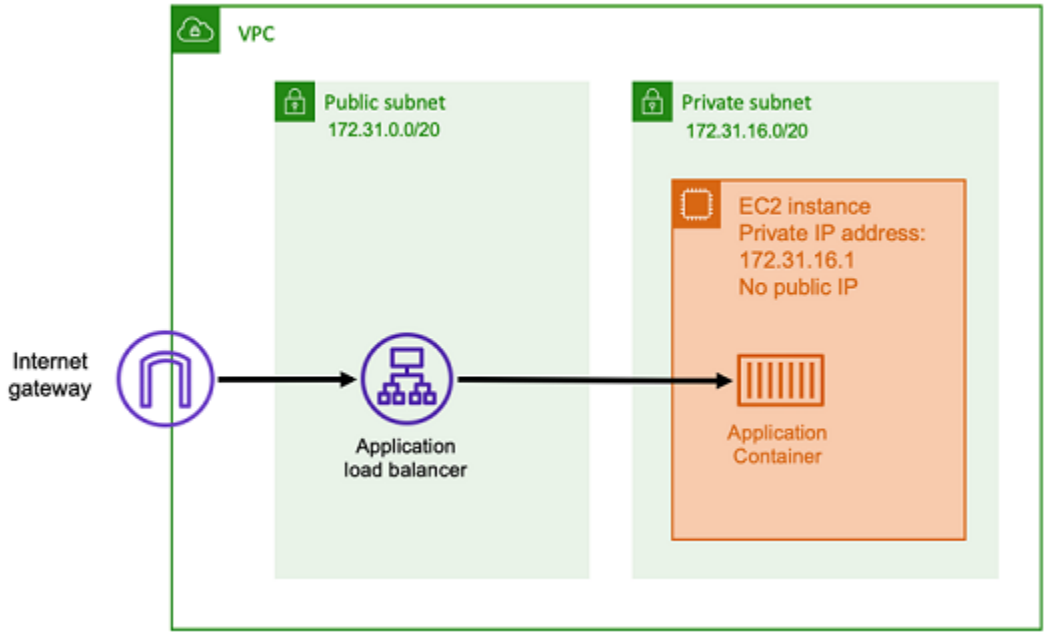
Praktik terbaik untuk menerima koneksi masuk ke Amazon ECS dari internet

Jika Anda menjalankan layanan publik, Anda harus menerima lalu lintas masuk dari internet. Misalnya, situs web publik Anda harus menerima permintaan HTTP masuk dari browser. Dalam kasus seperti itu, host lain di internet juga harus memulai koneksi masuk ke host aplikasi Anda.

Salah satu pendekatan untuk masalah ini adalah meluncurkan container Anda pada host yang berada di subnet publik dengan alamat IP publik. Namun, kami tidak merekomendasikan ini untuk aplikasi skala besar. Untuk ini, pendekatan yang lebih baik adalah memiliki lapisan input yang dapat diskalakan yang berada di antara internet dan aplikasi Anda. Untuk pendekatan ini, Anda dapat menggunakan salah satu AWS layanan yang tercantum di bagian ini sebagai masukan.

Penyeimbang Beban Aplikasi

Application Load Balancer berfungsi pada layer aplikasi. Ini adalah lapisan ketujuh dari model Open Systems Interconnection (OSI). Hal ini membuat Application Load Balancer cocok untuk layanan HTTP publik. Jika Anda memiliki situs web atau HTTP REST API, maka Application Load Balancer adalah penyeimbang beban yang cocok untuk beban kerja ini. Untuk informasi selengkapnya, lihat Apa itu Application Load Balancer? dalam Panduan Pengguna untuk Penyeimbang Beban Aplikasi.



Dengan arsitektur ini, Anda membuat Application Load Balancer di subnet publik sehingga memiliki alamat IP publik dan dapat menerima koneksi inbound dari internet. Ketika Application Load Balancer menerima koneksi masuk, atau lebih khusus permintaan HTTP, ia membuka koneksi ke aplikasi menggunakan alamat IP pribadinya. Kemudian, meneruskan permintaan melalui koneksi internal.

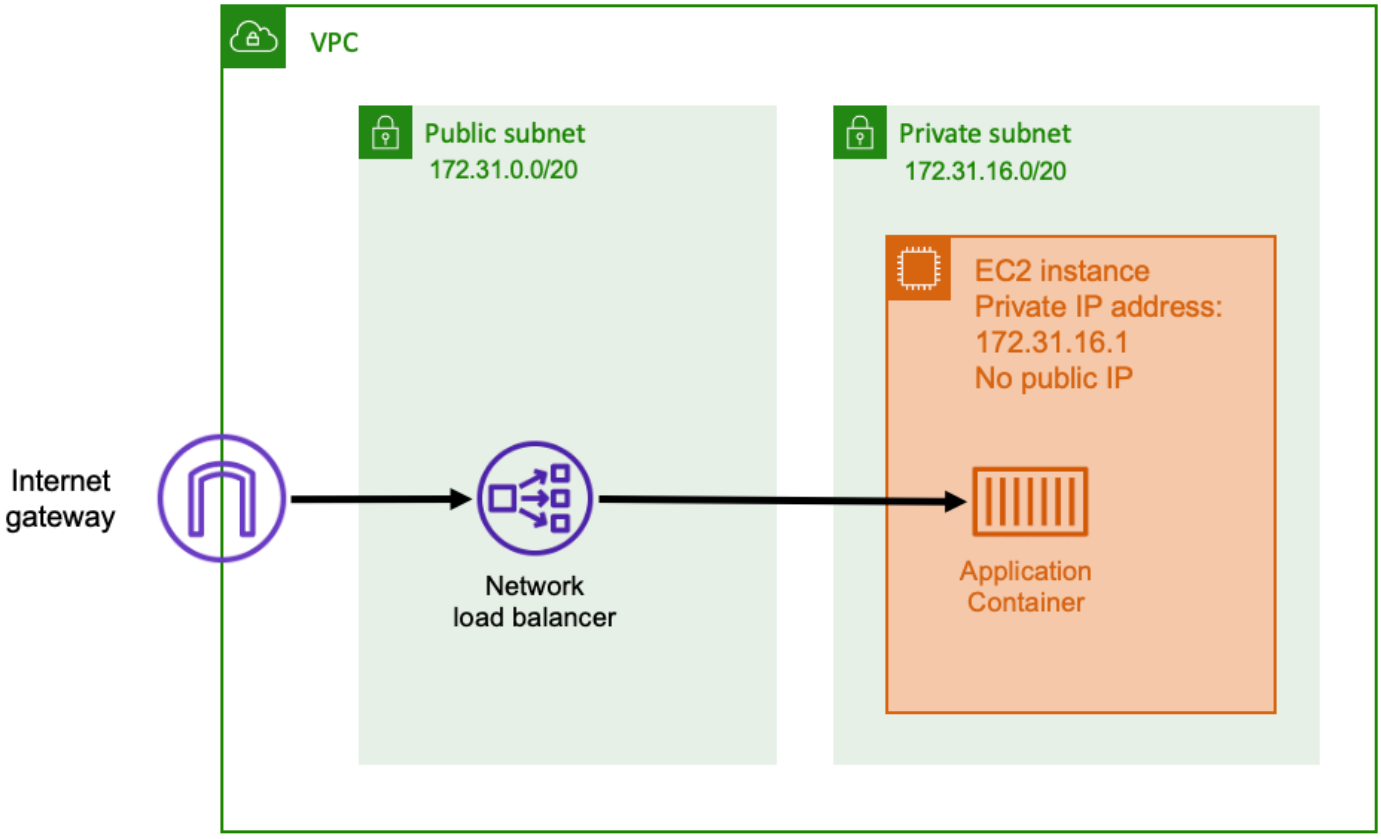
Application Load Balancer memiliki keunggulan sebagai berikut.

- Pengakhiran SSL/TLS — Application Load Balancer dapat mempertahankan komunikasi HTTPS yang aman dan sertifikat untuk komunikasi dengan klien. Ini secara opsional dapat menghentikan koneksi SSL di tingkat penyeimbang beban sehingga Anda tidak perlu menangani sertifikat dalam aplikasi Anda sendiri.
- Perutean lanjutan - Application Load Balancer dapat memiliki beberapa nama host DNS. Ini juga memiliki kemampuan routing lanjutan untuk mengirim permintaan HTTP masuk ke tujuan yang berbeda berdasarkan metrik seperti nama host atau jalur permintaan. Ini berarti Anda dapat menggunakan Application Load Balancer tunggal sebagai input untuk banyak layanan internal yang berbeda, atau bahkan layanan mikro pada jalur yang berbeda dari REST API.
- Dukungan gRPC dan soket web - Application Load Balancer dapat menangani lebih dari sekadar HTTP. Itu juga dapat memuat keseimbangan gRPC dan layanan berbasis websocket, dengan dukungan HTTP/2.
- Keamanan - Application Load Balancer membantu melindungi aplikasi Anda dari lalu lintas berbahaya. Ini mencakup fitur-fitur seperti mitigasi sinkronisasi HTTP, dan terintegrasi dengan

AWS Web Application Firewall ().AWS WAF AWS WAF selanjutnya dapat menyaring lalu lintas berbahaya yang mungkin berisi pola serangan, seperti injeksi SQL atau skrip lintas situs.

Penyeimbang Beban Jaringan

Penyeimbang Beban jaringan berfungsi pada lapisan keempat dari model Open Systems Interkoneksi (OSI). Ini cocok untuk protokol non-HTTP atau skenario di mana end-to-end enkripsi diperlukan, tetapi tidak memiliki fitur HTTP-spesifik yang sama dari Application Load Balancer. Oleh karena itu, Network Load Balancer paling cocok untuk aplikasi yang tidak menggunakan HTTP. Untuk informasi selengkapnya, lihat Apa itu Network Load Balancer? dalam Panduan Pengguna untuk Network Load Balancers.



Ketika Network Load Balancer digunakan sebagai input, fungsinya mirip dengan Application Load Balancer. Ini karena dibuat di subnet publik dan memiliki alamat IP publik yang dapat diakses di internet. Network Load Balancer kemudian membuka koneksi ke alamat IP pribadi host yang menjalankan container Anda, dan mengirimkan paket dari sisi publik ke sisi pribadi.

Fitur Network Load Balancer

Karena Network Load Balancer beroperasi pada tingkat yang lebih rendah dari tumpukan jaringan, ia tidak memiliki serangkaian fitur yang sama dengan Application Load Balancer. Namun, ia memang memiliki fitur-fitur penting berikut.

- End-to-end Enkripsi — Karena Network Load Balancer beroperasi pada lapisan keempat model OSI, ia tidak membaca isi paket. Ini membuatnya cocok untuk komunikasi load balancing yang membutuhkan end-to-end enkripsi.
- Enkripsi TLS — Selain end-to-end enkripsi, Network Load Balancer juga dapat menghentikan koneksi TLS. Dengan cara ini, aplikasi backend Anda tidak harus mengimplementasikan TLS mereka sendiri.
- Dukungan UDP — Karena Network Load Balancer beroperasi pada lapisan keempat model OSI, ini cocok untuk beban kerja dan protokol non HTTP selain TCP.

Menutup koneksi

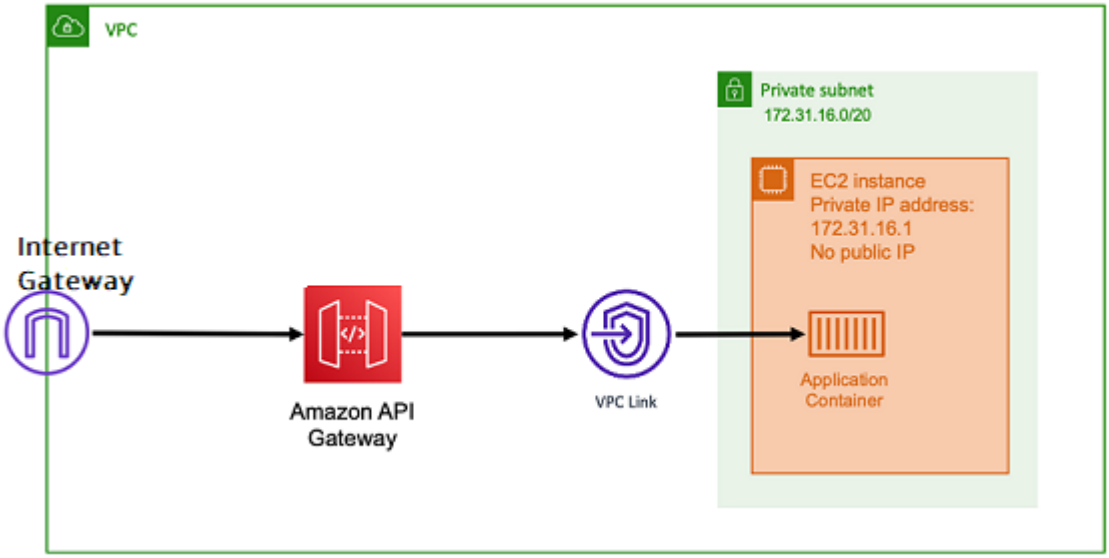
Karena Network Load Balancer tidak mengamati protokol aplikasi pada lapisan yang lebih tinggi dari model OSI, ia tidak dapat mengirim pesan penutupan ke klien dalam protokol tersebut. Berbeda dengan Application Load Balancer, koneksi tersebut harus ditutup oleh aplikasi atau Anda dapat mengkonfigurasi Network Load Balancer untuk menutup koneksi lapisan keempat ketika tugas dihentikan atau diganti. Lihat setelan penghentian sambungan untuk grup target Network Load Balancer dalam dokumentasi Network Load Balancer.

Membiarkan Network Load Balancer menutup koneksi pada lapisan keempat dapat menyebabkan klien menampilkan pesan kesalahan yang tidak diinginkan, jika klien tidak menanganinya. Lihat Builders Library untuk informasi selengkapnya tentang konfigurasi klien yang direkomendasikan di sini.

Metode untuk menutup koneksi akan bervariasi menurut aplikasi, namun salah satu caranya adalah memastikan bahwa penundaan deregistrasi target Network Load Balancer lebih lama daripada batas waktu koneksi klien. Klien akan timeout terlebih dahulu dan menyambung kembali dengan anggun melalui Network Load Balancer ke tugas berikutnya sementara tugas lama perlahan-lahan menguras semua kliennya. Untuk informasi selengkapnya tentang penundaan deregistrasi target Network Load Balancer, lihat dokumentasi Network Load Balancer.

API HTTP Gerbang Amazon API

Amazon API Gateway cocok untuk aplikasi HTTP dengan semburan mendadak dalam volume permintaan atau volume permintaan rendah. Untuk informasi selengkapnya, lihat Apa itu Amazon API Gateway? di Panduan Pengembang API Gateway.



Model harga untuk Application Load Balancer dan Network Load Balancer mencakup harga per jam untuk menjaga penyeimbang beban tersedia untuk menerima koneksi masuk setiap saat. Sebaliknya, API Gateway mengenakan biaya untuk setiap permintaan secara terpisah. Ini memiliki efek bahwa, jika tidak ada permintaan masuk, tidak ada biaya. Di bawah beban lalu lintas yang tinggi, Application Load Balancer atau Network Load Balancer dapat menangani volume permintaan yang lebih besar dengan harga per permintaan yang lebih murah daripada API Gateway. Namun, jika Anda memiliki jumlah permintaan yang rendah secara keseluruhan atau memiliki periode lalu lintas rendah, maka harga kumulatif untuk menggunakan API Gateway harus lebih hemat biaya daripada membayar biaya per jam untuk mempertahankan penyeimbang beban yang kurang dimanfaatkan. API Gateway juga dapat men-cache respons API, yang mungkin menghasilkan tingkat permintaan backend yang lebih rendah.

Fungsi API Gateway yang menggunakan tautan VPC yang memungkinkan layanan AWS terkelola terhubung ke host di dalam subnet pribadi VPC Anda, menggunakan alamat IP pribadinya. Ini dapat mendeteksi alamat IP pribadi ini dengan melihat catatan penemuan AWS Cloud Map layanan yang dikelola oleh Amazon ECS Service Discovery.

API Gateway mendukung fitur-fitur berikut.

- Operasi API Gateway mirip dengan penyeimbang beban, tetapi memiliki kemampuan tambahan yang unik untuk manajemen API
- API Gateway menyediakan kemampuan tambahan seputar otorisasi klien, tingkatan penggunaan, dan request/response modifikasi. Untuk informasi selengkapnya, lihat fitur Amazon API Gateway.
- API Gateway dapat mendukung titik akhir gateway API edge, regional, dan privat. Titik akhir tepi tersedia melalui CloudFront distribusi terkelola. Titik akhir regional dan swasta keduanya lokal untuk suatu Wilayah.
- Pengakhiran SSL/TLS
- Merutekan jalur HTTP yang berbeda ke layanan mikro backend yang berbeda

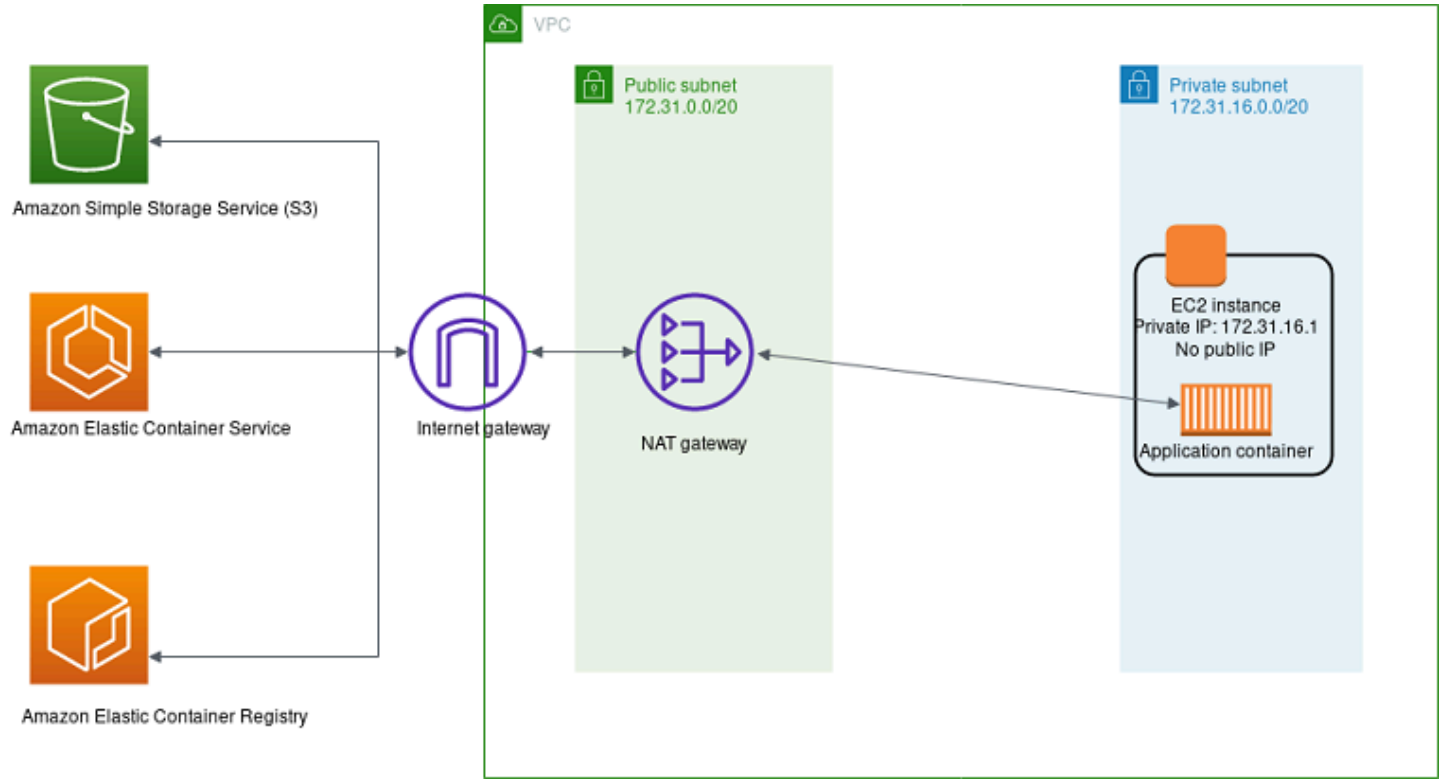
Selain fitur sebelumnya, API Gateway juga mendukung penggunaan otorisasi Lambda khusus yang dapat Anda gunakan untuk melindungi API Anda dari penggunaan yang tidak sah. Untuk informasi selengkapnya, lihat Catatan Bidang: Berbasis Kontainer Tanpa Server APIs dengan Amazon ECS dan Amazon API Gateway.

Praktik terbaik untuk menghubungkan Amazon ECS ke AWS layanan dari dalam VPC Anda

Agar Amazon ECS berfungsi dengan baik, agen kontainer Amazon ECS yang berjalan di setiap host harus berkomunikasi dengan bidang kontrol Amazon ECS. Jika Anda menyimpan gambar kontainer di Amazon ECR, EC2 host Amazon harus berkomunikasi ke titik akhir layanan Amazon ECR, dan ke Amazon S3, tempat lapisan gambar disimpan. Jika Anda menggunakan AWS layanan lain untuk aplikasi kontainer Anda, seperti menyimpan data yang disimpan di DynamoDB, periksa kembali apakah layanan ini juga memiliki dukungan jaringan yang diperlukan.

Gateway NAT

Menggunakan gateway NAT adalah cara termudah untuk memastikan bahwa tugas Amazon ECS Anda dapat mengakses layanan lain AWS . Untuk informasi lebih lanjut tentang pendekatan ini, lihatSubnet pribadi dan gateway NAT.



Berikut ini adalah kerugian untuk menggunakan pendekatan ini:

- Anda tidak dapat membatasi tujuan apa yang dapat berkomunikasi dengan gateway NAT. Anda juga tidak dapat membatasi tujuan mana yang dapat dikomunikasikan oleh tingkat backend Anda tanpa mengganggu semua komunikasi keluar dari VPC Anda.
- Gateway NAT mengenakan biaya untuk setiap GB data yang lewat. Jika Anda menggunakan gateway NAT untuk salah satu operasi berikut, Anda dikenakan biaya untuk setiap GB bandwidth:
  - Mengunduh file besar dari Amazon S3
  - Melakukan volume query database yang tinggi ke DynamoDB
  - Menarik gambar dari Amazon ECR

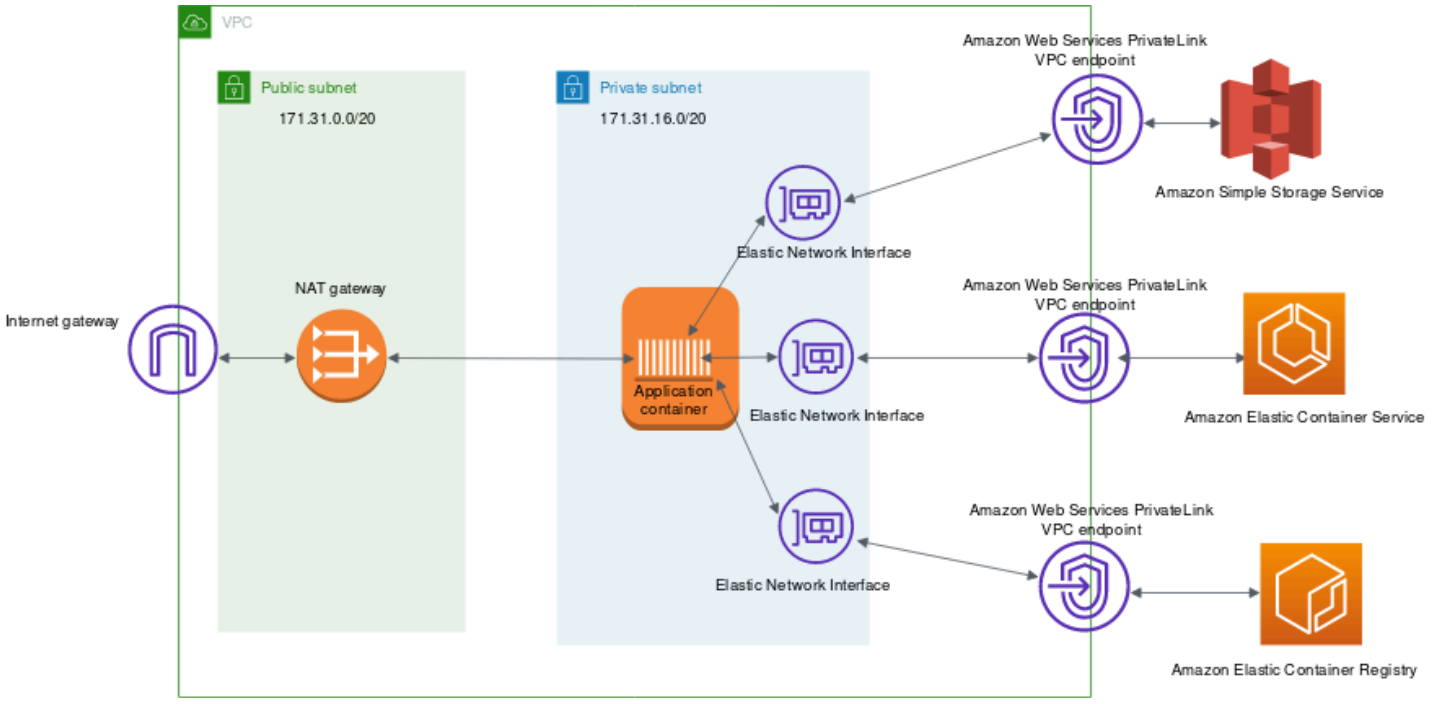
  Selain itu, gateway NAT mendukung bandwidth 5 Gbps dan secara otomatis menskalakan hingga 45 Gbps. Jika Anda merutekan melalui gateway NAT tunggal, aplikasi yang memerlukan koneksi bandwidth yang sangat tinggi mungkin mengalami kendala jaringan. Sebagai solusinya, Anda dapat membagi beban kerja Anda di beberapa subnet dan memberikan masing-masing subnet gateway NAT sendiri.

AWS PrivateLink

AWS PrivateLink menyediakan konektivitas pribadi antara VPCs, AWS layanan, dan jaringan lokal Anda tanpa mengekspos lalu lintas Anda ke internet publik.

Endpoint VPC memungkinkan koneksi pribadi antara VPC Anda AWS dan layanan yang didukung serta layanan titik akhir VPC. Lalu lintas antara VPC Anda dan layanan lainnya tidak meninggalkan jaringan Amazon. Titik akhir VPC tidak memerlukan gateway internet, gateway pribadi virtual, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect EC2Instans Amazon di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan sumber daya dalam layanan.

Diagram berikut menunjukkan cara kerja komunikasi ke AWS layanan saat Anda menggunakan titik akhir VPC alih-alih gateway internet. AWS PrivateLink ketentuan antarmuka jaringan elastis (ENIs) di dalam subnet, dan aturan perutean VPC digunakan untuk mengirim komunikasi apa pun ke nama host layanan melalui ENI, langsung ke layanan tujuan. AWS Lalu lintas ini tidak perlu lagi menggunakan gateway NAT atau gateway internet.



Berikut ini adalah beberapa titik akhir VPC umum yang digunakan dengan layanan Amazon ECS.

- Titik akhir Gateway untuk Amazon S3
- Titik akhir DynamoDB VPC
- Titik akhir Amazon ECS VPC
- Titik akhir Amazon ECR VPC

Banyak AWS layanan lain yang mendukung titik akhir VPC. Jika Anda menggunakan AWS layanan apa pun secara berlebihan, Anda harus mencari dokumentasi khusus untuk layanan itu dan cara membuat titik akhir VPC untuk lalu lintas tersebut.

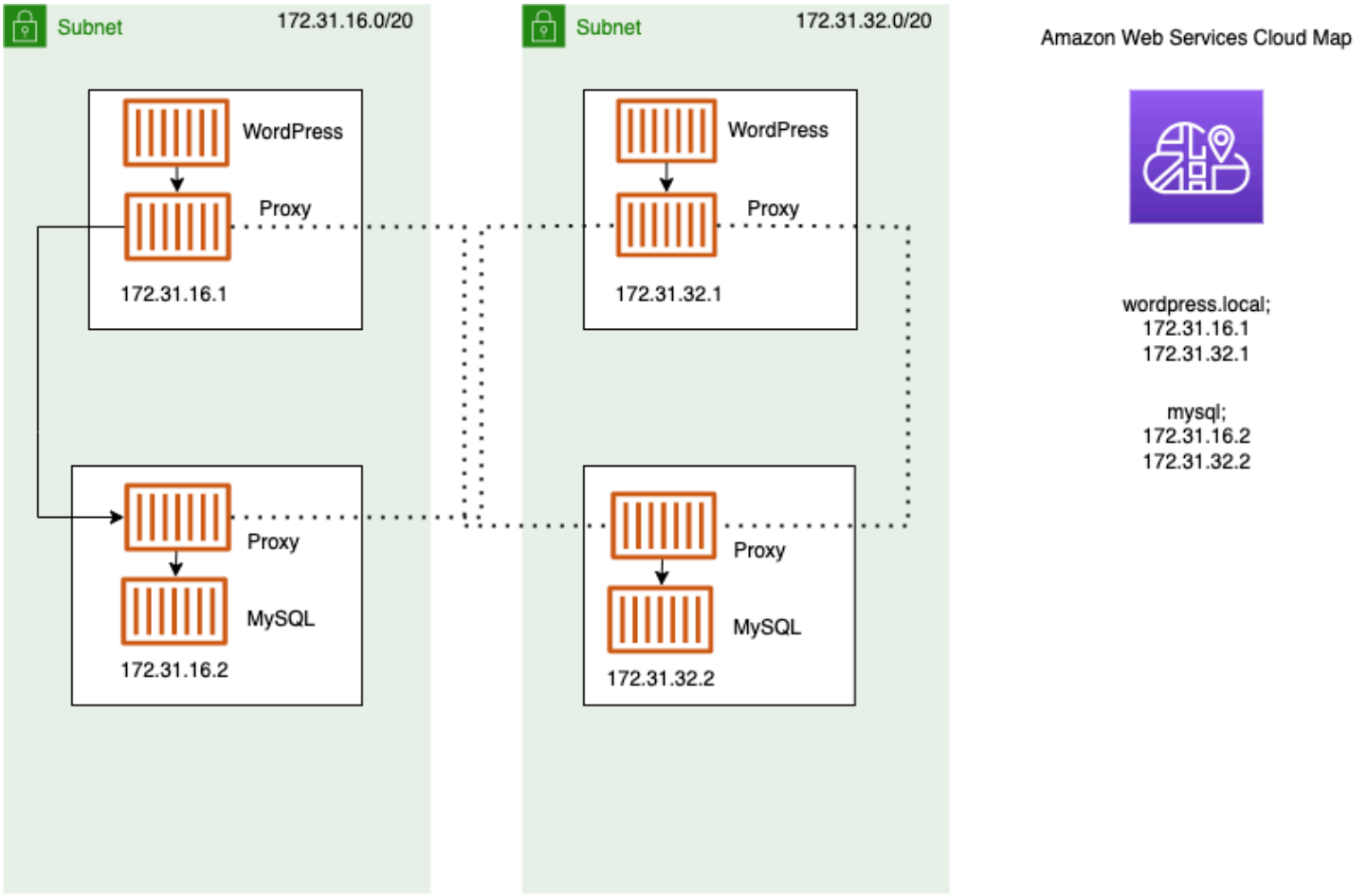
Praktik terbaik untuk menghubungkan layanan Amazon ECS di VPC

Dengan menggunakan tugas Amazon ECS di VPC, Anda dapat membagi aplikasi monolitik menjadi beberapa bagian terpisah yang dapat digunakan dan diskalakan secara independen di lingkungan yang aman. Arsitektur ini disebut arsitektur berorientasi layanan (SOA) atau layanan mikro. Namun, mungkin sulit untuk memastikan bahwa semua bagian ini, baik di dalam maupun di luar VPC, dapat berkomunikasi satu sama lain. Ada beberapa pendekatan untuk memfasilitasi komunikasi, semuanya dengan kelebihan dan kekurangan yang berbeda.

Menggunakan Service Connect

Kami merekomendasikan Service Connect, yang menyediakan konfigurasi Amazon ECS untuk penemuan layanan, konektivitas, dan pemantauan lalu lintas. Dengan Service Connect, aplikasi Anda dapat menggunakan nama pendek dan port standar untuk terhubung ke layanan di cluster yang sama, cluster lain, termasuk VPCs di seluruh Wilayah yang sama. Untuk informasi selengkapnya, lihat Amazon ECS Service Connect.

Saat Anda menggunakan Service Connect, Amazon ECS mengelola semua bagian penemuan layanan: membuat nama yang dapat ditemukan, mengelola entri secara dinamis untuk setiap tugas saat tugas dimulai dan dihentikan, menjalankan agen di setiap tugas yang dikonfigurasi untuk menemukan nama. Aplikasi Anda dapat mencari nama dengan menggunakan fungsionalitas standar untuk nama DNS dan membuat koneksi. Jika aplikasi Anda sudah melakukan ini, Anda tidak perlu memodifikasi aplikasi Anda untuk menggunakan Service Connect.



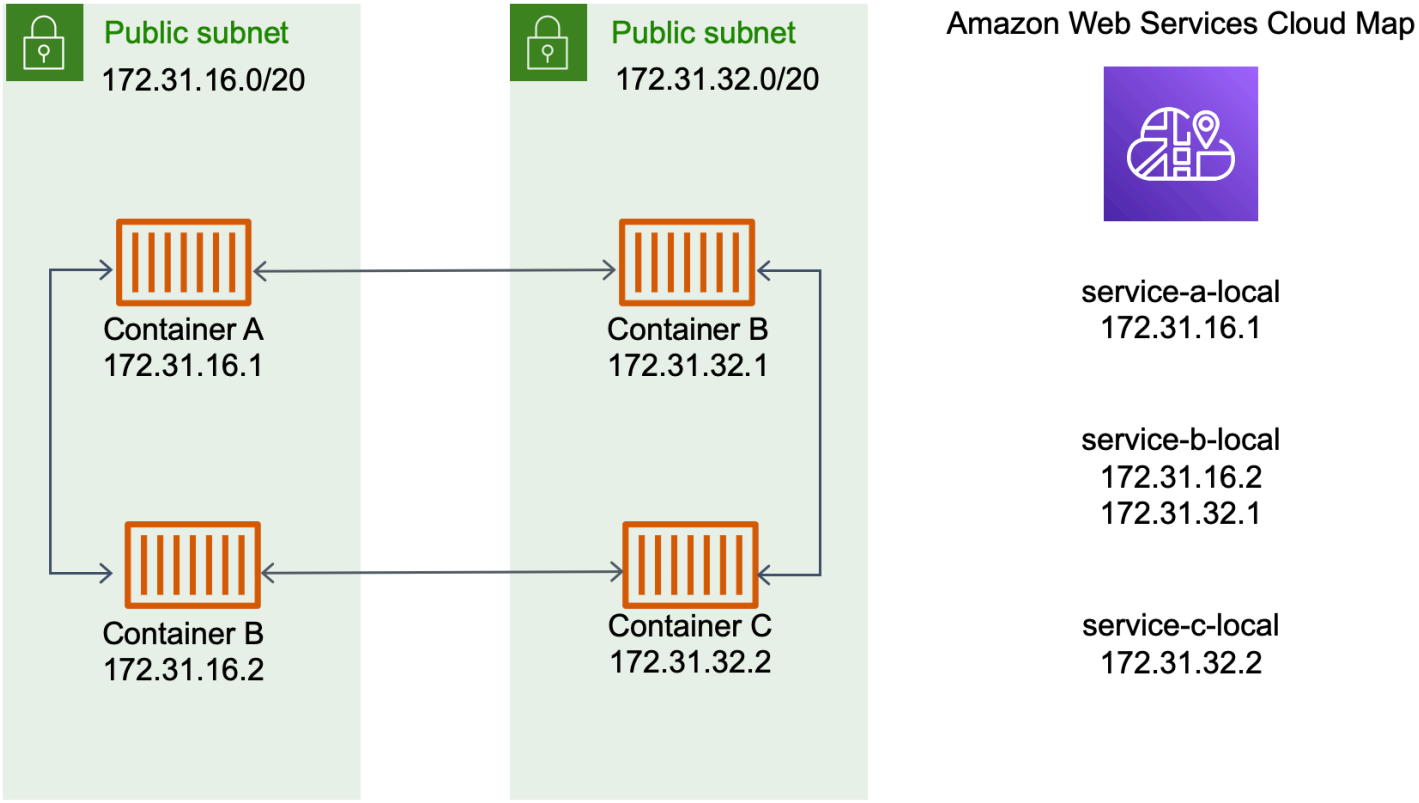
Perubahan hanya terjadi selama penerapan

Anda menyediakan konfigurasi lengkap di dalam setiap layanan dan definisi tugas. Amazon ECS mengelola perubahan pada konfigurasi ini di setiap penyebaran layanan, untuk memastikan bahwa semua tugas dalam penerapan berperilaku dengan cara yang sama. Misalnya, masalah umum dengan DNS sebagai penemuan layanan adalah mengendalikan migrasi. Jika Anda mengubah nama DNS untuk menunjuk ke alamat IP pengganti baru, mungkin diperlukan waktu TTL maksimum sebelum semua klien mulai menggunakan layanan baru. Dengan Service Connect, penyebaran klien memperbarui konfigurasi dengan mengganti tugas klien. Anda dapat mengonfigurasi pemutus sirkuit penyebaran dan konfigurasi penerapan lainnya untuk memengaruhi perubahan Service Connect dengan cara yang sama seperti penerapan lainnya.

Menggunakan penemuan layanan

Pendekatan lain untuk service-to-service komunikasi adalah komunikasi langsung menggunakan penemuan layanan. Dalam pendekatan ini, Anda dapat menggunakan integrasi penemuan AWS Cloud Map layanan dengan Amazon ECS. Menggunakan penemuan layanan, Amazon ECS

menyinkronkan daftar tugas yang diluncurkan ke AWS Cloud Map, yang mempertahankan nama host DNS yang menyelesaikan ke alamat IP internal dari satu atau lebih tugas dari layanan tertentu. Layanan lain di Amazon VPC dapat menggunakan nama host DNS ini untuk mengirim lalu lintas langsung ke wadah lain menggunakan alamat IP internalnya. Untuk informasi selengkapnya, lihat Penemuan Layanan.



Dalam diagram sebelumnya, ada tiga layanan. `service-a-local`memiliki satu wadah dan berkomunikasi dengan`service-b-local`, yang memiliki dua wadah. `service-b-local`juga harus berkomunikasi dengan`service-c-local`, yang memiliki satu wadah. Setiap kontainer di ketiga layanan ini dapat menggunakan nama DNS internal AWS Cloud Map untuk menemukan alamat IP internal kontainer dari layanan hilir yang perlu dikomunikasikan.

Pendekatan service-to-service komunikasi ini memberikan latensi rendah. Pada pandangan pertama, ini juga sederhana karena tidak ada komponen tambahan di antara wadah. Lalu lintas bergerak langsung dari satu kontainer ke kontainer lainnya.

Pendekatan ini cocok saat menggunakan mode `awsvpc` jaringan, di mana setiap tugas memiliki alamat IP uniknya sendiri. Sebagian besar perangkat lunak hanya mendukung penggunaan `A` catatan DNS, yang menyelesaikan langsung ke alamat IP. Saat menggunakan mode `awsvpc` jaringan, alamat IP untuk setiap tugas adalah `A` catatan. Namun, jika Anda menggunakan mode `bridge`

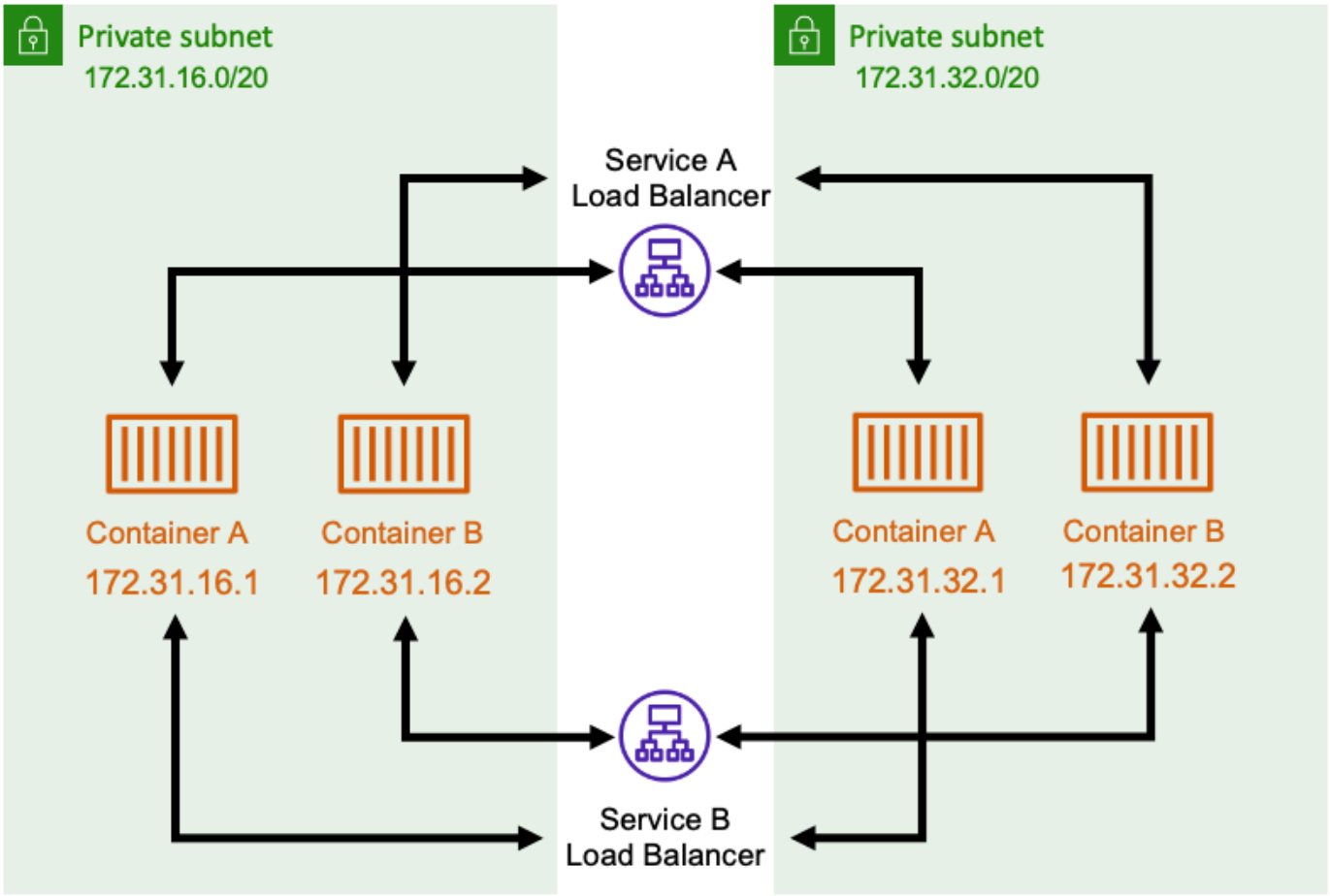
jaringan, beberapa kontainer dapat berbagi alamat IP yang sama. Selain itu, pemetaan port dinamis menyebabkan kontainer diberi nomor port secara acak pada alamat IP tunggal itu. Pada titik ini, `A` catatan tidak lagi cukup untuk penemuan layanan. Anda juga harus menggunakan `SRV` catatan. Jenis catatan ini dapat melacak alamat IP dan nomor port tetapi mengharuskan Anda mengonfigurasi aplikasi dengan tepat. Beberapa aplikasi bawaan yang Anda gunakan mungkin tidak mendukung `SRV` catatan.

Keuntungan lain dari mode `awsvpc` jaringan adalah Anda memiliki grup keamanan unik untuk setiap layanan. Anda dapat mengonfigurasi grup keamanan ini untuk mengizinkan koneksi masuk hanya dari layanan hulu tertentu yang perlu berbicara dengan layanan tersebut.

Kerugian utama dari service-to-service komunikasi langsung menggunakan penemuan layanan adalah Anda harus menerapkan logika tambahan untuk mencoba ulang dan menangani kegagalan koneksi. Catatan DNS memiliki periode time-to-live (TTL) yang mengontrol berapa lama mereka di-cache. Dibutuhkan beberapa waktu agar catatan DNS diperbarui dan cache kedaluwarsa sehingga aplikasi Anda dapat mengambil versi terbaru dari catatan DNS. Jadi, aplikasi Anda mungkin akan menyelesaikan catatan DNS untuk menunjuk ke wadah lain yang sudah tidak ada lagi. Aplikasi Anda perlu menangani percobaan ulang dan memiliki logika untuk mengabaikan backend yang buruk.

Menggunakan penyeimbang beban internal

Pendekatan lain untuk service-to-service komunikasi adalah dengan menggunakan penyeimbang beban internal. Penyeimbang beban internal ada sepenuhnya di dalam VPC Anda dan hanya dapat diakses oleh layanan di dalam VPC Anda.



Penyeimbang beban mempertahankan ketersediaan tinggi dengan menyebarkan sumber daya yang berlebihan ke setiap subnet. Ketika sebuah wadah dari `serviceA` kebutuhan untuk berkomunikasi dengan wadah dari`serviceB`, itu membuka koneksi ke penyeimbang beban. Penyeimbang beban kemudian membuka koneksi ke wadah dari`service B`. Load balancer berfungsi sebagai tempat terpusat untuk mengelola semua koneksi antara setiap layanan.

Jika wadah `serviceB` berhenti, maka penyeimbang beban dapat mengeluarkan wadah itu dari kolam. Load balancer juga melakukan pemeriksaan kesehatan terhadap setiap target hilir di kolam dan secara otomatis dapat menghapus target buruk dari kolam sampai mereka menjadi sehat kembali. Aplikasi tidak perlu lagi menyadari berapa banyak kontainer hilir yang ada. Mereka hanya membuka koneksi mereka ke penyeimbang beban.

Pendekatan ini menguntungkan untuk semua mode jaringan. Penyeimbang beban dapat melacak alamat IP tugas saat menggunakan mode `awsvpc` jaringan, serta kombinasi alamat IP dan port yang lebih canggih saat menggunakan mode `bridge` jaringan. Ini mendistribusikan lalu lintas secara

merata di semua alamat IP dan kombinasi port, bahkan jika beberapa kontainer benar-benar di-host pada EC2 instance Amazon yang sama, hanya pada port yang berbeda.

Salah satu kelemahan dari pendekatan ini adalah biaya. Agar sangat tersedia, penyeimbang beban harus memiliki sumber daya di setiap Availability Zone. Ini menambah biaya tambahan karena overhead membayar penyeimbang beban dan untuk jumlah lalu lintas yang melewati penyeimbang beban.

Namun, Anda dapat mengurangi biaya overhead dengan memiliki beberapa layanan berbagi penyeimbang beban. Ini sangat cocok untuk layanan REST yang menggunakan Application Load Balancer. Anda dapat membuat aturan perutean berbasis jalur yang merutekan lalu lintas ke layanan yang berbeda. Misalnya, `/api/user/*` mungkin merutekan ke kontainer yang merupakan bagian dari `user` layanan, sedangkan `/api/order/*` mungkin merutekan ke `order` layanan terkait. Dengan pendekatan ini, Anda hanya membayar satu Application Load Balancer, dan memiliki satu URL yang konsisten untuk API Anda. Namun, Anda dapat membagi lalu lintas ke berbagai layanan mikro di backend.

Praktik terbaik untuk jaringan layanan Amazon ECS di seluruh Akun AWS dan VPCs

Jika Anda bagian dari organisasi dengan beberapa tim dan divisi, Anda mungkin menerapkan layanan secara independen ke dalam yang terpisah VPCs di dalam yang dibagikan Akun AWS atau VPCs yang terkait dengan beberapa individu. Akun AWS Tidak peduli ke arah mana Anda menyebarkan layanan Anda, kami sarankan Anda melengkapi komponen jaringan Anda untuk membantu mengarahkan lalu lintas di antaranya VPCs. Untuk ini, beberapa AWS layanan dapat digunakan untuk melengkapi komponen jaringan Anda yang ada.

- AWS Transit Gateway — Anda harus mempertimbangkan layanan jaringan ini terlebih dahulu. Layanan ini berfungsi sebagai hub pusat untuk merutekan koneksi Anda antara Amazon VPCs Akun AWS, dan jaringan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit? di Panduan Gerbang Transit VPC Amazon.
- Dukungan Amazon VPC dan VPN — Anda dapat menggunakan layanan ini untuk membuat koneksi site-to-site VPN untuk menghubungkan jaringan lokal ke VPC Anda. Untuk informasi lebih lanjut, lihat Apa itu AWS Site-to-Site VPN? dalam AWS Site-to-Site VPN User Guide.
- Amazon VPC — Anda dapat menggunakan peering VPC Amazon untuk membantu Anda menghubungkan beberapa VPCs, baik di akun yang sama, atau di seluruh akun. Untuk informasi selengkapnya, lihat Apa yang dimaksud peering VPC? di Panduan Peering Amazon VPC.

- Dibagikan VPCs - Anda dapat menggunakan subnet VPC dan VPC di beberapa akun. AWS Untuk informasi selengkapnya, lihat Bekerja dengan dibagikan VPCs di Panduan Pengguna Amazon VPC.

AWS layanan untuk pemecahan masalah jaringan Amazon ECS

Layanan dan fitur berikut dapat membantu Anda mendapatkan wawasan tentang konfigurasi jaringan dan layanan Anda. Anda dapat menggunakan informasi ini untuk memecahkan masalah jaringan dan mengoptimalkan layanan Anda dengan lebih baik.

CloudWatch Wawasan Kontainer

CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Metrik mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan. Mereka tersedia di dasbor CloudWatch otomatis. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer di Amazon ECS di CloudWatch Panduan Pengguna Amazon.

AWS X-Ray

AWS X-Ray adalah layanan penelusuran yang dapat Anda gunakan untuk mengumpulkan informasi tentang permintaan jaringan yang dibuat aplikasi Anda. Anda dapat menggunakan SDK untuk instrumen aplikasi Anda dan menangkap timing dan kode respons lalu lintas antara layanan Anda, dan antara layanan dan titik akhir AWS layanan Anda. Untuk informasi selengkapnya, lihat Apa yang ada AWS X-Ray di Panduan AWS X-Ray Pengembang.

Anda juga dapat menjelajahi AWS X-Ray grafik tentang bagaimana jaringan layanan Anda satu sama lain. Atau, gunakan mereka untuk mengeksplorasi statistik agregat tentang bagaimana kinerja setiap service-to-service tautan. Terakhir, Anda dapat menyelam lebih dalam ke transaksi tertentu untuk melihat bagaimana segmen yang mewakili panggilan jaringan terkait dengan transaksi tertentu.

Anda dapat menggunakan fitur-fitur ini untuk mengidentifikasi apakah ada hambatan jaringan atau jika layanan tertentu dalam jaringan Anda tidak berfungsi seperti yang diharapkan.

Log Alur VPC

Anda dapat menggunakan log aliran VPC Amazon untuk menganalisis performa jaringan dan masalah konektivitas debug. Dengan log aliran VPC diaktifkan, Anda dapat menangkap log semua

koneksi di VPC Anda. Ini termasuk koneksi ke antarmuka jaringan yang terkait dengan Elastic Load Balancing, Amazon RDS, gateway NAT, dan layanan AWS utama lainnya yang mungkin Anda gunakan. Untuk informasi selengkapnya, lihat Log Alur VPC di Panduan Pengguna Amazon VPC.

Kiat penyetelan jaringan

Ada beberapa pengaturan yang dapat Anda selaraskan untuk meningkatkan jaringan Anda.

nofile ulimit

Jika Anda mengharapkan aplikasi Anda memiliki lalu lintas tinggi dan menangani banyak koneksi bersamaan, Anda harus mempertimbangkan kuota sistem untuk jumlah file yang diizinkan. Ketika ada banyak soket jaringan terbuka, masing-masing harus diwakili oleh deskriptor file. Jika kuota deskriptor file Anda terlalu rendah, itu akan membatasi soket jaringan Anda. Ini mengakibatkan koneksi atau kesalahan yang gagal. Anda dapat memperbarui kuota khusus penampung untuk jumlah file dalam definisi tugas Amazon ECS. Jika Anda menjalankan di Amazon EC2 (bukan AWS Fargate), maka Anda mungkin juga perlu menyesuaikan kuota ini pada EC2 instans Amazon yang mendasarinya.

sysctl bersih

Kategori lain dari pengaturan yang dapat disetel adalah pengaturan `sysctl` bersih. Anda harus merujuk ke pengaturan khusus untuk distribusi Linux pilihan Anda. Banyak dari pengaturan ini menyesuaikan ukuran buffer baca dan tulis. Ini dapat membantu dalam beberapa situasi saat menjalankan EC2 instans Amazon skala besar yang memiliki banyak wadah di dalamnya.

Optimalkan waktu peluncuran tugas Amazon ECS

Untuk mempercepat peluncuran tugas Anda, pertimbangkan rekomendasi berikut.

- Cache gambar wadah dan instance binpack

  Jika Anda menggunakan tipe EC2 peluncuran, Anda dapat mengonfigurasi perilaku tarik agen penampung Amazon ECS ke`ECS_IMAGE_PULL_BEHAVIOR`:`prefer-cached`. Gambar ditarik dari jarak jauh jika tidak ada gambar yang di-cache. Jika tidak, citra cache pada instans digunakan. Pembersihan gambar otomatis dimatikan untuk wadah untuk memastikan bahwa gambar yang di-cache tidak dihapus. Ini mengurangi waktu tarik gambar untuk peluncuran berikutnya. Efek caching bahkan lebih besar ketika Anda memiliki kepadatan tugas yang tinggi dalam instance container Anda, yang dapat Anda konfigurasikan menggunakan strategi `binpack` penempatan. Caching gambar kontainer sangat bermanfaat untuk beban kerja berbasis windows yang biasanya memiliki

ukuran gambar kontainer yang besar (puluhan GBs). Saat menggunakan strategi `binpack` penempatan, Anda juga dapat mempertimbangkan untuk menggunakan trunking Elastic Network Interface (ENI) untuk menempatkan lebih banyak tugas dengan mode `awsvpc` jaringan pada setiap instance container. Trunking ENI meningkatkan jumlah tugas yang dapat Anda jalankan dalam mode. `awsvpc` Misalnya, instance c5.large yang mungkin mendukung menjalankan hanya 2 tugas secara bersamaan, dapat menjalankan hingga 10 tugas dengan trunking ENI.

- Pilih mode jaringan yang optimal

  Meskipun ada banyak contoh di mana mode `awsvpc` jaringan ideal, mode jaringan ini secara inheren dapat meningkatkan latensi peluncuran tugas, karena untuk setiap tugas dalam mode`awsvpc`, alur kerja Amazon ECS perlu menyediakan dan melampirkan ENI dengan menjalankan Amazon EC2 APIs yang menambahkan overhead beberapa detik ke peluncuran tugas Anda. Sebaliknya, keuntungan utama menggunakan mode `awsvpc` jaringan adalah bahwa setiap tugas memiliki grup keamanan untuk mengizinkan atau menolak lalu lintas. Ini berarti Anda memiliki fleksibilitas yang lebih besar untuk mengontrol komunikasi antara tugas dan layanan pada tingkat yang lebih terperinci. Jika kecepatan penerapan adalah prioritas Anda, Anda dapat mempertimbangkan untuk menggunakan `bridge` mode untuk mempercepat peluncuran tugas. Untuk informasi selengkapnya, lihat the section called “AWSVPC modus jaringan”.

- Lacak siklus hidup peluncuran tugas Anda untuk menemukan peluang pengoptimalan

  Seringkali sulit untuk mengetahui jumlah waktu yang dibutuhkan aplikasi Anda untuk memulai. Meluncurkan image container Anda, menjalankan skrip start-up, dan konfigurasi lainnya selama start-up aplikasi dapat memakan waktu yang mengejutkan. Anda dapat menggunakan titik akhir metadata Tugas untuk memposting metrik guna melacak waktu mulai aplikasi dari `StartedAt` respons metadata penampung hingga waktu tugas atau layanan Anda. `StartedAt` Dengan data ini, Anda dapat memahami bagaimana aplikasi Anda berkontribusi terhadap total waktu peluncuran, dan menemukan area di mana Anda dapat mengurangi overhead khusus aplikasi yang tidak perlu dan mengoptimalkan gambar kontainer Anda. Untuk informasi selengkapnya, lihat Praktik terbaik penskalaan dan manajemen kapasitas Amazon ECS Auto.

- Pilih jenis instans yang optimal (untuk jenis EC2 peluncuran)

  Memilih jenis instans yang benar didasarkan pada reservasi sumber daya (misalnya, CPU, memori) yang Anda konfigurasikan pada tugas Anda. Oleh karena itu, saat mengukur instance, Anda dapat menghitung berapa banyak tugas yang dapat ditempatkan pada satu instance. Contoh sederhana dari tugas yang ditempatkan dengan baik, adalah hosting 4 tugas yang membutuhkan 0,5 vCPU dan 2GB reservasi memori dalam instance m5.large (mendukung 2 vCPU dan memori 8 GB). Reservasi definisi tugas ini memanfaatkan sepenuhnya sumber daya instans.

Mengoperasikan Amazon ECS dalam skala besar

Saat Anda mulai mengoperasikan Amazon ECS dalam skala besar, pertimbangkan bagaimana kuota layanan dan pembatasan API untuk Amazon ECS dan Layanan AWS yang terintegrasi dengan Amazon ECS dapat memengaruhi Anda.

Topik

- Kuota layanan Amazon ECS dan batas pembatasan API
- Menangani masalah pelambatan Amazon ECS

Kuota layanan Amazon ECS dan batas pembatasan API

Amazon ECS terintegrasi dengan beberapa Layanan AWS, termasuk Elastic Load Balancing AWS Cloud Map,, dan Amazon. EC2 Dengan integrasi yang ketat ini, Amazon ECS mencakup beberapa fitur seperti penyeimbangan beban layanan, penemuan layanan, jaringan tugas, dan penskalaan otomatis cluster. Amazon ECS dan lainnya Layanan AWS yang terintegrasi dengan semua kuota layanan pemeliharaan dan batas tarif API untuk memastikan kinerja dan pemanfaatan yang konsisten. Kuota layanan ini juga mencegah penyediaan sumber daya yang tidak disengaja daripada yang dibutuhkan dan melindungi dari tindakan jahat yang dapat meningkatkan tagihan Anda.

Dengan membiasakan diri dengan kuota layanan dan batas tarif AWS API, Anda dapat merencanakan penskalaan beban kerja Anda tanpa khawatir tentang penurunan kinerja yang tidak terduga. Untuk informasi selengkapnya, lihat kuota layanan Amazon ECS dan Meminta pembatasan untuk Amazon ECS API.

Saat menskalakan beban kerja Anda di Amazon ECS, pertimbangkan kuota layanan berikut. Untuk petunjuk tentang cara meminta peningkatan kuota layanan, lihat Mengelola Amazon ECS dan kuota AWS Fargate layanan di. AWS Management Console

- AWS Fargate memiliki kuota yang membatasi jumlah tugas yang berjalan bersamaan di masing-masing tugas. Wilayah AWS Ada kuota untuk tugas On-Demand dan Fargate Spot di Amazon ECS. Setiap kuota layanan juga mencakup semua pod Amazon EKS yang Anda jalankan di Fargate. Untuk informasi selengkapnya tentang kuota Fargate, lihat kuota layanan di AWS Fargate Panduan Pengguna Layanan Amazon Elastic Container untuk. AWS Fargate
- Untuk tugas yang berjalan di EC2 instans Amazon, jumlah maksimum EC2 instans Amazon yang dapat Anda daftarkan untuk setiap cluster adalah 5.000. Jika Anda menggunakan penskalaan

otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, atau jika Anda mengelola instans EC2 Amazon untuk klaster Anda sendiri, kuota ini mungkin menjadi hambatan penerapan. Jika Anda membutuhkan kapasitas lebih, Anda dapat membuat lebih banyak cluster atau meminta peningkatan kuota layanan.

- Jika Anda menggunakan penskalaan otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, pertimbangkan kuota saat menskalakan layanan Anda. `Tasks in the PROVISIONING state per cluster` Kuota ini adalah jumlah maksimum tugas di `PROVISIONING` negara bagian untuk setiap cluster di mana penyedia kapasitas dapat meningkatkan kapasitas. Ketika Anda meluncurkan sejumlah besar tugas sekaligus, Anda dapat dengan mudah memenuhi kuota ini. Salah satu contohnya adalah jika Anda secara bersamaan menyebarkan puluhan layanan, masing-masing dengan ratusan tugas. Ketika ini terjadi, penyedia kapasitas perlu meluncurkan instance kontainer baru untuk menempatkan tugas ketika cluster memiliki kapasitas yang tidak mencukupi. Sementara penyedia kapasitas meluncurkan EC2 instans Amazon tambahan, penjadwal layanan Amazon ECS kemungkinan akan terus meluncurkan tugas secara paralel. Namun, aktivitas ini mungkin terhambat karena kapasitas cluster yang tidak mencukupi. Penjadwal layanan Amazon ECS mengimplementasikan strategi back-off dan exponential throttling untuk mencoba kembali penempatan tugas saat instance container baru diluncurkan. Akibatnya, Anda mungkin mengalami waktu penerapan atau penskalaan yang lebih lambat. Untuk menghindari situasi ini, Anda dapat merencanakan penerapan layanan Anda dengan salah satu cara berikut. Gunakan sejumlah besar tugas yang tidak memerlukan peningkatan kapasitas cluster, atau pertahankan kapasitas cluster cadangan untuk peluncuran tugas baru.

Selain mempertimbangkan kuota layanan Amazon ECS saat menskalakan beban kerja Anda, pertimbangkan juga kuota layanan untuk yang lain Layanan AWS yang terintegrasi dengan Amazon ECS. Bagian berikut mencakup batas tarif utama untuk setiap layanan secara rinci, dan memberikan rekomendasi untuk menangani potensi masalah pelambatan.

Penyeimbang Beban Elastis

Anda dapat mengonfigurasi layanan Amazon ECS agar menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas. Untuk informasi selengkapnya tentang cara memilih penyeimbang beban, lihat Menggunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Kuota layanan Elastic Load Balancing

Saat Anda menskalakan beban kerja Anda, pertimbangkan kuota layanan Elastic Load Balancing berikut. Sebagian besar kuota layanan Elastic Load Balancing dapat disesuaikan, dan Anda dapat meminta peningkatan konsol Service Quotas.

Penyeimbang Beban Aplikasi

Bila Anda menggunakan Application Load Balancer, tergantung pada kasus penggunaan Anda, Anda mungkin perlu meminta peningkatan kuota untuk:

- `Targets per Application Load Balancer`Kuota yang merupakan jumlah target di belakang Application Load Balancer Anda.
- `Targets per Target Group per Region`Kuota yang merupakan jumlah target di belakang Grup Target Anda.

Untuk informasi selengkapnya, lihat Kuota untuk Penyeimbang Beban Aplikasi Anda.

Penyeimbang Beban Jaringan

Ada batasan yang lebih ketat pada jumlah target yang dapat Anda daftarkan dengan Network Load Balancer. Saat menggunakan Network Load Balancer, Anda sering ingin mengaktifkan dukungan lintas zona, yang dilengkapi dengan batasan penskalaan tambahan pada `Targets per Availability Zone Per Network Load Balancer` jumlah maksimum target per Availability Zone untuk setiap Network Load Balancer. Untuk informasi selengkapnya, lihat Kuota untuk Network Load Balancer Anda.

Pelambatan API Elastic Load Balancing

Saat Anda mengonfigurasi layanan Amazon ECS untuk menggunakan penyeimbang beban, pemeriksaan kesehatan grup target harus lulus sebelum layanan dianggap sehat. Untuk melakukan pemeriksaan kesehatan ini, Amazon ECS memanggil operasi Elastic Load Balancing API atas nama Anda. Jika Anda memiliki sejumlah besar layanan yang dikonfigurasi dengan penyeimbang beban di akun Anda, Anda mungkin mengalami penerapan layanan yang lebih lambat karena potensi pembatasan khusus untuk operasi API Elastic Load Balancing, dan `RegisterTarget` Elastic `DeregisterTarget` Load `DescribeTargetHealth` Balancing. Saat pelambatan terjadi, kesalahan pelambatan terjadi di pesan peristiwa layanan Amazon ECS Anda.

Jika mengalami pelambatan AWS Cloud Map API, Anda dapat menghubungi Dukungan untuk panduan tentang cara meningkatkan batas pelambatan AWS Cloud Map API Anda. Untuk informasi

selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat Menangani masalah pelambatan.

Antarmuka jaringan elastis

Saat tugas Anda menggunakan mode `awsvpc` jaringan, Amazon ECS menyediakan elastic network interface (ENI) yang unik untuk setiap tugas. Jika layanan Amazon ECS Anda menggunakan penyeimbang beban Elastic Load Balancing, antarmuka jaringan ini juga terdaftar sebagai target ke grup target yang sesuai yang ditentukan dalam layanan.

Kuota layanan antarmuka jaringan elastis

Ketika Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, sebuah elastic network interface yang unik dilampirkan ke setiap tugas. Jika tugas-tugas tersebut harus dicapai melalui internet, tetapkan alamat IP publik ke elastic network interface untuk tugas-tugas tersebut. Saat Anda menskalakan beban kerja Amazon ECS Anda, pertimbangkan dua kuota penting ini:

- `Network interfaces per Region`Kuota yang merupakan jumlah maksimum antarmuka jaringan di akun Wilayah AWS Anda.
- `Elastic IP addresses per Region`Kuota yang merupakan jumlah maksimum alamat IP elastis dalam file Wilayah AWS.

Kedua kuota layanan ini dapat disesuaikan dan Anda dapat meminta peningkatan dari konsol Service Quotas Anda untuk ini. Untuk informasi selengkapnya, lihat kuota layanan Amazon VPC.

Untuk beban kerja Amazon ECS yang dihosting di EC2 instans Amazon, saat menjalankan tugas yang menggunakan mode `awsvpc` jaringan, pertimbangkan kuota `Maximum network interfaces` layanan, jumlah maksimum instans jaringan untuk setiap instans Amazon. EC2 Kuota ini membatasi jumlah tugas yang dapat Anda tempatkan pada sebuah instance. Anda tidak dapat menyesuaikan kuota dan tidak tersedia di konsol Service Quotas. Untuk informasi selengkapnya, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan EC2 Pengguna Amazon.

Meskipun Anda tidak dapat mengubah jumlah antarmuka jaringan yang dapat dilampirkan ke EC2 instans Amazon, Anda dapat menggunakan fitur trunking elastic network interface untuk meningkatkan jumlah antarmuka jaringan yang tersedia. Misalnya, secara default sebuah `c5.large` instance dapat memiliki hingga tiga antarmuka jaringan. Antarmuka jaringan utama untuk instance dihitung sebagai satu. Jadi, Anda dapat melampirkan dua antarmuka jaringan tambahan ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukan antarmuka

jaringan, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini. Hal ini dapat menyebabkan kurangnya pemanfaatan kapasitas cluster Anda. Jika Anda mengaktifkan trunking elastic network interface, Anda dapat meningkatkan kepadatan antarmuka jaringan untuk menempatkan lebih banyak tugas pada setiap instance. Dengan trunking dihidupkan, sebuah `c5.large` instance dapat memiliki hingga 12 antarmuka jaringan. Instans memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan "trunk" ke instance. Akibatnya, dengan konfigurasi ini Anda dapat menjalankan 10 tugas pada instance alih-alih dua tugas default. Untuk informasi selengkapnya, lihat Trunking antarmuka jaringan elastis.

Pelambatan API antarmuka jaringan elastis

Saat Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, Amazon ECS bergantung pada Amazon berikut. EC2 APIs Masing-masing APIs memiliki throttle API yang berbeda. Untuk informasi selengkapnya, lihat Meminta pembatasan untuk Amazon EC2 API.

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

Jika panggilan Amazon EC2 API dibatasi selama alur kerja penyediaan elastic network interface, penjadwal layanan Amazon ECS secara otomatis mencoba ulang dengan back-off eksponensial. Percobaan ulang otomatis ini terkadang dapat menyebabkan penundaan dalam meluncurkan tugas, yang menghasilkan kecepatan penerapan yang lebih lambat. Ketika API throttling terjadi, Anda akan melihat pesan `Operations are being throttled. Will try again later.` pada pesan acara layanan Anda. Jika Anda secara konsisten memenuhi throttle Amazon EC2 API, Anda dapat menghubungi Dukungan untuk panduan tentang cara meningkatkan batas pembatasan API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan, lihat Menangani masalah pelambatan.

AWS Cloud Map

Penemuan layanan Amazon ECS digunakan AWS Cloud Map APIs untuk mengelola ruang nama untuk layanan Amazon ECS Anda. Jika layanan Anda memiliki banyak tugas, pertimbangkan rekomendasi berikut. Untuk informasi selengkapnya, lihat pertimbangan penemuan layanan Amazon ECS.

AWS Cloud Map kuota layanan

Ketika layanan Amazon ECS dikonfigurasi untuk menggunakan penemuan layanan, `Tasks per service` kuota yang merupakan jumlah maksimum tugas untuk layanan, dipengaruhi oleh kuota AWS Cloud Map `Instances per service` layanan yang merupakan jumlah maksimum instans untuk layanan tersebut. Secara khusus, kuota AWS Cloud Map layanan mengurangi jumlah tugas yang dapat Anda jalankan hingga paling banyak 1.000 tugas per layanan. Anda tidak dapat mengubah AWS Cloud Map kuota. Untuk informasi selengkapnya, lihat AWS Cloud Map service quotas.

AWS Cloud Map Pelambatan API

Amazon ECS memanggil`ListInstances`,, `GetInstancesHealthStatus``RegisterInstance`, dan `DeregisterInstance` AWS Cloud Map APIs atas nama Anda. Mereka membantu dengan penemuan layanan dan melakukan pemeriksaan kesehatan ketika Anda meluncurkan tugas. Ketika beberapa layanan yang menggunakan penemuan layanan dengan sejumlah besar tugas diterapkan pada saat yang sama, hal ini dapat mengakibatkan melebihi batas pembatasan AWS Cloud Map API. Ketika ini terjadi, Anda mungkin akan melihat pesan berikut: `Operations are being throttled. Will try again later` di pesan acara layanan Amazon ECS Anda dan penerapan yang lebih lambat dan kecepatan peluncuran tugas. AWS Cloud Map tidak mendokumentasikan batas pelambatan untuk ini. APIs Jika Anda mengalami pembatasan dari ini, Anda dapat menghubungi Dukungan untuk panduan tentang peningkatan batas pelambatan API Anda. Untuk rekomendasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat Menangani masalah pelambatan.

Menangani masalah pelambatan Amazon ECS

Kesalahan pelambatan terbagi dalam dua kategori utama: throttling sinkron dan pelambatan asinkron.

Pelambatan sinkron

Saat pelambatan sinkron terjadi, Anda segera menerima respons kesalahan dari Amazon ECS. Kategori ini biasanya terjadi ketika Anda memanggil Amazon ECS APIs saat menjalankan tugas atau membuat layanan. Untuk informasi selengkapnya tentang pembatasan yang terlibat dan batas throttle yang relevan, lihat Meminta pembatasan untuk Amazon ECS API.

Saat aplikasi Anda memulai permintaan API, misalnya, dengan menggunakan AWS CLI atau AWS SDK, Anda dapat memulihkan pelambatan API. Anda dapat melakukan ini dengan merancang aplikasi Anda untuk menangani kesalahan atau dengan menerapkan strategi backoff dan jitter eksponensial dengan logika coba lagi untuk panggilan API. Untuk informasi selengkapnya, lihat Timeout, percobaan ulang, dan backoff dengan jitter.

Jika Anda menggunakan AWS SDK, logika coba ulang otomatis sudah terpasang dan dapat dikonfigurasi.

Pelambatan asinkron

Pelambatan asinkron terjadi karena alur kerja asinkron di mana Amazon ECS atau AWS CloudFormation mungkin memanggil atas nama Anda untuk menyediakan sumber daya. APIs Penting untuk mengetahui mana AWS APIs yang dipanggil Amazon ECS atas nama Anda. Misalnya, `CreateNetworkInterface` API dipanggil untuk tugas yang menggunakan mode `awsvpc` jaringan, dan `DescribeTargetHealth` API dipanggil saat melakukan pemeriksaan kesehatan untuk tugas yang terdaftar ke penyeimbang beban.

Ketika beban kerja Anda mencapai skala yang cukup besar, operasi API ini mungkin akan dibatasi. Artinya, mereka mungkin cukup dibatasi untuk melanggar batas yang diberlakukan oleh Amazon ECS atau Layanan AWS yang sedang disebut. Misalnya, jika Anda menerapkan ratusan layanan, masing-masing memiliki ratusan tugas secara bersamaan yang menggunakan mode `awsvpc` jaringan, Amazon ECS akan memanggil operasi Amazon EC2 API seperti dan operasi Elastic Load `CreateNetworkInterface` Balancing API seperti `RegisterTarget` atau untuk mendaftarkan interface network `DescribeTargetHealth` elastis dan penyeimbang beban. Panggilan API ini dapat melebihi batas API, yang mengakibatkan kesalahan pelambatan. Berikut ini adalah contoh kesalahan pelambatan Elastic Load Balancing yang disertakan dalam pesan acara layanan.

```
{
   "userIdentity":{
      "arn":"arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForECS/ecs-service-
scheduler",
```

```
        "eventTime":"2022-03-21T08:11:24Z",
        "eventSource":"elasticloadbalancing.amazonaws.com",
        "eventName":" DescribeTargetHealth ",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"ecs.amazonaws.com",
        "userAgent":"ecs.amazonaws.com",
        "errorCode":"ThrottlingException",
        "errorMessage":"Rate exceeded",
        "eventID":"0aeb38fc-229b-4912-8b0d-2e8315193e9c"
    }
}
```

Ketika panggilan API ini berbagi batas dengan lalu lintas API lain di akun Anda, mereka mungkin sulit dipantau meskipun dipancarkan sebagai peristiwa layanan.

Pantau pelambatan

Penting untuk mengidentifikasi permintaan API mana yang dibatasi dan siapa yang mengeluarkan permintaan ini. Anda dapat menggunakan monitor AWS CloudTrail mana yang membatasi, dan terintegrasi dengan, Amazon CloudWatch Athena, dan Amazon. EventBridge Anda dapat mengonfigurasi CloudTrail untuk mengirim peristiwa tertentu ke CloudWatch Log. CloudWatch Wawasan log log mengurai dan menganalisis peristiwa. Ini mengidentifikasi detail dalam peristiwa pelambatan seperti peran pengguna atau IAM yang melakukan panggilan dan jumlah panggilan API yang dilakukan. Untuk informasi selengkapnya, lihat Memantau file CloudTrail CloudWatch log dengan Log.

Untuk informasi selengkapnya tentang wawasan dan petunjuk CloudWatch Log tentang cara menanyakan file log, lihat Menganalisis data CloudWatch log dengan Wawasan Log.

Dengan Amazon Athena, Anda dapat membuat kueri dan menganalisis data menggunakan SQL standar. Misalnya, Anda dapat membuat tabel Athena untuk mengurai CloudTrail acara. Untuk informasi selengkapnya, lihat Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk CloudTrail log.

Setelah membuat tabel Athena, Anda dapat menggunakan kueri SQL seperti yang berikut untuk menyelidiki kesalahan. `ThrottlingException`

Ganti *`user-input`* dengan nilai-nilai Anda.

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
```

```
FROM cloudtrail_table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2024-09-24T00:00:08Z' and '2024-09-23T23:15:08Z'
group by errorcode, awsregion, eventsource, useragent, eventname
order by count desc;
```

Amazon ECS juga memancarkan pemberitahuan acara ke Amazon. EventBridge Ada peristiwa perubahan status sumber daya dan peristiwa tindakan layanan. Mereka termasuk peristiwa pelambatan API seperti `ECS_OPERATION_THROTTLED` dan. `SERVICE_DISCOVERY_OPERATION_THROTTLED` Untuk informasi selengkapnya, lihat Acara tindakan layanan Amazon ECS.

Peristiwa ini dapat dikonsumsi oleh layanan seperti AWS Lambda untuk melakukan tindakan sebagai tanggapan. Untuk informasi selengkapnya, lihat Menangani acara Amazon ECS.

Jika Anda menjalankan tugas mandiri, beberapa operasi API seperti `RunTask` asinkron, dan operasi coba lagi tidak dilakukan secara otomatis. Dalam kasus seperti itu, Anda dapat menggunakan layanan seperti AWS Step Functions EventBridge integrasi untuk mencoba kembali operasi yang dibatasi atau gagal. Untuk informasi selengkapnya, lihat Mengelola tugas penampung (Amazon ECS, Amazon SNS).

Gunakan CloudWatch untuk memonitor throttling

CloudWatch menawarkan pemantauan penggunaan API pada `Usage` namespace di bawah By AWS Resource. Metrik ini dicatat dengan API tipe dan nama CallCountmetrik. Anda dapat membuat alarm untuk memulai setiap kali metrik ini mencapai ambang batas tertentu. Untuk informasi selengkapnya, lihat Memvisualisasikan kuota layanan Anda dan menyetel alarm.

CloudWatch juga menawarkan deteksi anomali. Fitur ini menggunakan pembelajaran mesin untuk menganalisis dan menetapkan garis dasar berdasarkan perilaku tertentu dari metrik yang Anda aktifkan. Jika ada aktivitas API yang tidak biasa, Anda dapat menggunakan fitur ini bersama dengan CloudWatch alarm. Untuk informasi lebih lanjut, lihat Menggunakan CloudWatch deteksi anomali.

Dengan secara proaktif memantau kesalahan pelambatan, Anda dapat menghubungi Dukungan untuk meningkatkan batas pelambatan yang relevan dan juga menerima panduan untuk kebutuhan aplikasi unik Anda.

Praktik terbaik penskalaan dan manajemen kapasitas Amazon ECS Auto

Anda dapat menjalankan beban kerja aplikasi kontainer dari semua ukuran di Amazon ECS. Ini termasuk lingkungan pengujian minimal dan lingkungan produksi besar yang beroperasi pada skala global.

Dengan Amazon ECS, seperti semua AWS layanan, Anda hanya membayar untuk apa yang Anda gunakan. Ketika Anda merancang aplikasi Anda dengan tepat, Anda dapat menghemat biaya dengan hanya mengkonsumsi sumber daya yang Anda butuhkan saat Anda membutuhkannya.

Rekomendasi berikut menunjukkan kepada Anda cara menjalankan beban kerja Amazon ECS Anda untuk memenuhi tujuan tingkat layanan Anda saat beroperasi dengan hemat biaya.

Topik

- Menentukan ukuran tugas untuk Amazon ECS
- Mengoptimalkan penskalaan otomatis layanan Amazon ECS
- Kapasitas dan ketersediaan Amazon ECS
- Kapasitas kluster Amazon ECS
- Memilih ukuran tugas Fargate untuk Amazon ECS
- Mempercepat penyediaan kapasitas cluster Amazon ECS dengan penyedia kapasitas di Amazon EC2

Menentukan ukuran tugas untuk Amazon ECS

Salah satu pilihan terpenting saat Anda menerapkan kontainer di Amazon ECS adalah wadah dan ukuran tugas Anda. Ukuran wadah dan tugas Anda sangat penting untuk penskalaan dan perencanaan kapasitas.

Amazon ECS menggunakan dua metrik sumber daya untuk kapasitas: CPU dan memori. Amazon ECS mengukur CPU dalam satuan 1/1024 dari vCPU penuh (di mana 1024 unit sama dengan 1 vCPU utuh). Amazon ECS mengukur memori dalam megabyte.

Dalam definisi tugas Anda, Anda dapat mendeklarasikan reservasi dan batasan sumber daya.

Saat Anda mendeklarasikan reservasi, Anda menyatakan jumlah minimum sumber daya yang dibutuhkan suatu tugas. Tugas Anda menerima setidaknya jumlah sumber daya yang Anda minta.

Aplikasi Anda mungkin dapat menggunakan lebih banyak CPU atau memori daripada reservasi yang Anda deklarasikan. Namun, ini tunduk pada batasan apa pun yang juga Anda nyatakan.

Menggunakan lebih dari jumlah reservasi dikenal sebagai bursting. Meledak berarti aplikasi Anda menggunakan lebih banyak sumber daya daripada yang Anda pesan tetapi tetap dalam batas yang Anda deklarasikan. Amazon ECS menjamin reservasi. Misalnya, jika Anda menggunakan EC2 instans Amazon untuk menyediakan kapasitas, Amazon ECS tidak menempatkan tugas pada instans yang tidak dapat memenuhi reservasi.

Batas adalah jumlah maksimum unit CPU atau memori yang dapat digunakan oleh wadah atau tugas Anda. Jika kontainer Anda mencoba menggunakan lebih banyak CPU daripada batas ini, Amazon ECS melambatnya. Jika penampung Anda mencoba menggunakan lebih banyak memori daripada batas ini, Amazon ECS menghentikan penampung Anda.

Memilih nilai-nilai ini bisa menjadi tantangan. Nilai yang paling sesuai untuk aplikasi Anda sangat bergantung pada kebutuhan sumber daya aplikasi Anda.

Pengujian beban aplikasi Anda adalah kunci keberhasilan perencanaan kebutuhan sumber daya. Pengujian beban membantu Anda lebih memahami persyaratan aplikasi Anda.

Aplikasi tanpa kewarganegaraan

Untuk aplikasi stateless yang menskalakan secara horizontal, seperti aplikasi di belakang penyeimbang beban, sebaiknya Anda terlebih dahulu menentukan berapa banyak memori yang dikonsumsi aplikasi Anda saat melayani permintaan.

Untuk melakukan ini, Anda dapat menggunakan alat tradisional seperti `ps` atau`top`. Anda juga dapat menggunakan solusi pemantauan seperti CloudWatch Wawasan Kontainer.

Saat Anda menentukan reservasi CPU, pertimbangkan bagaimana Anda ingin menskalakan aplikasi Anda untuk memenuhi persyaratan bisnis Anda.

Anda dapat menggunakan reservasi CPU yang lebih kecil, seperti 256 unit CPU (atau 1/4 vCPU), untuk meningkatkan skala dengan cara halus yang meminimalkan biaya. Tetapi mereka mungkin tidak berskala cukup cepat untuk memenuhi lonjakan permintaan yang signifikan.

Anda dapat menggunakan reservasi CPU yang lebih besar untuk meningkatkan skala masuk dan keluar lebih cepat. Ini membantu Anda mencocokkan lonjakan permintaan lebih cepat. Namun, pemesanan CPU yang lebih besar harganya lebih mahal.

Aplikasi lainnya

Untuk aplikasi yang tidak menskalakan secara horizontal, seperti pekerja tunggal atau server database, kapasitas dan biaya yang tersedia adalah pertimbangan terpenting Anda.

Pilih jumlah memori dan CPU berdasarkan pengujian beban yang menunjukkan bahwa Anda perlu melayani lalu lintas dan memenuhi tujuan tingkat layanan Anda. Amazon ECS memastikan bahwa aplikasi Anda ditempatkan pada host yang memiliki kapasitas yang memadai.

Mengoptimalkan penskalaan otomatis layanan Amazon ECS

Layanan Amazon ECS adalah kumpulan tugas yang dikelola. Setiap layanan memiliki definisi tugas terkait, jumlah tugas yang diinginkan, dan strategi penempatan opsional.

Penskalaan otomatis layanan Amazon ECS berfungsi melalui layanan Application Auto Scaling. Application Auto Scaling menggunakan CloudWatch metrik sebagai sumber untuk menskalakan metrik. Ini juga menggunakan CloudWatch alarm untuk menetapkan ambang batas kapan harus menskalakan layanan Anda masuk atau keluar.

Anda memberikan ambang batas untuk penskalaan. Anda dapat menetapkan target metrik, yang disebut penskalaan pelacakan target. Anda juga dapat menentukan ambang batas, yang disebut penskalaan langkah.

Setelah Anda mengonfigurasi Application Auto Scaling, Application Auto Scaling terus menghitung jumlah tugas yang diinginkan untuk layanan. Ini juga memberi tahu Amazon ECS ketika jumlah tugas yang diinginkan harus berubah, baik dengan menskalakannya atau menskalakannya.

Untuk menggunakan penskalaan otomatis servis secara efektif, Anda harus memilih metrik penskalaan yang sesuai. Kami membahas cara memilih metrik di bagian berikut.

Karakterisasi aplikasi Anda

Menskalakan aplikasi dengan benar mengharuskan Anda mengetahui kondisi di mana Anda harus menskalakan aplikasi Anda dan kapan Anda harus menskalakannya.

Intinya, Anda harus memperkecil aplikasi Anda jika permintaan diperkirakan melebihi kapasitas. Sebaliknya, Anda dapat menskalakan aplikasi Anda untuk menghemat biaya ketika sumber daya melebihi permintaan.

Mengidentifikasi metrik pemanfaatan

Untuk menskalakan secara efektif, Anda harus mengidentifikasi metrik yang menunjukkan pemanfaatan atau saturasi. Metrik ini harus menunjukkan properti berikut agar berguna untuk penskalaan.

- Metrik harus berkorelasi dengan permintaan. Ketika Anda mempertahankan sumber daya tetap stabil tetapi permintaan berubah, nilai metrik juga harus berubah. Metrik harus meningkat atau menurun ketika permintaan meningkat atau menurun.
- Nilai metrik harus diskalakan sebanding dengan kapasitas. Ketika permintaan tetap konstan, menambahkan lebih banyak sumber daya harus menghasilkan perubahan proporsional dalam nilai metrik. Jadi, menggandakan jumlah tugas akan menyebabkan metrik berkurang 50%.

Cara terbaik untuk mengidentifikasi metrik pemanfaatan adalah melalui pengujian beban di lingkungan pra-produksi seperti lingkungan pementasan. Solusi pengujian beban komersial dan sumber terbuka tersedia secara luas. Solusi ini biasanya dapat menghasilkan beban sintetis atau mensimulasikan lalu lintas pengguna nyata.

Untuk memulai proses pengujian beban, Anda harus terlebih dahulu membuat dasbor untuk metrik pemanfaatan aplikasi Anda. Metrik ini termasuk pemanfaatan CPU, pemanfaatan memori, I/O operasi, kedalaman I/O antrian, dan throughput jaringan. Anda dapat mengumpulkan metrik ini dengan layanan seperti CloudWatch Wawasan Kontainer. Anda juga dapat mengumpulkannya dengan menggunakan Amazon Managed Service untuk Prometheus bersama dengan Amazon Managed Grafana. Selama proses ini, pastikan Anda mengumpulkan dan memplot metrik untuk waktu respons aplikasi atau tingkat penyelesaian pekerjaan Anda.

Saat Anda memuat tes, mulailah dengan permintaan kecil atau tingkat penyisipan pekerjaan. Pertahankan kecepatan ini stabil selama beberapa menit untuk memungkinkan aplikasi Anda memanas. Kemudian, perlahan-lahan tingkatkan laju dan tahan selama beberapa menit. Ulangi siklus ini, tingkatkan laju setiap kali hingga waktu respons atau penyelesaian aplikasi Anda terlalu lambat untuk memenuhi tujuan tingkat layanan Anda ()SLOs.

Saat Anda memuat tes, periksa masing-masing metrik pemanfaatan. Metrik yang meningkat seiring dengan beban adalah kandidat teratas untuk dijadikan metrik pemanfaatan terbaik Anda.

Selanjutnya, identifikasi sumber daya yang mencapai saturasi. Pada saat yang sama, periksa juga metrik pemanfaatan untuk melihat mana yang rata pada tingkat tinggi terlebih dahulu. Atau, periksa mana yang mencapai puncak dan kemudian crash aplikasi Anda terlebih dahulu. Misalnya, jika pemanfaatan CPU meningkat dari 0% menjadi 70-80% saat Anda menambahkan beban, maka tetap

pada tingkat itu setelah Anda menambahkan lebih banyak beban, maka aman untuk mengatakan bahwa CPU jenuh. Tergantung pada arsitektur CPU, mungkin tidak akan pernah mencapai 100%. Misalnya, asumsikan bahwa pemanfaatan memori meningkat saat Anda menambahkan beban, dan kemudian aplikasi Anda tiba-tiba macet ketika mencapai tugas atau batas memori EC2 instans Amazon. Dalam situasi ini, kemungkinan memori telah dikonsumsi sepenuhnya. Beberapa sumber daya dapat dikonsumsi oleh aplikasi Anda. Oleh karena itu, pilih metrik yang mewakili sumber daya yang habis terlebih dahulu.

Terakhir, coba muat pengujian lagi setelah Anda menggandakan jumlah tugas atau EC2 instans Amazon. Asumsikan bahwa metrik kunci meningkat, atau menurun, pada setengah tingkat seperti sebelumnya. Jika ini masalahnya, maka metrik sebanding dengan kapasitas. Ini adalah metrik pemanfaatan yang baik untuk penskalaan otomatis.

Sekarang pertimbangkan skenario hipotetis ini. Misalkan Anda memuat uji aplikasi dan menemukan bahwa pemanfaatan CPU akhirnya mencapai 80% pada 100 permintaan per detik. Ketika Anda menambahkan lebih banyak beban, itu tidak membuat pemanfaatan CPU meningkat lagi. Namun, itu membuat aplikasi Anda merespons lebih lambat. Kemudian, Anda menjalankan uji beban lagi, menggandakan jumlah tugas tetapi menahan laju pada nilai puncak sebelumnya. Jika Anda menemukan penggunaan CPU rata-rata turun menjadi sekitar 40%, maka pemanfaatan CPU rata-rata adalah kandidat yang baik untuk metrik penskalaan. Di sisi lain, jika pemanfaatan CPU tetap pada 80% setelah meningkatkan jumlah tugas, maka pemanfaatan CPU rata-rata bukanlah metrik penskalaan yang baik. Dalam hal ini, Anda memerlukan lebih banyak penelitian untuk menemukan metrik yang sesuai.

Model aplikasi umum dan properti penskalaan

Anda dapat menjalankan perangkat lunak dari semua jenis pada AWS. Banyak beban kerja yang homegrown, sedangkan yang lain didasarkan pada perangkat lunak open-source populer. Terlepas dari mana asalnya, kami telah mengamati beberapa pola desain umum untuk layanan. Bagaimana Anda menskalakan secara efektif sebagian besar tergantung pada pola.

Server CPU-bound yang efisien

Server CPU-bound yang efisien hampir tidak menggunakan sumber daya selain CPU dan throughput jaringan. Setiap permintaan dapat ditangani oleh aplikasi saja. Permintaan tidak bergantung pada layanan lain seperti database. Aplikasi ini dapat menangani ratusan ribu permintaan bersamaan, dan secara efisien dapat memanfaatkan beberapa CPUs untuk melakukannya. Setiap permintaan dilayani oleh thread khusus dengan overhead memori rendah, atau ada loop peristiwa asinkron yang berjalan pada setiap CPU yang diminta layanan. Setiap replika aplikasi sama-sama mampu

menangani permintaan. Satu-satunya sumber daya yang mungkin habis sebelum CPU adalah bandwidth jaringan. Dalam layanan batas CPU, pemanfaatan memori, bahkan pada throughput puncak, adalah sebagian kecil dari sumber daya yang tersedia.

Anda dapat menggunakan penskalaan otomatis berbasis CPU untuk jenis aplikasi ini. Aplikasi ini menikmati fleksibilitas maksimum dalam hal penskalaan. Anda dapat menskalakannya secara vertikal dengan menyediakan EC2 instance Amazon yang lebih besar atau Fargate v untuk itu. CPUs Dan, Anda juga dapat menskalakannya secara horizontal dengan menambahkan lebih banyak replika. Menambahkan lebih banyak replika, atau menggandakan ukuran instans, memotong rata-rata pemanfaatan CPU relatif terhadap kapasitas hingga setengahnya.

Jika Anda menggunakan EC2 kapasitas Amazon untuk aplikasi ini, pertimbangkan untuk menempatkannya pada instans yang dioptimalkan komputasi seperti atau keluarga. `c5` `c6g`

Server terikat memori yang efisien

Server yang terikat memori yang efisien mengalokasikan sejumlah besar memori per permintaan. Pada konkurensi maksimum, tetapi belum tentu throughput, memori habis sebelum sumber daya CPU habis. Memori yang terkait dengan permintaan dibebaskan saat permintaan berakhir. Permintaan tambahan dapat diterima selama ada memori yang tersedia.

Anda dapat menggunakan penskalaan otomatis berbasis memori untuk jenis aplikasi ini. Aplikasi ini menikmati fleksibilitas maksimum dalam hal penskalaan. Anda dapat menskalakannya secara vertikal dengan menyediakan sumber daya memori Amazon EC2 atau Fargate yang lebih besar untuknya. Dan, Anda juga dapat menskalakannya secara horizontal dengan menambahkan lebih banyak replika. Menambahkan lebih banyak replika, atau menggandakan ukuran instans, dapat memotong pemanfaatan memori rata-rata relatif terhadap kapasitas hingga setengahnya.

Jika Anda menggunakan EC2 kapasitas Amazon untuk aplikasi ini, pertimbangkan untuk menempatkannya pada instance yang dioptimalkan memori seperti atau keluarga. `r5` `r6g`

Beberapa aplikasi yang terikat memori tidak membebaskan memori yang terkait dengan permintaan ketika itu berakhir, sehingga pengurangan konkurensi tidak menghasilkan pengurangan memori yang digunakan. Untuk ini, kami tidak menyarankan Anda menggunakan penskalaan berbasis memori.

Server berbasis pekerja

Server berbasis pekerja memproses satu permintaan untuk setiap thread pekerja individu satu demi satu. Benang pekerja dapat berupa benang ringan, seperti utas POSIX. Mereka juga bisa menjadi benang yang lebih berat, seperti proses UNIX. Apa pun utas mereka, selalu ada konkurensi maksimum yang dapat didukung aplikasi. Biasanya batas konkurensi diatur secara proporsional

dengan sumber daya memori yang tersedia. Jika batas konkurensi tercapai, aplikasi menempatkan permintaan tambahan ke dalam antrean backlog. Jika antrian backlog meluap, aplikasi segera menolak permintaan masuk tambahan. Aplikasi umum yang sesuai dengan pola ini termasuk server web Apache dan Gunicorn.

Request concurrency biasanya merupakan metrik terbaik untuk menskalakan aplikasi ini. Karena ada batas konkurensi untuk setiap replika, penting untuk menskalakan sebelum batas rata-rata tercapai.

Cara terbaik untuk mendapatkan metrik konkurensi permintaan adalah meminta aplikasi Anda melaporkannya. CloudWatch Setiap replika aplikasi Anda dapat mempublikasikan jumlah permintaan bersamaan sebagai metrik kustom pada frekuensi tinggi. Sebaiknya frekuensinya diatur setidaknya sekali setiap menit. Setelah beberapa laporan dikumpulkan, Anda dapat menggunakan konkurensi rata-rata sebagai metrik penskalaan. Anda menghitung metrik ini dengan mengambil konkurensi total dan membaginya dengan jumlah replika. Misalnya, jika total konkurensi adalah 1000 dan jumlah replika adalah 10, maka konkurensi rata-rata adalah 100.

Jika aplikasi Anda berada di belakang Application Load Balancer, Anda juga dapat menggunakan `ActiveConnectionCount` metrik untuk penyeimbang beban sebagai faktor dalam metrik penskalaan. Anda harus membagi `ActiveConnectionCount` metrik dengan jumlah replika untuk mendapatkan nilai rata-rata. Anda harus menggunakan nilai rata-rata untuk penskalaan, sebagai lawan dari nilai hitungan mentah.

Agar desain ini bekerja paling baik, standar deviasi latensi respons harus kecil pada tingkat permintaan rendah. Kami merekomendasikan bahwa, selama periode permintaan rendah, sebagian besar permintaan dijawab dalam waktu singkat, dan tidak ada banyak permintaan yang membutuhkan waktu lebih lama daripada waktu rata-rata untuk merespons. Waktu respons rata-rata harus mendekati waktu respons persentil ke-95. Jika tidak, luapan antrian mungkin terjadi sebagai hasilnya. Ini mengarah pada kesalahan. Kami menyarankan Anda memberikan replika tambahan jika diperlukan untuk mengurangi risiko meluap.

Server yang menunggu

Server menunggu melakukan beberapa pemrosesan untuk setiap permintaan, tetapi sangat tergantung pada satu atau lebih layanan hilir untuk berfungsi. Aplikasi kontainer sering menggunakan layanan hilir seperti database dan layanan API lainnya. Butuh beberapa waktu bagi layanan ini untuk merespons, terutama dalam skenario berkapasitas tinggi atau konkurensi tinggi. Ini karena aplikasi ini cenderung menggunakan beberapa sumber daya CPU dan memanfaatkan konkurensi maksimumnya dalam hal memori yang tersedia.

Layanan tunggu cocok baik dalam pola server terikat memori atau pola server berbasis pekerja, tergantung pada bagaimana aplikasi dirancang. Jika konkurensi aplikasi hanya dibatasi oleh memori, maka pemanfaatan memori rata-rata harus digunakan sebagai metrik penskalaan. Jika konkurensi aplikasi didasarkan pada batas pekerja, maka konkurensi rata-rata harus digunakan sebagai metrik penskalaan.

Server berbasis Java

Jika server berbasis Java Anda terikat CPU dan skala secara proporsional dengan sumber daya CPU, maka mungkin cocok untuk pola server terikat CPU yang efisien. Jika demikian, penggunaan CPU rata-rata mungkin sesuai sebagai metrik penskalaan. Namun, banyak aplikasi Java tidak terikat CPU, membuatnya sulit untuk diskalakan.

Untuk kinerja terbaik, kami menyarankan Anda mengalokasikan memori sebanyak mungkin ke tumpukan Java Virtual Machine (JVM). Versi terbaru dari JVM, termasuk pembaruan Java 8 191 atau yang lebih baru, secara otomatis mengatur ukuran tumpukan sebesar mungkin agar muat di dalam wadah. Ini berarti bahwa, di Jawa, pemanfaatan memori jarang sebanding dengan pemanfaatan aplikasi. Ketika tingkat permintaan dan konkurensi meningkat, pemanfaatan memori tetap konstan. Karena itu, kami tidak menyarankan penskalaan server berbasis Java berdasarkan pemanfaatan memori. Sebagai gantinya, kami biasanya merekomendasikan penskalaan pada pemanfaatan CPU.

Dalam beberapa kasus, server berbasis Java mengalami banyak kelelahan sebelum menghabiskan CPU. Jika aplikasi Anda rentan terhadap kelelahan heap pada konkurensi tinggi, maka koneksi rata-rata adalah metrik penskalaan terbaik. Jika aplikasi Anda rentan terhadap heap exhaustion pada throughput tinggi, maka tingkat permintaan rata-rata adalah metrik penskalaan terbaik.

Server yang menggunakan runtime yang dikumpulkan sampah lainnya

Banyak aplikasi server didasarkan pada runtime yang melakukan pengumpulan sampah seperti .NET dan Ruby. Aplikasi server ini mungkin cocok dengan salah satu pola yang dijelaskan sebelumnya. Namun, seperti halnya Java, kami tidak merekomendasikan penskalaan aplikasi ini berdasarkan memori, karena pemanfaatan memori rata-rata yang diamati seringkali tidak berkorelasi dengan throughput atau konkurensi.

Untuk aplikasi ini, kami menyarankan Anda menskalakan pemanfaatan CPU jika aplikasi terikat CPU. Jika tidak, kami menyarankan Anda menskalakan throughput rata-rata atau konkurensi rata-rata, berdasarkan hasil pengujian beban Anda.

Prosesor Job

Banyak beban kerja melibatkan pemrosesan pekerjaan asinkron. Mereka termasuk aplikasi yang tidak menerima permintaan secara real time, tetapi berlangganan antrian kerja untuk menerima pekerjaan. Untuk jenis aplikasi ini, metrik penskalaan yang tepat hampir selalu kedalaman antrian. Pertumbuhan antrian merupakan indikasi bahwa pekerjaan yang tertunda melebihi kapasitas pemrosesan, sedangkan antrian kosong menunjukkan bahwa ada lebih banyak kapasitas daripada pekerjaan yang harus dilakukan.

AWS layanan pesan, seperti Amazon SQS dan Amazon Kinesis Data Streams, CloudWatch menyediakan metrik yang dapat digunakan untuk penskalaan. Untuk Amazon SQS, `ApproximateNumberOfMessagesVisible` adalah metrik terbaik. Untuk Kinesis Data Streams, pertimbangkan untuk `MillisBehindLatest` menggunakan metrik, yang diterbitkan oleh Kinesis Client Library (KCL). Metrik ini harus dirata-ratakan di semua konsumen sebelum menggunakannya untuk penskalaan.

Kapasitas dan ketersediaan Amazon ECS

Ketersediaan aplikasi sangat penting untuk memberikan pengalaman bebas kesalahan dan untuk meminimalkan latensi aplikasi. Ketersediaan tergantung pada memiliki sumber daya yang dapat diakses dan memiliki kapasitas yang cukup untuk memenuhi permintaan. AWS menyediakan beberapa mekanisme untuk mengelola ketersediaan. Untuk aplikasi yang Anda host di Amazon ECS, ini termasuk penskalaan otomatis dan Availability Zones (). AZs Penskalaan otomatis mengelola jumlah tugas atau instance berdasarkan metrik yang Anda tentukan, sementara Availability Zone memungkinkan Anda untuk meng-host aplikasi Anda di lokasi yang terisolasi namun dekat secara geografis.

Seperti halnya ukuran tugas, kapasitas dan ketersediaan menyajikan trade-off tertentu yang harus Anda pertimbangkan. Idealnya, kapasitas akan sangat selaras dengan permintaan. Akan selalu ada kapasitas yang cukup untuk melayani permintaan dan memproses pekerjaan untuk memenuhi Tujuan Tingkat Layanan (SLOs) termasuk latensi rendah dan tingkat kesalahan. Kapasitas tidak akan pernah terlalu tinggi, menyebabkan biaya yang berlebihan; juga tidak akan pernah terlalu rendah, yang mengarah ke latensi tinggi dan tingkat kesalahan.

Autoscaling adalah proses laten. Pertama, CloudWatch harus menerima metrik real-time. Kemudian, CloudWatch perlu menggabungkan mereka untuk analisis, yang dapat memakan waktu hingga beberapa menit tergantung pada granularitas metrik. CloudWatch membandingkan metrik dengan ambang alarm untuk mengidentifikasi kekurangan atau kelebihan sumber daya. Untuk

mencegah ketidakstabilan, Anda harus mengonfigurasi alarm agar ambang batas yang ditetapkan dilintasi selama beberapa menit sebelum alarm berbunyi. Hal ini juga membutuhkan waktu untuk menyediakan tugas-tugas baru dan untuk mengakhiri tugas-tugas yang tidak lagi diperlukan.

Karena potensi keterlambatan dalam sistem ini, Anda harus mempertahankan beberapa ruang kepala dengan penyediaan berlebihan. Penyediaan berlebih dapat membantu mengakomodasi ledakan permintaan jangka pendek. Ini juga membantu permintaan tambahan layanan aplikasi Anda tanpa mencapai saturasi. Sebagai praktik yang baik, Anda dapat menetapkan target penskalaan Anda antara 60-80% pemanfaatan. Ini membantu aplikasi Anda menangani semburan permintaan ekstra dengan lebih baik sementara kapasitas tambahan masih dalam proses penyediaan.

Alasan lain kami menyarankan Anda untuk menyediakan berlebihan adalah agar Anda dapat dengan cepat menanggapi kegagalan Availability Zone. AWS merekomendasikan agar beban kerja produksi dilayani dari beberapa Availability Zone. Ini karena, jika terjadi kegagalan Availability Zone, tugas Anda yang berjalan di Availability Zone yang tersisa masih dapat memenuhi permintaan. Jika aplikasi Anda berjalan di dua Availability Zones, Anda perlu menggandakan jumlah tugas normal Anda. Ini agar Anda dapat memberikan kapasitas langsung selama potensi kegagalan. Jika aplikasi Anda berjalan di tiga Availability Zones, kami sarankan Anda menjalankan 1,5 kali jumlah tugas normal Anda. Artinya, jalankan tiga tugas untuk setiap dua tugas yang dibutuhkan untuk penyajian biasa.

Memaksimalkan kecepatan penskalaan

Autoscaling adalah proses reaktif yang membutuhkan waktu untuk diterapkan. Namun, ada beberapa cara untuk membantu meminimalkan waktu yang diperlukan untuk meningkatkan skala.

Minimalkan ukuran gambar. Gambar yang lebih besar membutuhkan waktu lebih lama untuk diunduh dari repositori gambar dan membongkar. Oleh karena itu, menjaga ukuran gambar lebih kecil mengurangi jumlah waktu yang dibutuhkan wadah untuk memulai. Untuk mengurangi ukuran gambar, Anda dapat mengikuti rekomendasi spesifik ini:

- Jika Anda dapat membangun biner statis atau menggunakan Golang, buat `FROM` goresan gambar Anda dan sertakan hanya aplikasi biner Anda dalam gambar yang dihasilkan.
- Gunakan gambar dasar yang diminimalkan dari vendor distro hulu, seperti Amazon Linux atau Ubuntu.
- Jangan sertakan artefak build apa pun di gambar akhir Anda. Menggunakan build multi-tahap dapat membantu dalam hal ini.
- `RUN`Tahapan kompak sedapat mungkin. Setiap `RUN` tahap menciptakan layer gambar baru, yang mengarah ke perjalanan pulang pergi tambahan untuk mengunduh layer. Satu `RUN` tahap yang

memiliki beberapa perintah bergabung dengan && memiliki lebih sedikit lapisan daripada satu dengan beberapa RUN tahap.

- Jika Anda ingin menyertakan data, seperti data inferensi ML, dalam gambar akhir Anda, sertakan hanya data yang diperlukan untuk memulai dan mulai melayani lalu lintas. Jika Anda mengambil data sesuai permintaan dari Amazon S3 atau penyimpanan lain tanpa memengaruhi layanan, maka simpan data Anda di tempat tersebut.

Jaga agar gambar Anda tetap dekat. Semakin tinggi latensi jaringan, semakin lama waktu yang dibutuhkan untuk mengunduh gambar. Host gambar Anda di repositori di AWS Wilayah yang sama dengan beban kerja Anda. Amazon ECR adalah repositori gambar berkinerja tinggi yang tersedia di setiap Wilayah tempat Amazon ECS tersedia. Hindari melintasi Internet atau tautan VPN untuk mengunduh gambar kontainer. Hosting gambar Anda di Wilayah yang sama meningkatkan keandalan secara keseluruhan. Ini mengurangi risiko masalah konektivitas jaringan dan masalah ketersediaan di Wilayah yang berbeda. Atau, Anda juga dapat menerapkan replikasi lintas wilayah Amazon ECR untuk membantu hal ini.

Kurangi ambang batas pemeriksaan kesehatan penyeimbang beban. Load balancer melakukan pemeriksaan kesehatan sebelum mengirim lalu lintas ke aplikasi Anda. Konfigurasi pemeriksaan kesehatan default untuk grup target dapat memakan waktu 90 detik atau lebih lama. Selama ini, penyeimbang beban memeriksa status kesehatan dan menerima permintaan. Menurunkan interval pemeriksaan kesehatan dan jumlah ambang batas dapat membuat aplikasi Anda menerima lalu lintas lebih cepat dan mengurangi beban pada tugas lain.

Pertimbangkan kinerja start dingin. Beberapa aplikasi menggunakan runtime seperti kompilasi Java perform Just-In-Time (JIT). Proses kompilasi setidaknya saat dimulai dapat menunjukkan kinerja aplikasi. Solusinya adalah menulis ulang bagian penting latensi dari beban kerja Anda dalam bahasa yang tidak memberlakukan penalti kinerja awal yang dingin.

Gunakan penskalaan langkah, bukan kebijakan penskalaan pelacakan target. Anda memiliki beberapa opsi Application Auto Scaling untuk tugas Amazon ECS. Pelacakan target adalah mode termudah untuk digunakan. Dengan itu, yang perlu Anda lakukan adalah menetapkan nilai target untuk metrik, seperti pemanfaatan rata-rata CPU. Kemudian, auto scaler secara otomatis mengelola jumlah tugas yang diperlukan untuk mencapai nilai itu. Dengan penskalaan langkah, Anda dapat bereaksi lebih cepat terhadap perubahan permintaan, karena Anda menentukan ambang batas tertentu untuk metrik penskalaan Anda, dan berapa banyak tugas yang harus ditambahkan atau dihapus saat ambang batas dilintasi. Dan, yang lebih penting, Anda dapat bereaksi sangat cepat terhadap perubahan permintaan dengan meminimalkan jumlah waktu alarm ambang batas dilanggar.

Untuk informasi lebih lanjut, lihat Service Auto Scaling dalam Panduan Pengembang Layanan Amazon Elastic Container.

Jika Anda menggunakan EC2 instans Amazon untuk menyediakan kapasitas klaster, pertimbangkan rekomendasi berikut:

Gunakan EC2 instans Amazon yang lebih besar dan volume Amazon EBS yang lebih cepat. Anda dapat meningkatkan kecepatan pengunduhan dan persiapan gambar dengan menggunakan EC2 instans Amazon yang lebih besar dan volume Amazon EBS yang lebih cepat. Dalam rangkaian EC2 instans Amazon tertentu, throughput maksimum jaringan dan Amazon EBS meningkat seiring dengan peningkatan ukuran instans (misalnya, dari `m5.xlarge` kem`5.2xlarge`). Selain itu, Anda juga dapat menyesuaikan volume Amazon EBS untuk meningkatkan throughput dan IOPS mereka. Misalnya, jika Anda menggunakan `gp2` volume, gunakan volume yang lebih besar yang menawarkan lebih banyak throughput dasar. Jika Anda menggunakan `gp3` volume, tentukan throughput dan IOPS saat Anda membuat volume.

Gunakan mode jaringan jembatan untuk tugas yang berjalan di EC2 instans Amazon. Tugas yang menggunakan mode `bridge` jaringan di Amazon EC2 dimulai lebih cepat daripada tugas yang menggunakan mode `awsvpc` jaringan. Saat mode `awsvpc` jaringan digunakan, Amazon ECS melampirkan elastic network interface (ENI) ke instance sebelum meluncurkan tugas. Ini memperkenalkan latensi tambahan. Ada beberapa pengorbanan untuk menggunakan jaringan jembatan. Tugas-tugas ini tidak mendapatkan grup keamanan mereka sendiri, dan ada beberapa implikasi untuk load balancing. Untuk informasi selengkapnya, lihat Grup target penyeimbang beban di Panduan Pengguna Elastic Load Balancing.

Menangani guncangan permintaan

Beberapa aplikasi mengalami guncangan besar yang tiba-tiba dalam permintaan. Ini terjadi karena berbagai alasan: acara berita, penjualan besar, acara media, atau peristiwa lain yang menjadi viral dan menyebabkan lalu lintas meningkat dengan cepat dan signifikan dalam rentang waktu yang sangat singkat. Jika tidak direncanakan, ini dapat menyebabkan permintaan dengan cepat melampaui sumber daya yang tersedia.

Cara terbaik untuk menangani guncangan permintaan adalah dengan mengantisipasi dan merencanakannya dengan tepat. Karena penskalaan otomatis dapat memakan waktu, kami menyarankan Anda untuk menskalakan aplikasi Anda sebelum kejutan permintaan dimulai. Untuk hasil terbaik, kami sarankan memiliki rencana bisnis yang melibatkan kolaborasi erat antara tim yang menggunakan kalender bersama. Tim yang merencanakan acara harus bekerja sama dengan tim yang bertanggung jawab atas aplikasi terlebih dahulu. Ini memberi tim itu cukup waktu untuk memiliki

rencana penjadwalan yang jelas. Mereka dapat menjadwalkan kapasitas untuk skala sebelum acara dan untuk skala setelah acara. Untuk informasi lebih lanjut, lihat Penskalaan terjadwal dalam Panduan Pengguna Application Auto Scaling.

Jika Anda memiliki paket Enterprise Support, pastikan juga untuk bekerja dengan Technical Account Manager (TAM) Anda. TAM Anda dapat memverifikasi kuota layanan Anda dan memastikan bahwa kuota yang diperlukan dinaikkan sebelum acara dimulai. Dengan cara ini, Anda tidak secara tidak sengaja mencapai kuota layanan apa pun. Mereka juga dapat membantu Anda dengan melakukan pemanasan awal seperti penyeimbang beban untuk memastikan acara Anda berjalan lancar.

Menangani guncangan permintaan yang tidak terjadwal adalah masalah yang lebih sulit. Guncangan tak terjadwal, jika amplitudo cukup besar, dapat dengan cepat menyebabkan permintaan melebihi kapasitas. Ini juga dapat melampaui kemampuan autoscaling untuk bereaksi. Cara terbaik untuk mempersiapkan guncangan yang tidak terjadwal adalah dengan menyediakan sumber daya yang berlebihan. Anda harus memiliki sumber daya yang cukup untuk menangani permintaan lalu lintas maksimum yang diantisipasi setiap saat.

Mempertahankan kapasitas maksimum untuk mengantisipasi guncangan permintaan yang tidak terjadwal bisa mahal. Untuk mengurangi dampak biaya, temukan metrik indikator utama atau peristiwa yang memprediksi guncangan permintaan besar akan segera terjadi. Jika metrik atau peristiwa secara andal memberikan pemberitahuan awal yang signifikan, segera mulailah proses penskalaan saat peristiwa terjadi atau saat metrik melewati ambang batas tertentu yang Anda tetapkan.

Jika aplikasi Anda rentan terhadap guncangan permintaan tak terjadwal yang tiba-tiba, pertimbangkan untuk menambahkan mode kinerja tinggi ke aplikasi Anda yang mengorbankan fungsionalitas non-kritis tetapi mempertahankan fungsionalitas penting bagi pelanggan. Misalnya, asumsikan bahwa aplikasi Anda dapat beralih dari menghasilkan respons khusus yang mahal menjadi menyajikan halaman respons statis. Dalam skenario ini, Anda dapat meningkatkan throughput secara signifikan tanpa menskalakan aplikasi sama sekali.

Terakhir, Anda dapat mempertimbangkan untuk memecah layanan monolitik untuk mengatasi guncangan permintaan dengan lebih baik. Jika aplikasi Anda adalah layanan monolitik yang mahal untuk dijalankan dan lambat untuk skala, Anda mungkin dapat mengekstrak atau menulis ulang bagian penting kinerja dan menjalankannya sebagai layanan terpisah. Layanan baru ini kemudian dapat diskalakan secara independen dari komponen yang kurang kritis. Memiliki fleksibilitas untuk meningkatkan fungsionalitas penting kinerja secara terpisah dari bagian lain dari aplikasi Anda dapat mengurangi waktu yang diperlukan untuk menambah kapasitas dan membantu menghemat biaya.

Kapasitas kluster Amazon ECS

Anda dapat menyediakan kapasitas ke cluster Amazon ECS dalam beberapa cara. Misalnya, Anda dapat meluncurkan EC2 instans Amazon dan mendaftarkannya ke cluster saat start-up menggunakan agen penampung Amazon ECS. Namun, metode ini bisa menjadi tantangan karena Anda perlu mengelola penskalaan sendiri. Oleh karena itu, kami menyarankan Anda menggunakan penyedia kapasitas Amazon ECS. Penyedia kapasitas mengelola penskalaan sumber daya untuk Anda. Ada tiga jenis penyedia kapasitas: Amazon EC2, Fargate, dan Fargate Spot. Untuk informasi selengkapnya tentang penyedia kapasitas Fargate, lihat klaster Amazon ECS untuk beban kerja tipe peluncuran Fargate dan untuk jenis peluncuran, lihat klaster Amazon ECS untuk beban kerja tipe EC2 peluncuran. EC2

Penyedia kapasitas Fargate dan Fargate Spot menangani siklus hidup tugas Fargate untuk Anda. Fargate menyediakan kapasitas sesuai permintaan, dan Fargate Spot menyediakan kapasitas Spot. Saat Anda meluncurkan tugas, Amazon ECS menyediakan sumber daya Fargate untuk Anda. Sumber daya Fargate ini dilengkapi dengan memori dan unit CPU yang secara langsung sesuai dengan batas tingkat tugas yang Anda nyatakan dalam definisi tugas Anda. Setiap tugas menerima sumber daya Fargate sendiri, membuat hubungan 1:1 antara tugas dan sumber daya komputasi.

Tugas yang berjalan di Fargate Spot dapat mengalami gangguan. Interupsi datang setelah peringatan dua menit. Ini terjadi selama periode permintaan berat. Fargate Spot bekerja paling baik untuk beban kerja yang toleran interupsi seperti pekerjaan batch, pengembangan, atau lingkungan pementasan. Mereka juga cocok untuk skenario lain di mana ketersediaan tinggi dan latensi rendah bukanlah persyaratan.

Anda dapat menjalankan tugas Fargate Spot bersama tugas Fargate sesuai permintaan. Dengan menggunakannya bersama-sama, Anda menerima kapasitas “burst” penyediaan dengan biaya lebih rendah.

Amazon ECS juga dapat mengelola kapasitas EC2 instans Amazon untuk tugas Anda. Setiap penyedia EC2 kapasitas Amazon dikaitkan dengan grup EC2 Auto Scaling Amazon yang Anda tentukan. Saat Anda menggunakan penyedia EC2 kapasitas Amazon, penskalaan otomatis cluster mempertahankan ukuran grup EC2 Auto Scaling Amazon untuk memastikan semua tugas terjadwal dapat ditempatkan.

Memilih ukuran tugas Fargate untuk Amazon ECS

Untuk definisi AWS Fargate tugas, Anda harus menentukan CPU dan memori pada tingkat tugas, dan tidak perlu memperhitungkan overhead apa pun. Anda juga dapat menentukan CPU dan

memori pada tingkat kontainer untuk tugas Fargate. Namun, melakukannya tidak diperlukan. Batas sumber daya harus lebih besar dari atau sama dengan reservasi apa pun yang Anda nyatakan. Dalam kebanyakan kasus, Anda dapat mengaturnya ke jumlah reservasi dari setiap kontainer yang dideklarasikan dalam definisi tugas Anda. Kemudian, bulatkan angka ke ukuran Fargate terdekat. Untuk informasi selengkapnya tentang ukuran yang tersedia, lihat CPU tugas dan memori.

Mempercepat penyediaan kapasitas cluster Amazon ECS dengan penyedia kapasitas di Amazon EC2

Pelanggan yang menjalankan Amazon ECS di Amazon EC2 dapat memanfaatkan Amazon ECS Cluster Auto Scaling (CAS) untuk mengelola penskalaan grup Amazon Auto Scaling (EC2 ASG). Dengan CAS, Anda dapat mengonfigurasi Amazon ECS untuk menskalakan ASG Anda secara otomatis, dan hanya fokus menjalankan tugas Anda. Amazon ECS akan memastikan skala ASG masuk dan keluar sesuai kebutuhan tanpa intervensi lebih lanjut diperlukan. Penyedia kapasitas Amazon ECS digunakan untuk mengelola infrastruktur di klaster Anda dengan memastikan ada cukup instance kontainer untuk memenuhi permintaan aplikasi Anda. Untuk mempelajari cara kerja Amazon ECS CAS, lihat Deep Dive di Auto Scaling Amazon ECS Cluster.

Karena CAS bergantung pada integrasi CloudWatch berbasis dengan ASG untuk menyesuaikan kapasitas cluster, ia memiliki latensi bawaan yang terkait dengan penerbitan CloudWatch metrik, waktu yang dibutuhkan metrik `CapacityProviderReservation` untuk melanggar CloudWatch alarm (baik tinggi maupun rendah), dan waktu yang dibutuhkan oleh instans Amazon yang baru diluncurkan untuk pemanasan. EC2 Anda dapat mengambil tindakan berikut untuk membuat CAS lebih responsif untuk penerapan yang lebih cepat:

Ukuran penskalaan langkah penyedia kapasitas

Penyedia kapasitas Amazon ECS pada akhirnya akan grow/shrink menggunakan instans kontainer untuk memenuhi permintaan aplikasi Anda. Jumlah minimum instans yang akan diluncurkan Amazon ECS diatur ke 1 secara default. Ini dapat menambah waktu tambahan untuk penerapan Anda, jika beberapa instance diperlukan untuk menempatkan tugas Anda yang tertunda. Anda dapat meningkatkan `minimumScalingStepSize`dengan menggunakan Amazon ECS API untuk meningkatkan jumlah minimum instans yang diskalakan Amazon ECS masuk atau keluar sekaligus. A `maximumScalingStepSize`yang terlalu rendah dapat membatasi berapa banyak instance kontainer yang diskalakan masuk atau keluar sekaligus, yang dapat memperlambat penerapan Anda.

> **Note**
>
> Konfigurasi ini saat ini hanya tersedia dengan menggunakan `CreateCapacityProvider`atau `UpdateCapacityProvider` APIs.

Periode pemanasan contoh

Periode pemanasan instans adalah periode waktu setelah EC2 instans Amazon yang baru diluncurkan dapat berkontribusi pada CloudWatch metrik untuk grup Auto Scaling. Setelah periode pemanasan yang ditentukan berakhir, instance dihitung terhadap metrik agregat ASG, dan CAS melanjutkan dengan iterasi perhitungan berikutnya untuk memperkirakan jumlah instance yang diperlukan.

Nilai default untuk `instanceWarmupPeriod`adalah 300 detik, yang dapat Anda konfigurasikan ke nilai yang lebih rendah dengan menggunakan `CreateCapacityProvider`atau `UpdateCapacityProvider` APIs untuk penskalaan yang lebih responsif.

Kapasitas cadangan

Jika penyedia kapasitas Anda tidak memiliki instans kontainer yang tersedia untuk menempatkan tugas, maka perlu meningkatkan (skala) kapasitas klaster dengan meluncurkan EC2 instans Amazon dengan cepat, dan menunggu hingga boot sebelum dapat meluncurkan kontainer di dalamnya. Ini dapat secara signifikan menurunkan tingkat peluncuran tugas. Anda memiliki dua opsi di sini.

Dalam hal ini, memiliki EC2 kapasitas Amazon cadangan yang sudah diluncurkan dan siap menjalankan tugas akan meningkatkan tingkat peluncuran tugas yang efektif. Anda dapat menggunakan `Target Capacity` konfigurasi untuk menunjukkan bahwa Anda ingin mempertahankan kapasitas cadangan di cluster Anda. Misalnya, dengan menetapkan `Target Capacity` 80%, Anda menunjukkan bahwa cluster Anda membutuhkan kapasitas cadangan 20% setiap saat. Kapasitas cadangan ini dapat memungkinkan tugas mandiri apa pun segera diluncurkan, memastikan peluncuran tugas tidak dibatasi. Trade-off untuk pendekatan ini adalah potensi peningkatan biaya untuk menjaga kapasitas cluster cadangan.

Pendekatan alternatif yang dapat Anda pertimbangkan adalah menambahkan ruang kepala ke layanan Anda, bukan ke penyedia kapasitas. Ini berarti bahwa alih-alih mengurangi `Target Capacity` konfigurasi untuk meluncurkan kapasitas cadangan, Anda dapat meningkatkan jumlah replika di layanan Anda dengan memodifikasi metrik penskalaan pelacakan target atau ambang batas penskalaan langkah dari penskalaan otomatis layanan. Perhatikan bahwa pendekatan

ini hanya akan membantu untuk beban kerja runcing, tetapi tidak akan berpengaruh saat Anda menerapkan layanan baru dan beralih dari 0 ke N tugas untuk pertama kalinya. Untuk informasi selengkapnya tentang kebijakan penskalaan terkait, lihat Kebijakan Penskalaan Pelacakan Target atau Kebijakan Penskalaan Langkah

Akses fitur Amazon ECS dengan pengaturan akun

Anda dapat masuk ke pengaturan akun Amazon ECS untuk memilih masuk atau keluar dari fitur tertentu. Untuk masing-masing Wilayah AWS, Anda dapat memilih untuk, atau memilih keluar dari, setiap pengaturan akun di tingkat akun atau untuk pengguna atau peran tertentu.

Anda mungkin ingin memilih masuk atau keluar dari fitur tertentu jika salah satu dari berikut ini relevan bagi Anda:

- Pengguna atau peran dapat memilih atau memilih keluar dari pengaturan akun tertentu untuk akun individu mereka.
- Pengguna atau peran dapat mengatur pengaturan opt-in atau opt-out default untuk semua pengguna di akun.
- Pengguna root atau pengguna dengan hak administrator dapat memilih untuk, atau memilih keluar dari, peran atau pengguna tertentu apa pun di akun. Jika pengaturan akun untuk pengguna root diubah, itu menetapkan default untuk semua pengguna dan peran yang tidak dipilih oleh pengaturan akun individual.

> **Note**
>
> Pengguna federasi mengasumsikan pengaturan akun pengguna root dan tidak dapat mengatur pengaturan akun eksplisit untuk mereka secara terpisah.

Pengaturan akun berikut tersedia. Anda harus secara terpisah opt-in dan opt-out ke setiap pengaturan akun.

Nama sumber daya	Pelajari selengkapnya
`containerInsights`	Wawasan Kontainer
`serviceLongArnFormat`	Nama Sumber Daya Amazon (ARNs) dan IDs

Nama sumber daya	Pelajari selengkapnya

`taskLongArnFormat`
 	
`containerInstanceLongArnFormat`	

`tagResourceAuthorization`	Otorisasi penandaan
`fargateFIPSMode`	AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140)
`fargateTaskRetirementWaitPeriod`	AWS Fargate tugas pensiun waktu tunggu
`guardDutyActivate`	Pemantauan Runtime (GuardDuty integrasi Amazon)
`dualStackIPv6`	IPv6 VPC tumpukan ganda
`awsvpcTrunking`	Meningkatkan antarmuka jaringan instance kontainer Linux
`defaultLogDriverMode`	Mode driver log default

Nama Sumber Daya Amazon (ARNs) dan IDs

Saat sumber daya Amazon ECS dibuat, setiap sumber daya diberi Nama Sumber Daya Amazon (ARN) dan pengenal sumber daya (ID) yang unik. Jika Anda menggunakan alat baris perintah atau Amazon ECS API untuk bekerja dengan Amazon ECS, sumber daya ARNs atau IDs diperlukan untuk perintah tertentu. Misalnya, jika Anda menggunakan AWS CLI perintah stop-task untuk menghentikan tugas, Anda harus menentukan tugas ARN atau ID dalam perintah.

Amazon ECS memperkenalkan format baru untuk Amazon Resource Names (ARNs) dan sumber daya IDs untuk layanan, tugas, dan instans penampung Amazon ECS. Status keikutsertaan untuk setiap jenis sumber daya menentukan format Amazon Resource Name (ARN) yang digunakan sumber daya. Anda harus memilih format ARN baru untuk menggunakan fitur seperti penandaan sumber daya untuk jenis sumber daya tersebut.

Anda dapat memilih dan memilih keluar dari format Nama Sumber Daya Amazon (ARN) dan ID sumber daya baru berdasarkan per wilayah. Saat ini, setiap akun baru yang dibuat akan disertakan secara default.

Anda dapat menyertakan atau menolak format Amazon Resource Name (ARN) dan ID sumber daya baru kapan saja. Setelah Anda ikut serta, sumber daya baru apa pun yang Anda buat menggunakan format baru.

> **Note**
>
> ID sumber daya tidak berubah setelah dibuat. Oleh karena itu, memilih masuk atau keluar dari format baru tidak memengaruhi sumber daya IDs yang ada.

Bagian berikut menjelaskan cara mengubah format ARN dan ID sumber daya. Untuk informasi selengkapnya tentang transisi ke format baru, lihat FAQ Amazon Elastic Container Service.

Format Amazon Resource Name (ARN)

Beberapa sumber daya memiliki nama yang mudah digunakan, seperti layanan bernama `production`. Dalam hal lain, Anda harus menentukan sumber daya menggunakan format Amazon Resource Name (ARN). Format ARN baru untuk tugas, layanan, dan instans penampung Amazon ECS menyertakan nama cluster. Untuk informasi lebih lanjut tentang menyertakan format baru ARN, lihat Memodifikasi pengaturan akun Amazon ECS.

Tabel berikut menunjukkan format saat ini dan format baru untuk setiap jenis sumber daya.

Jenis sumber daya	ARN
Instans kontainer	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`container-instance-id`* Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`cluster-name`* `/`*`container-instance-id`*
Layanan Amazon ECS	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:service/` *`service-name`* Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:service/` *`cluster-name`* `/`*`service-name`*
Tugas Amazon ECS	Saat ini: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:task/`*`task-id`*

Jenis sumber daya	ARN
	Baru: `arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:task/`*`cluster-name`*` /`*`task-id`*

Panjang ID Sumber Daya

ID sumber daya berisi kombinasi unik dari huruf dan angka. Format ID sumber daya baru mencakup lebih pendek IDs untuk tugas Amazon ECS dan instans penampung. Format ID sumber daya saat ini adalah 36 karakter. Yang baru IDs dalam format 32 karakter yang tidak menyertakan tanda hubung apa pun. Untuk informasi lebih lanjut tentang memilih format ID sumber daya baru, lihat Memodifikasi pengaturan akun Amazon ECS.

Nilai default-nya `enabled`.

Sumber daya hanya akan diluncurkan setelah format ARN dan ID sumber daya baru menerima status disertakan. Semua sumber daya yang ada tidak terpengaruh. Agar layanan dan tugas Amazon ECS dapat beralih ke format ARN dan ID sumber daya baru, Anda harus membuat ulang layanan atau tugas. Untuk mentransisikan instance container ke ARN baru dan format ID resource, instance container harus dikeringkan dan instance container baru harus diluncurkan dan didaftarkan ke cluster.

> **Note**
>
> Tugas yang diluncurkan oleh layanan Amazon ECS hanya dapat menerima ARN baru dan format ID sumber daya jika layanan dibuat pada atau setelah 16 November 2018, dan pengguna yang membuat layanan telah memilih format baru untuk tugas.

Lini masa format ARN dan ID sumber daya

Garis waktu untuk periode opt-in dan opt-out untuk Amazon Resource Name (ARN) baru dan format ID sumber daya untuk sumber daya Amazon ECS berakhir pada 1 April 2021. Secara default, semua akun dipilih ke format baru. Semua sumber daya baru yang dibuat menerima format baru, dan Anda tidak dapat lagi memilih keluar.

Wawasan Kontainer

Pada 2 Desember 2024, AWS merilis Container Insights dengan peningkatan observabilitas untuk Amazon ECS. Versi ini mendukung peningkatan observabilitas untuk cluster Amazon ECS menggunakan jenis peluncuran Amazon dan EC2 Fargate. Setelah mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang disempurnakan di Amazon ECS, Container Insights secara otomatis mengumpulkan telemetri infrastruktur terperinci dari level cluster hingga level container di lingkungan Anda dan menampilkan data Anda di dasbor yang menampilkan berbagai metrik dan dimensi. Anda kemudian dapat menggunakan out-of-the-box dasbor ini di konsol Container Insights untuk lebih memahami kesehatan dan kinerja container Anda, dan untuk mengurangi masalah lebih cepat dengan mengidentifikasi anomali.

Kami menyarankan Anda menggunakan Wawasan Kontainer dengan observabilitas yang ditingkatkan, bukan Wawasan Kontainer karena ini memberikan visibilitas mendetail di lingkungan penampung Anda, sehingga mengurangi waktu rata-rata untuk resolusi. Untuk informasi selengkapnya, lihat Amazon ECS Container Insights dengan metrik observabilitas yang disempurnakan di Panduan Pengguna.Amazon CloudWatch

Default untuk pengaturan `containerInsights` akun adalah`disabled`.

Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan

Gunakan perintah berikut untuk mengaktifkan Container Insights dengan observabilitas yang ditingkatkan.

Atur pengaturan `containerInsights` akun ke`enhanced`.

```
aws ecs put-account-setting --name containerInsights --value enhanced
```

Contoh Output

```
{
    "setting": {
        "name": "containerInsights",
        "value": "enhanced",
        "principalArn": "arn:aws:iam::123456789012:johndoe",
         "type": user
    }
}
```

Setelah Anda menyetel setelan akun ini, semua cluster baru secara otomatis menggunakan Wawasan Kontainer dengan kemampuan observasi yang disempurnakan. Gunakan `update-cluster-settings` perintah untuk menambahkan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan ke klaster yang ada, atau untuk memutakhirkan klaster dari Container Insights ke Container Insights dengan peningkatan observabilitas.

```
aws ecs update-cluster-settings --cluster cluster-name --settings
 name=containerInsights,value=enhanced
```

Anda juga dapat menggunakan konsol untuk mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan. Untuk informasi selengkapnya, lihat Memodifikasi pengaturan akun Amazon ECS.

Wawasan Kontainer

Saat Anda menyetel setelan `containerInsights` akun`enabled`, semua cluster baru mengaktifkan Wawasan Kontainer secara default. Anda dapat memodifikasi cluster yang ada dengan menggunakan`update-cluster-settings`.

Untuk menggunakan Wawasan Kontainer, setel setelan `containerInsights` akun ke`enabled`. Gunakan perintah berikut untuk mengaktifkan Wawasan Kontainer.

```
aws ecs put-account-setting --name containerInsights --value enabled
```

Contoh Output

```
{
    "setting": {
        "name": "containerInsights",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:johndoe",
         "type": user
    }
}
```

Saat Anda menyetel setelan `containerInsights` akun`enabled`, semua cluster baru mengaktifkan Wawasan Kontainer secara default. Gunakan `update-cluster-settings` perintah untuk menambahkan Wawasan Kontainer ke cluster yang ada.

```
aws ecs update-cluster-settings --cluster cluster-name --settings
 name=containerInsights,value=enabled
```

Anda juga dapat menggunakan konsol untuk mengonfigurasi Wawasan Kontainer. Untuk informasi selengkapnya, lihat Memodifikasi pengaturan akun Amazon ECS.

AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140)

Fargate mendukung Federal Information Processing Standard (FIPS-140) yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Ini adalah standar pemerintah Amerika Serikat dan Kanada saat ini, dan berlaku untuk sistem yang harus mematuhi Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) atau Program Manajemen Risiko dan Otorisasi Federal (FedRAMP).

Nama sumber daya adalah`fargateFIPSMode`.

Nilai default-nya `disabled`.

Anda harus mengaktifkan kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140) di Fargate. Untuk informasi selengkapnya, lihat the section called “AWS Fargate Kepatuhan FIPS-140”.

> **Important**
>
> Pengaturan `fargateFIPSMode` akun hanya dapat diubah menggunakan Amazon ECS API atau. AWS CLI Untuk informasi selengkapnya, lihat Memodifikasi pengaturan akun Amazon ECS.

Jalankan `put-account-setting-default` dengan `fargateFIPSMode` opsi yang disetel ke`enabled`. Untuk informasi selengkapnya, lihat, put-account-setting-defaultdi Referensi API Amazon Elastic Container Service.

- Anda dapat menggunakan perintah berikut untuk mengaktifkan kepatuhan FIPS-140.

  ```
  aws ecs put-account-setting-default --name fargateFIPSMode --value enabled
  ```

  Contoh Output

```
{
    "setting": {
        "name": "fargateFIPSMode",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
         "type": user
    }
}
```

Anda dapat menjalankan `list-account-settings` untuk melihat status kepatuhan FIPS-140 saat ini. Gunakan `effective-settings` opsi untuk melihat pengaturan tingkat akun.

```
aws ecs list-account-settings --effective-settings
```

Otorisasi penandaan

Amazon ECS memperkenalkan otorisasi penandaan untuk pembuatan sumber daya. Pengguna harus memiliki izin penandaan untuk tindakan yang membuat sumber daya, seperti. `ecsCreateCluster` Saat Anda membuat sumber daya dan menentukan tag untuk sumber daya tersebut, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa ada izin untuk membuat tag. Oleh karena itu, Anda harus memberikan izin eksplisit untuk menggunakan tindakan. `ecs:TagResource` Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”.

Untuk ikut serta dalam menandai otorisasi, jalankan `put-account-setting-default` dengan `tagResourceAuthorization` opsi yang disetel ke. `enable` Untuk informasi selengkapnya, lihat, put-account-setting-defaultdi Referensi API Amazon Elastic Container Service. Anda dapat menjalankan `list-account-settings` untuk melihat status otorisasi penandaan saat ini.

- Anda dapat menggunakan perintah berikut untuk mengaktifkan otorisasi penandaan.

  ```
  aws ecs put-account-setting-default --name tagResourceAuthorization --value on --
  region region
  ```

  Contoh Output

  ```
  {
      "setting": {
  ```

```
        "name": "tagResourceAuthorization",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": user
    }
}
```

Setelah mengaktifkan otorisasi penandaan, Anda harus mengonfigurasi izin yang sesuai untuk memungkinkan pengguna menandai sumber daya saat pembuatan. Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”.

Anda dapat menjalankan `list-account-settings` untuk melihat status otorisasi penandaan saat ini. Gunakan `effective-settings` opsi untuk melihat pengaturan tingkat akun.

```
aws ecs list-account-settings --effective-settings
```

Menandai garis waktu otorisasi

Anda dapat mengonfirmasi apakah otorisasi penandaan aktif dengan menjalankan `list-account-settings` untuk melihat nilainya. `tagResourceAuthorization` Ketika nilainyaon, itu berarti otorisasi penandaan sedang digunakan. Untuk informasi selengkapnya, lihat, list-account-settingsdi Referensi API Amazon Elastic Container Service.

Berikut ini adalah tanggal-tanggal penting yang terkait dengan otorisasi penandaan.

- 18 April 2023 - Otorisasi penandaan diperkenalkan. Semua akun baru dan yang sudah ada harus ikut serta untuk menggunakan fitur ini. Anda dapat memilih untuk mulai menggunakan otorisasi penandaan. Dengan ikut serta, Anda harus memberikan izin yang sesuai.
- 9 Februari 2024 - 6 Maret 2024 - Semua akun baru dan akun yang ada yang tidak terkena dampak telah mengaktifkan otorisasi penandaan secara default. Anda dapat mengaktifkan atau menonaktifkan pengaturan `tagResourceAuthorization` akun untuk memverifikasi kebijakan IAM Anda.

  AWS telah memberi tahu akun yang terkena dampak.

  Untuk menonaktifkan fitur, jalankan `put-account-setting-default` dengan `tagResourceAuthorization` opsi yang disetel ke`off`.
- 7 Maret 2024 - Jika Anda telah mengaktifkan otorisasi penandaan, Anda tidak dapat lagi menonaktifkan pengaturan akun.

Kami menyarankan Anda menyelesaikan pengujian kebijakan IAM Anda sebelum tanggal ini.

- 29 Maret 2024 - Semua akun menggunakan otorisasi penandaan. Pengaturan tingkat akun tidak akan lagi tersedia di konsol Amazon ECS atau. AWS CLI

AWS Fargate tugas pensiun waktu tunggu

AWS mengirimkan pemberitahuan ketika Anda memiliki tugas Fargate yang berjalan pada revisi versi platform yang ditandai untuk pensiun. Untuk informasi selengkapnya, lihat Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS .

AWS bertanggung jawab untuk menambal dan memelihara infrastruktur yang mendasari AWS Fargate. Saat AWS menentukan bahwa pembaruan keamanan atau infrastruktur diperlukan untuk tugas Amazon ECS yang dihosting di Fargate, tugas harus dihentikan dan tugas baru diluncurkan untuk menggantikannya. Anda dapat mengonfigurasi periode tunggu sebelum tugas dihentikan untuk ditambal. Anda memiliki opsi untuk segera menghentikan tugas, menunggu 7 hari kalender, atau menunggu 14 hari kalender.

Pengaturan ini ada di tingkat akun.

Anda dapat mengkonfigurasi waktu Fargate memulai tugas pensiun. Untuk beban kerja yang memerlukan aplikasi pembaruan segera, pilih pengaturan langsung (`0`). Ketika Anda membutuhkan kontrol lebih, misalnya, ketika tugas hanya dapat dihentikan selama jendela tertentu, konfigurasikan opsi 7 hari (7), atau 14 hari (14).

Kami menyarankan Anda memilih masa tunggu yang lebih pendek untuk mengambil revisi versi platform yang lebih baru lebih cepat.

Konfigurasikan periode tunggu dengan menjalankan `put-account-setting-default` atau `put-account-setting` sebagai pengguna root atau pengguna administratif. Gunakan `fargateTaskRetirementWaitPeriod` opsi untuk `name` dan `value` opsi yang disetel ke salah satu nilai berikut:

- `0`- AWS mengirim pemberitahuan, dan segera mulai pensiun tugas yang terpengaruh.
- 7- AWS mengirim pemberitahuan, dan menunggu 7 hari kalender sebelum mulai pensiun tugas yang terpengaruh.
- 14- AWS mengirim pemberitahuan, dan menunggu 14 hari kalender sebelum mulai pensiun tugas yang terpengaruh.

Defaultnya adalah 7 hari.

Untuk informasi selengkapnya, lihat, put-account-setting-defaultdan put-account-settingdi Referensi API Amazon Elastic Container Service.

Anda dapat menjalankan perintah berikut untuk mengatur periode tunggu menjadi 14 hari.

```
aws ecs put-account-setting-default --name fargateTaskRetirementWaitPeriod --value 14
```

Contoh Output

```
{
    "setting": {
        "name": "fargateTaskRetirementWaitPeriod",
        "value": "14",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type: user"
    }
}
```

Anda dapat menjalankan `list-account-settings` untuk melihat waktu tunggu pensiun tugas Fargate saat ini. Gunakan `effective-settings` opsi.

```
aws ecs list-account-settings --effective-settings
```

Meningkatkan antarmuka jaringan instance kontainer Linux

Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) miliknya sendiri, yang dilampirkan ke instance container yang menghostingnya. Ada batas default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke EC2 instance Amazon, dan antarmuka jaringan utama dihitung sebagai satu. Misalnya, secara default sebuah `c5.large` instance mungkin memiliki hingga tiga yang ENIs melekat padanya. Antarmuka jaringan utama untuk instance dihitung sebagai satu, sehingga Anda dapat melampirkan dua tambahan ENIs ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukanENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini.

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan ENI kepadatan menggunakan jenis EC2 instans Amazon yang didukung. Saat Anda menggunakan jenis instans ini dan mengaktifkan setelan `awsvpcTrunking` akun, tambahan ENIs tersedia pada instance container

yang baru diluncurkan. Konfigurasi ini memungkinkan Anda untuk menempatkan lebih banyak tugas pada setiap instance kontainer.

Misalnya, `c5.large` contoh dengan `awsvpcTrunking` memiliki ENI batas peningkatan dua belas. Instance container akan memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan "trunk" ke instance container. Jadi, konfigurasi ini mengizinkan Anda meluncurkan sepuluh tugas pada instans kontainer, tidak hanya dua tugas seperti saat ini.

Pemantauan Runtime (GuardDuty integrasi Amazon)

Runtime Monitoring adalah layanan deteksi ancaman cerdas yang melindungi beban kerja yang berjalan di Fargate dan instance EC2 container dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah.

`guardDutyActivate`Parameter ini hanya-baca di Amazon ECS dan menunjukkan apakah Runtime Monitoring diaktifkan atau dimatikan oleh administrator keamanan Anda di akun Amazon ECS Anda. GuardDuty mengontrol pengaturan akun ini atas nama Anda. Untuk informasi selengkapnya, lihat Melindungi beban kerja Amazon ECS dengan Runtime Monitoring.

Anda dapat menjalankan `list-account-settings` untuk melihat pengaturan GuardDuty integrasi saat ini.

```
aws ecs list-account-settings
```

Contoh Output

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:doej",
        "type": aws-managed"
    }
}
```

IPv6 VPC tumpukan ganda

Amazon ECS mendukung penyediaan tugas dengan IPv6 alamat selain IPv4 alamat pribadi utama.

Agar tugas menerima IPv6 alamat, tugas harus menggunakan mode `awsvpc` jaringan, harus diluncurkan dalam VPC yang dikonfigurasi untuk mode dual-stack, dan pengaturan `dualStackIPv6`

akun harus diaktifkan. Untuk informasi selengkapnya tentang persyaratan lainnya, lihat Menggunakan VPC dalam mode tumpukan dobel untuk jenis EC2 peluncuran dan Menggunakan VPC dalam mode tumpukan dobel untuk jenis peluncuran Fargate.

> **Important**
>
> Pengaturan `dualStackIPv6` akun hanya dapat diubah menggunakan Amazon ECS API atau. AWS CLI Untuk informasi selengkapnya, lihat Memodifikasi pengaturan akun Amazon ECS.

Jika Anda memiliki tugas yang berjalan menggunakan mode `awsvpc` jaringan di subnet yang IPv6 diaktifkan antara tanggal 1 Oktober 2020 dan 2 November 2020, pengaturan `dualStackIPv6` akun default di Wilayah tempat tugas dijalankan adalah`disabled`. Jika kondisi tersebut tidak terpenuhi, `dualStackIPv6` pengaturan default di Wilayah adalah`enabled`.

Nilai default-nya `disabled`.

Mode driver log default

Amazon ECS mendukung pengaturan mode pengiriman default pesan log dari wadah ke driver log yang dipilih. Mode pengiriman mempengaruhi stabilitas aplikasi ketika aliran log dari kontainer ke driver log terganggu.

`defaultLogDriverMode`Pengaturan mendukung dua nilai: `blocking` dan`non-blocking`. Untuk informasi selengkapnya tentang mode pengiriman ini, lihat LogConfigurationdi Referensi API Amazon Elastic Container Service.

Jika Anda tidak menentukan mode pengiriman dalam definisi kontainer Anda`logConfiguration`, mode yang Anda tentukan menggunakan pengaturan akun ini akan digunakan sebagai default.

Mode pengiriman default adalah`non-blocking`.

> **Note**
>
> Pada 25 Juni 2025, Amazon ECS mengubah mode driver log default dari menjadi `blocking` `non-blocking` memprioritaskan ketersediaan tugas daripada pencatatan. Untuk terus menggunakan `blocking` mode setelah perubahan ini, lakukan salah satu hal berikut:

- Tetapkan `mode` opsi dalam definisi kontainer Anda `logConfiguration` sebagai`blocking`.
- Atur pengaturan `defaultLogDriverMode` akun ke`blocking`.

Untuk mengatur mode driver log default ke`blocking`, Anda dapat menjalankan perintah berikut.

```
aws ecs put-account-setting-default --name defaultLogDriverMode --value "blocking"
```

Melihat pengaturan akun Amazon ECS menggunakan konsol

Lihat pengaturan akun Anda di konsol untuk melihat fitur mana yang dapat Anda akses.

Important

Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan AWS CLI.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.

Memodifikasi pengaturan akun Amazon ECS

Ubah pengaturan akun Anda untuk mengakses fitur Amazon ECS.

`guardDutyActivate`Parameter ini hanya-baca di Amazon ECS dan menunjukkan apakah Runtime Monitoring diaktifkan atau dimatikan oleh administrator keamanan Anda di akun Amazon ECS Anda. GuardDuty mengontrol pengaturan akun ini atas nama Anda. Untuk informasi selengkapnya, lihat Melindungi beban kerja Amazon ECS dengan Runtime Monitoring.

> **Important**
>
> Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan AWS CLI.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.
4. Pilih Perbarui.
5. (Opsional) Untuk menambah atau mengurangi jumlah tugas yang dapat Anda jalankan dalam mode jaringan awsvpc untuk setiap EC2 instance, di bawah AWSVPCTrunking, pilih Trunking. AWSVPC
6. (Opsional) Untuk menggunakan, atau berhenti menggunakan CloudWatch Wawasan Kontainer secara default untuk cluster, di bawah observabilitas CloudWatch Wawasan Kontainer, pilih salah satu opsi berikut:
   - Untuk menggunakan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan, pilih Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan.
   - Untuk menggunakan Container Insights, pilih Container Insights.
   - Untuk berhenti menggunakan Wawasan Kontainer, pilih Dimatikan.
7. (Opsional) Untuk mengaktifkan atau menonaktifkan otorisasi penandaan, di bawah Otorisasi Penandaan Sumber Daya, pilih atau hapus Otorisasi Penandaan Sumber Daya.
8. (Opsional) Untuk mengonfigurasi mode driver log default ketika mode pengiriman log tidak ditentukan dalam wadah`logConfiguration`, di bawah mode driver log default, pilih salah satu opsi berikut:
   - Untuk mengatur mode driver log default sebagai`blocking`, pilih Memblokir.
   - Untuk mengatur mode driver log default sebagai`non-blocking`, pilih Non-blocking.
9. Pilih Simpan perubahan.
10. Pada layar konfirmasi, pilih Konfirmasi untuk menyimpan pilihan.

Langkah selanjutnya

Jika mengaktifkan Container Insights dengan observabilitas yang disempurnakan atau Wawasan Kontainer, Anda dapat memperbarui kluster yang ada secara opsional untuk menggunakan fitur tersebut. Untuk informasi selengkapnya, lihat Memperbarui kluster Amazon ECS.

Mengembalikan ke pengaturan akun Amazon ECS default

Anda dapat menggunakan AWS Management Console untuk mengembalikan pengaturan akun Amazon ECS Anda ke default.

Opsi Kembalikan ke akun default hanya tersedia jika pengaturan akun Anda tidak lagi menjadi pengaturan default.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di bilah navigasi di bagian atas layar, pilih Wilayah yang akan dilihat pengaturan akunnya.
3. Di halaman navigasi, pilih Pengaturan Akun.
4. Pilih Perbarui.
5. Pilih Kembalikan ke akun default.
6. Pada layar konfirmasi, pilih Konfirmasi untuk menyimpan pilihan.

Mengelola setelan akun Amazon ECS menggunakan AWS CLI

Anda dapat mengelola pengaturan akun menggunakan Amazon ECS API, AWS CLI atau SDKs. Pengaturan`dualStackIPv6`, `fargateFIPSMode` dan `fargateTaskRetirementWaitPeriod` akun hanya dapat dilihat atau diubah menggunakan alat tersebut.

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Untuk informasi tentang tindakan API yang tersedia untuk definisi tugas, lihat Tindakan setelan akun di Referensi API Amazon Elastic Container Service.

Gunakan salah satu perintah berikut untuk memodifikasi pengaturan akun default untuk semua pengguna atau peran di akun Anda. Perubahan ini berlaku untuk seluruh AWS akun kecuali pengguna atau peran secara eksplisit mengesampingkan pengaturan ini untuk diri mereka sendiri.

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled --
region us-east-2
```

  Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter name dengan pengaturan akun yang sesuai.

- Menulis- ECSAccount Pengaturan (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-
east-1 -Force
```

Untuk mengubah pengaturan akun untuk akun pengguna Anda (AWS CLI)

Gunakan salah satu perintah berikut untuk mengubah pengaturan akun untuk pengguna Anda. Jika Anda menggunakan perintah ini sebagai pengguna root, perubahan berlaku untuk seluruh AWS akun kecuali; pengguna atau peran secara eksplisit mengesampingkan pengaturan ini untuk diri mereka sendiri.

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --region us-
east-1
```

  Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter name dengan pengaturan akun yang sesuai.

- Menulis- ECSAccount Pengaturan (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

Untuk mengubah pengaturan akun untuk pengguna atau peran tertentu (AWS CLI)

Gunakan salah satu perintah berikut dan tentukan ARN pengguna, peran, atau pengguna root dalam permintaan untuk mengubah pengaturan akun untuk pengguna atau peran tertentu.

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --principal-
arn arn:aws:iam::aws_account_id:user/principalName --region us-east-1
```

  Anda juga dapat menggunakan perintah ini untuk mengubah pengaturan akun lainnya. Untuk melakukan hal ini, ganti parameter `name` dengan pengaturan akun yang sesuai.

- Menulis- ECSAccount Pengaturan (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

Peran IAM untuk Amazon ECS

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Di Amazon ECS, Anda dapat membuat peran untuk memberikan izin ke sumber daya Amazon ECS seperti kontainer atau layanan.

Peran yang diperlukan Amazon ECS bergantung pada jenis peluncuran definisi tugas dan fitur yang Anda gunakan. Gunakan tabel berikut untuk menentukan peran IAM mana yang Anda butuhkan untuk Amazon ECS.

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran pelaksanaan tugas	Peran ini memungkinkan Amazon ECS untuk menggunakan AWS layanan lain atas nama Anda.	Tugas Anda di-host pada AWS Fargateatau pada instans eksternal dan: • menarik gambar kontainer dari	Peran IAM eksekusi tugas Amazon ECS

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
		repositori pribadi Amazon ECR. • menarik gambar kontainer dari repositori pribadi Amazon ECR di akun yang berbeda dari akun yang menjalankan tugas. • mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log. Tugas Anda di-host di salah satu AWS Fargate atau EC2 instans Amazon dan: • menggunakan otentikasi registri pribadi. • menggunakan Runtime Monitoring. • definisi tugas mereferensikan data sensitif menggunakan rahasia Secrets Manager atau parameter AWS	

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
		Systems Manager Parameter Store.	
role tugas	Peran ini memungkinkan kode aplikasi Anda (pada wadah) untuk menggunakan AWS layanan lain.	Aplikasi Anda mengakses AWS layanan lain, seperti Amazon S3.	Peran IAM tugas Amazon ECS
role instans kontainer	Peran ini memungkinkan EC2 instance atau instans eksternal Anda untuk mendaftar dengan cluster.	Tugas Anda di-host di EC2 instans Amazon atau instans eksternal.	Peran IAM instance wadah Amazon ECS
Peran Amazon ECS Anywhere	Peran ini memungkinkan instans eksternal Anda untuk mengakses AWS APIs.	Tugas Anda di-host pada instance eksternal.	Peran IAM Amazon ECS Anywhere
Infrastruktur Amazon ECS untuk peran penyeimbang beban	Peran ini memungkinkan Amazon ECS mengelola sumber daya penyeimbang beban di klaster Anda atas nama Anda untuk penerapan. blue/green	Anda ingin menggunakan blue/green penerapan Amazon ECS.	Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban
Peran Amazon ECS CodeDeploy	Peran ini memungkinkan CodeDeploy untuk membuat pembaruan pada layanan Anda.	Anda menggunakan jenis penyebaran CodeDeploy biru/hijau untuk menyebarkan layanan.	Peran Amazon ECS CodeDeploy IAM

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran Amazon ECS EventBridge	Peran ini memungkinkan EventBridge untuk membuat pembaruan pada layanan Anda.	Anda menggunakan EventBridge aturan dan target untuk menjadwalkan tugas Anda.	Peran Amazon ECS EventBridge IAM
Peran infrastruktur Amazon ECS	Peran ini memungkinkan Amazon ECS mengelola sumber daya infrastruktur di cluster Anda.	• Anda ingin melampirkan volume Amazon EBS ke Fargate Anda EC2 atau meluncurkan tugas Amazon ECS jenis. Peran infrastruktur memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas Anda. • Anda ingin menggunakan Transport Layer Security (TLS) untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda. • Anda ingin membuat grup target VPC Lattice.	Peran IAM infrastruktur Amazon ECS

Definisi tugas Amazon ECS

Definisi tugas adalah cetak biru untuk aplikasi Anda. Ini adalah file teks dalam format JSON yang menjelaskan parameter dan satu atau lebih wadah yang membentuk aplikasi Anda.

Berikut ini adalah beberapa parameter yang dapat Anda tentukan dalam definisi tugas:

- Jenis peluncuran yang akan digunakan, yang menentukan infrastruktur tempat tugas Anda di-host
- Gambar Docker untuk digunakan dengan setiap kontainer dalam tugas Anda
- Berapa banyak CPU dan memori untuk digunakan dengan setiap tugas atau setiap kontainer dalam tugas
- Memori dan persyaratan CPU
- Sistem operasi wadah tempat tugas berjalan
- Mode jaringan Docker digunakan untuk kontainer dalam tugas.
- Konfigurasi pencatatan digunakan untuk tugas Anda
- Apakah tugas terus berjalan jika penampung selesai atau gagal
- Perintah yang dijalankan kontainer saat dimulai
- Setiap volume data yang digunakan dengan wadah dalam tugas
- Peran IAM yang digunakan tugas Anda

Untuk daftar lengkap parameter definisi tugas, lihatParameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate.

Setelah Anda membuat definisi tugas, Anda dapat menjalankan definisi tugas sebagai tugas atau layanan.

- tugas adalah penunjukan hal abstrak dengan instans yang konkret dari ketentuan tugas dalam sebuah klaster. Setelah Anda membuat definisi tugas untuk aplikasi Anda dalam Amazon ECS, Anda dapat menentukan jumlah tugas yang akan dijalankan di klaster Anda.
- Layanan Amazon ECS menjalankan dan mempertahankan jumlah tugas yang Anda inginkan secara bersamaan di kluster Amazon ECS. Cara kerjanya adalah, jika salah satu tugas Anda gagal atau berhenti karena alasan apa pun, penjadwal layanan Amazon ECS meluncurkan instance lain berdasarkan definisi tugas Anda. Ia melakukan ini untuk menggantikannya dan dengan demikian mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Topik

- Status definisi tugas Amazon ECS
- Arsitek aplikasi Anda untuk Amazon ECS
- Membuat definisi tugas Amazon ECS menggunakan konsol
- Memperbarui definisi tugas Amazon ECS menggunakan konsol
- Membatalkan pendaftaran revisi definisi tugas Amazon ECS menggunakan konsol
- Menghapus revisi definisi tugas Amazon ECS menggunakan konsol
- Kasus penggunaan definisi tugas Amazon ECS
- Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate
- Parameter definisi tugas Amazon ECS untuk jenis EC2 peluncuran
- Templat definisi tugas Amazon ECS
- Contoh definisi tugas Amazon ECS

Status definisi tugas Amazon ECS

Definisi tugas mengubah status saat Anda membuat, membatalkan pendaftaran, atau menghapusnya. Anda dapat melihat status definisi tugas di konsol, atau dengan menggunakan`DescribeTaskDefinition`.

Berikut ini adalah status yang mungkin untuk definisi tugas:

AKTIF

Definisi tugas adalah `ACTIVE` setelah terdaftar di Amazon ECS. Anda dapat menggunakan definisi tugas di `ACTIVE` negara bagian untuk menjalankan tugas, atau membuat layanan.

TIDAK AKTIF

Definisi tugas bertransisi dari `ACTIVE` status ke `INACTIVE` status saat Anda membatalkan pendaftaran definisi tugas. Anda dapat mengambil definisi `INACTIVE` tugas dengan meneleponDescribeTaskDefinition. Anda tidak dapat menjalankan tugas baru atau membuat layanan baru dengan definisi tugas di `INACTIVE` negara bagian. Tidak ada dampak pada layanan atau tugas yang ada.

DELETE_IN_PROGRESS

Definisi tugas transisi dari `INACTIVE` status ke `DELETE_IN_PROGRESS` status setelah Anda mengirimkan definisi tugas untuk dihapus. Setelah definisi tugas dalam `DELETE_IN_PROGRESS`

status, Amazon ECS secara berkala memverifikasi bahwa definisi tugas target tidak direferensikan oleh tugas atau penerapan aktif apa pun, lalu menghapus definisi tugas secara permanen. Anda tidak dapat menjalankan tugas baru atau membuat layanan baru dengan definisi tugas di `DELETE_IN_PROGRESS` negara bagian. Definisi tugas dapat dikirimkan untuk dihapus kapan saja tanpa memengaruhi tugas dan layanan yang ada.

Definisi tugas yang berada dalam `DELETE_IN_PROGRESS` status dapat dilihat di konsol dan Anda dapat mengambil definisi tugas dengan memanggil`DescribeTaskDefinition`.

Saat Anda menghapus semua revisi definisi `INACTIVE` tugas, nama definisi tugas tidak ditampilkan di konsol dan tidak ditampilkan di API. Jika revisi definisi tugas dalam `DELETE_IN_PROGRESS` status, nama definisi tugas ditampilkan di konsol dan dikembalikan di API. Nama definisi tugas dipertahankan oleh Amazon ECS dan revisi bertambah saat berikutnya Anda membuat definisi tugas dengan nama itu.

Jika Anda menggunakan AWS Config untuk mengelola definisi tugas Anda, AWS Config menagih Anda untuk semua pendaftaran definisi tugas. Anda hanya dikenakan biaya untuk membatalkan pendaftaran definisi tugas terbaru`ACTIVE`. Tidak ada biaya untuk menghapus definisi tugas. Untuk informasi lebih lanjut tentang harga, lihat AWS Config Harga.

Sumber daya Amazon ECS yang dapat memblokir penghapusan

Permintaan penghapusan definisi tugas tidak akan selesai ketika ada sumber daya Amazon ECS yang bergantung pada revisi definisi tugas. Sumber daya berikut mungkin mencegah definisi tugas dihapus:

- Tugas mandiri Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Tugas layanan Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Penerapan layanan Amazon ECS dan set tugas - Definisi tugas diperlukan saat peristiwa penskalaan dimulai untuk penyebaran atau set tugas Amazon ECS.

Jika definisi tugas Anda tetap dalam `DELETE_IN_PROGRESS` status, Anda dapat menggunakan konsol, atau AWS CLI untuk mengidentifikasi, dan kemudian menghentikan sumber daya yang memblokir penghapusan definisi tugas.

Penghapusan definisi tugas setelah sumber daya yang diblokir dihapus

Aturan berikut berlaku setelah Anda menghapus sumber daya yang memblokir penghapusan definisi tugas:

- Tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 1 jam untuk diselesaikan setelah tugas dihentikan.
- Penyebaran layanan Amazon ECS dan set tugas - Penghapusan definisi tugas dapat memakan waktu hingga 24 jam untuk diselesaikan setelah penerapan atau set tugas dihapus.

Arsitek aplikasi Anda untuk Amazon ECS

Anda merancang aplikasi Anda dengan membuat definisi tugas untuk aplikasi Anda. Definisi tugas berisi parameter yang menentukan informasi tentang aplikasi, termasuk:

- Jenis peluncuran yang akan digunakan, yang menentukan infrastruktur tempat tugas Anda di-host.

  Saat Anda menggunakan tipe EC2 peluncuran, Anda juga memilih jenis instans. Untuk beberapa jenis contoh, seperti GPU, Anda perlu mengatur parameter tambahan. Untuk informasi selengkapnya, lihat Kasus penggunaan definisi tugas Amazon ECS.
- Gambar kontainer, yang menyimpan kode aplikasi Anda dan semua dependensi yang diperlukan kode aplikasi Anda untuk dijalankan.
- Mode jaringan yang digunakan untuk wadah dalam tugas Anda.

  Mode jaringan menentukan bagaimana tugas Anda berkomunikasi melalui jaringan.

  Untuk tugas yang berjalan pada EC2 instance, ada beberapa opsi, tetapi kami menyarankan Anda menggunakan mode `awsvpc` jaringan. Mode `awsvpc` jaringan menyederhanakan jaringan kontainer dengan memberi Anda lebih banyak kontrol atas bagaimana aplikasi Anda berkomunikasi satu sama lain dan layanan lain di dalam Anda VPCs.

  Untuk tugas yang berjalan di Fargate, Anda harus menggunakan mode `awsvpc` jaringan.
- Konfigurasi logging yang akan digunakan untuk tugas Anda.
- Setiap volume data yang digunakan dengan wadah dalam tugas.

Untuk daftar lengkap parameter definisi tugas, lihatParameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate.

Gunakan panduan berikut saat membuat definisi tugas:

- Gunakan setiap keluarga definisi tugas hanya untuk satu tujuan bisnis.

  Jika Anda mengelompokkan beberapa jenis wadah aplikasi bersama-sama dalam definisi tugas yang sama, Anda tidak dapat menskalakan kontainer tersebut secara independen. Misalnya, situs web dan API biasanya memerlukan pola penskalaan yang berbeda. Saat lalu lintas meningkat, Anda mungkin memerlukan jumlah kontainer web yang berbeda dari kontainer API. Jika kedua kontainer ini digunakan dalam definisi tugas yang sama, setiap tugas menjalankan jumlah kontainer web dan kontainer API yang sama.
- Cocokkan setiap versi aplikasi dengan revisi definisi tugas dalam keluarga definisi tugas.

  Dalam keluarga definisi tugas, setiap revisi definisi tugas mewakili point-in-time snapshot dari pengaturan untuk gambar kontainer tertentu. Ini mirip dengan bagaimana container adalah snapshot dari semua komponen yang diperlukan untuk menjalankan versi tertentu dari kode aplikasi Anda.

  Buat one-to-one pemetaan antara versi kode aplikasi, tag gambar kontainer, dan revisi definisi tugas. Proses rilis tipikal melibatkan git commit yang diubah menjadi image container yang ditandai dengan git commit SHA. Kemudian, tag gambar kontainer itu mendapatkan revisi definisi tugas Amazon ECS sendiri. Terakhir, layanan Amazon ECS diperbarui untuk menerapkan revisi definisi tugas baru.
- Gunakan peran IAM yang berbeda untuk setiap keluarga definisi tugas.

  Tentukan setiap definisi tugas dengan peran IAM-nya sendiri. Terapkan praktik ini bersama dengan menyediakan setiap komponen bisnis keluarga definisi tugasnya sendiri. Dengan menerapkan kedua praktik terbaik ini, Anda dapat membatasi seberapa banyak akses yang dimiliki setiap layanan ke sumber daya di AWS akun Anda. Misalnya, Anda dapat memberikan akses layanan otentikasi untuk terhubung ke database kata sandi Anda. Pada saat yang sama, Anda dapat memastikan bahwa hanya layanan pesanan Anda yang memiliki akses ke informasi pembayaran kartu kredit.

Praktik terbaik untuk gambar kontainer Amazon ECS

Gambar kontainer adalah seperangkat instruksi tentang cara membangun wadah. Gambar kontainer menyimpan kode aplikasi Anda dan semua dependensi yang diperlukan kode aplikasi Anda untuk dijalankan. Dependensi aplikasi mencakup paket kode sumber yang diandalkan oleh kode aplikasi

Anda, runtime bahasa untuk bahasa yang ditafsirkan, dan paket biner yang diandalkan oleh kode yang ditautkan secara dinamis.

Gunakan panduan berikut saat Anda mendesain dan membuat gambar kontainer Anda:

- Buat gambar kontainer Anda lengkap dengan menyimpan semua dependensi aplikasi sebagai file statis di dalam gambar kontainer.

  Jika Anda mengubah sesuatu dalam gambar kontainer, buat gambar kontainer baru dengan perubahan.
- Jalankan satu proses aplikasi dalam wadah.

  Masa pakai kontainer selama proses aplikasi berjalan. Amazon ECS menggantikan proses yang macet dan menentukan di mana harus meluncurkan proses penggantian. Gambar yang lengkap membuat penerapan keseluruhan lebih tangguh.
- Buat pegangan aplikasi Anda`SIGTERM`.

  Ketika Amazon ECS menghentikan tugas, pertama-tama ia mengirimkan sinyal SIGTERM ke tugas untuk memberi tahu aplikasi bahwa ia harus menyelesaikan dan mematikan. Amazon ECS kemudian mengirim `SIGKILL` pesan. Ketika aplikasi mengabaikan`SIGTERM`, layanan Amazon ECS harus menunggu untuk mengirim `SIGKILL` sinyal untuk menghentikan proses.

  Anda perlu mengidentifikasi berapa lama waktu yang dibutuhkan aplikasi Anda untuk menyelesaikan pekerjaannya, dan memastikan bahwa aplikasi Anda menangani `SIGTERM` sinyal. Penanganan sinyal aplikasi perlu menghentikan aplikasi dari mengambil pekerjaan baru dan menyelesaikan pekerjaan yang sedang berlangsung, atau menyimpan pekerjaan yang belum selesai ke penyimpanan di luar tugas ketika pekerjaan terlalu lama untuk diselesaikan.
- Konfigurasikan aplikasi kontainer untuk menulis log ke `stdout` dan. `stderr`

  Memisahkan penanganan log dari kode aplikasi Anda memberi Anda fleksibilitas untuk menyesuaikan penanganan log di tingkat infrastruktur. Salah satu contohnya adalah mengubah sistem logging Anda. Alih-alih memodifikasi layanan Anda, dan membangun dan menerapkan gambar kontainer baru, Anda dapat menyesuaikan pengaturan.
- Gunakan tag untuk membuat versi gambar kontainer Anda.

  Gambar kontainer disimpan dalam registri kontainer. Setiap gambar dalam registri diidentifikasi oleh tag. Ada tag yang disebut`latest`. Tag ini berfungsi sebagai penunjuk ke versi terbaru dari gambar wadah aplikasi, mirip dengan `HEAD` di repositori git. Kami menyarankan Anda

menggunakan `latest` tag hanya untuk tujuan pengujian. Sebagai praktik terbaik, beri tag gambar wadah dengan tag unik untuk setiap build. Kami menyarankan Anda menandai gambar Anda menggunakan git SHA untuk git commit yang digunakan untuk membangun gambar.

Anda tidak perlu membuat image kontainer untuk setiap komit. Namun, kami menyarankan Anda membuat image kontainer baru setiap kali Anda merilis kode tertentu yang berkomitmen ke lingkungan produksi. Kami juga menyarankan Anda menandai gambar dengan tag yang sesuai dengan git commit dari kode yang ada di dalam gambar. Jika Anda menandai gambar dengan git commit, Anda dapat lebih cepat menemukan versi kode mana yang sedang dijalankan gambar.

Kami juga menyarankan Anda mengaktifkan tag gambar yang tidak dapat diubah di Amazon Elastic Container Registry. Dengan pengaturan ini, Anda tidak dapat mengubah gambar kontainer yang ditunjuk tag. Sebaliknya Amazon ECR memberlakukan bahwa gambar baru harus diunggah ke tag baru. Untuk informasi selengkapnya, lihat Mutabilitas tag gambar di Panduan Pengguna Amazon ECR.

Saat Anda merancang aplikasi untuk dijalankan AWS Fargate, Anda harus memutuskan antara menerapkan beberapa kontainer ke dalam definisi tugas yang sama dan menerapkan kontainer secara terpisah dalam beberapa definisi tugas. Jika kondisi berikut diperlukan, sebaiknya gunakan beberapa kontainer ke dalam definisi tugas yang sama:

- Kontainer Anda berbagi siklus hidup yang sama (yaitu, mereka diluncurkan dan dihentikan bersama-sama).
- Kontainer Anda harus berjalan pada host dasar yang sama (yaitu, satu kontainer mereferensikan yang lain pada port localhost).
- Kontainer Anda berbagi sumber daya.
- Kontainer Anda berbagi volume data.

Jika kondisi ini tidak diperlukan, sebaiknya gunakan kontainer secara terpisah dalam beberapa definisi tugas. Ini memungkinkan Anda untuk menskalakan, menyediakan, dan menghentikan penyediaan wadah secara terpisah.

Praktik terbaik untuk ukuran tugas Amazon ECS

Ukuran wadah dan tugas Anda sangat penting untuk penskalaan dan perencanaan kapasitas. Di Amazon ECS, CPU dan memori adalah dua metrik sumber daya yang digunakan untuk kapasitas. CPU diukur dalam satuan 1/1024 dari vCPU penuh (di mana 1024 unit sama dengan 1 vCPU utuh).

Memori diukur dalam megabyte. Dalam definisi tugas, Anda dapat mengonfigurasi reservasi dan batasan sumber daya.

Saat mengonfigurasi reservasi, Anda menetapkan jumlah minimum sumber daya yang dibutuhkan tugas. Tugas Anda menerima setidaknya jumlah sumber daya yang diminta. Aplikasi Anda mungkin dapat menggunakan lebih banyak CPU atau memori daripada reservasi yang Anda deklarasikan. Namun, ini tunduk pada batasan apa pun yang juga Anda nyatakan. Menggunakan lebih dari jumlah reservasi dikenal sebagai bursting. Di Amazon ECS, reservasi dijamin. Misalnya, jika Anda menggunakan EC2 instans Amazon untuk menyediakan kapasitas, Amazon ECS tidak menempatkan tugas pada instance yang reservasi tidak dapat dipenuhi.

Batas adalah jumlah maksimum unit CPU atau memori yang dapat digunakan oleh wadah atau tugas Anda. Setiap upaya untuk menggunakan lebih banyak CPU lebih dari batas ini menghasilkan pelambatan. Setiap upaya untuk menggunakan lebih banyak memori menghasilkan penampung Anda dihentikan.

Memilih nilai-nilai ini bisa menjadi tantangan. Ini karena nilai-nilai yang paling cocok untuk aplikasi Anda sangat bergantung pada kebutuhan sumber daya aplikasi Anda. Pengujian beban aplikasi Anda adalah kunci keberhasilan perencanaan kebutuhan sumber daya dan lebih memahami persyaratan aplikasi Anda.

Aplikasi tanpa kewarganegaraan

Untuk aplikasi stateless yang menskalakan secara horizontal, seperti aplikasi di belakang penyeimbang beban, sebaiknya Anda terlebih dahulu menentukan jumlah memori yang dikonsumsi aplikasi Anda saat melayani permintaan. Untuk melakukan ini, Anda dapat menggunakan alat tradisional seperti `ps` atau`top`, atau solusi pemantauan seperti CloudWatch Wawasan Kontainer.

Saat menentukan reservasi CPU, pertimbangkan bagaimana Anda ingin menskalakan aplikasi Anda untuk memenuhi persyaratan bisnis Anda. Anda dapat menggunakan reservasi CPU yang lebih kecil, seperti 256 unit CPU (atau 1/4 vCPU), untuk meningkatkan skala dengan cara halus yang meminimalkan biaya. Tapi, mereka mungkin tidak berskala cukup cepat untuk memenuhi lonjakan permintaan yang signifikan. Anda dapat menggunakan reservasi CPU yang lebih besar untuk skala masuk dan keluar lebih cepat dan karenanya mencocokkan lonjakan permintaan lebih cepat. Namun, pemesanan CPU yang lebih besar lebih mahal.

Aplikasi lainnya

Untuk aplikasi yang tidak menskalakan secara horizontal, seperti pekerja tunggal atau server database, kapasitas dan biaya yang tersedia mewakili pertimbangan Anda yang paling penting.

Anda harus memilih jumlah memori dan CPU berdasarkan pengujian beban yang menunjukkan bahwa Anda perlu melayani lalu lintas untuk memenuhi tujuan tingkat layanan Anda. Amazon ECS memastikan bahwa aplikasi ditempatkan pada host yang memiliki kapasitas yang memadai.

Opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran

Perilaku jaringan tugas Amazon ECS yang di-host di EC2 instans Amazon bergantung pada mode jaringan yang ditentukan dalam definisi tugas. Kami menyarankan Anda menggunakan mode `awsvpc` jaringan kecuali Anda memiliki kebutuhan khusus untuk menggunakan mode jaringan yang berbeda.

Berikut ini adalah mode jaringan yang tersedia.

Mode jaringan	Kontainer Linux di EC2	Wadah Windows aktif EC2	Deskripsi
`awsvpc`	Ya	Ya	Tugas ini dialokasikan sendiri elastic network interface (ENI) dan IPv4 alamat pribadi utama. Ini memberi tugas properti jaringan yang sama dengan EC2 instans Amazon.
`bridge`	Ya	Tidak	Tugas ini menggunakan jaringan virtual bawaan Docker di Linux, yang berjalan di dalam setiap EC2 instans Amazon yang menghosting tugas tersebut. Jaringan virtual bawaan di Linux menggunakan driver jaringan `bridge` Docker. Ini adalah mode jaringan default di Linux jika mode jaringan tidak ditentukan dalam definisi tugas.
`host`	Ya	Tidak	Tugas menggunakan jaringan host yang melewati jaringan virtual bawaan Docker dengan memetakan port kontainer langsung ke ENI EC2 instance Amazon yang menghosti ng tugas tersebut. Pemetaan port dinamis tidak dapat digunakan dalam mode jaringan ini. Wadah dalam definisi tugas yang menggunak

Mode jaringan	Kontainer Linux di EC2	Wadah Windows aktif EC2	Deskripsi
			an mode ini harus menentukan `hostPort` nomor tertentu. Nomor port pada host tidak dapat digunakan oleh banyak tugas. Akibatnya, Anda tidak dapat menjalankan beberapa tugas dengan definisi tugas yang sama pada satu EC2 instans Amazon.
`none`	Ya	Tidak	Tugas tidak memiliki konektivitas jaringan eksternal.
`default`	Tidak	Ya	Tugas ini menggunakan jaringan virtual bawaan Docker di Windows, yang berjalan di dalam setiap EC2 instans Amazon yang menghosting tugas tersebut. Jaringan virtual bawaan pada Windows menggunakan driver jaringan `nat` Docker. Ini adalah mode jaringan default pada Windows jika mode jaringan tidak ditentukan dalam definisi tugas.

Untuk informasi selengkapnya tentang jaringan Docker di Linux, lihat Ikhtisar jaringan di Dokumentasi Docker.

Untuk informasi selengkapnya tentang jaringan Docker di Windows, lihat Jaringan kontainer Windows di Microsoft Container di Dokumentasi Windows.

Alokasikan antarmuka jaringan untuk tugas Amazon ECS

Fitur jaringan tugas yang disediakan oleh mode `awsvpc` jaringan memberikan tugas Amazon ECS properti jaringan yang sama dengan EC2 instans Amazon. Menggunakan mode `awsvpc` jaringan menyederhanakan jaringan kontainer, karena Anda memiliki kontrol lebih besar atas bagaimana aplikasi Anda berkomunikasi satu sama lain dan layanan lain di dalam Anda VPCs. Mode `awsvpc` jaringan juga memberikan keamanan yang lebih besar untuk wadah Anda dengan memungkinkan Anda menggunakan grup keamanan dan alat pemantauan jaringan pada tingkat yang lebih terperinci dalam tugas Anda. Anda juga dapat menggunakan fitur EC2 jaringan Amazon lainnya seperti VPC

Flow Logs untuk memantau lalu lintas ke dan dari tugas Anda. Selain itu, kontainer yang memiliki tugas yang sama dapat berkomunikasi melalui antarmuka `localhost`.

Task elastic network interface (ENI) adalah fitur Amazon ECS yang dikelola sepenuhnya. Amazon ECS membuat ENI dan menempelkannya ke EC2 instans Amazon host dengan grup keamanan yang ditentukan. Tugas mengirim dan menerima lalu lintas jaringan melalui ENI dengan cara yang sama seperti EC2 instans Amazon lakukan dengan antarmuka jaringan utama mereka. Setiap tugas ENI diberi IPv4 alamat pribadi secara default. Jika VPC Anda diaktifkan untuk mode dual-stack dan Anda menggunakan subnet dengan blok IPv6 CIDR, tugas ENI juga akan menerima alamat. IPv6 Setiap tugas hanya dapat memiliki satu ENI.

Ini ENIs terlihat di EC2 konsol Amazon untuk akun Anda. Akun Anda tidak dapat melepaskan atau memodifikasi. ENIs Hal ini untuk mencegah penghapusan ENI disengaja yang berkaitan dengan tugas yang sedang berjalan. Anda dapat melihat informasi lampiran ENI untuk tugas di konsol Amazon ECS atau dengan operasi DescribeTasksAPI. Ketika tugas berhenti atau jika layanan menurunkan skala, tugas ENI terlepas dan dihapus.

Saat Anda membutuhkan peningkatan kepadatan ENI, gunakan pengaturan `awsvpcTrunking` akun. Amazon ECS juga membuat dan melampirkan antarmuka jaringan “trunk” untuk instance container Anda. Jaringan trunk sepenuhnya dikelola oleh Amazon ECS. Trunk ENI akan dihapus saat Anda menghentikan atau membatalkan pendaftaran instance container Anda dari klaster Amazon ECS. Untuk informasi selengkapnya tentang pengaturan `awsvpcTrunking` akun, lihatPrasyarat.

Anda menentukan `awsvpc` dalam `networkMode` parameter definisi tugas. Untuk informasi selengkapnya, lihat Mode jaringan.

Kemudian, ketika Anda menjalankan tugas atau membuat layanan, gunakan `networkConfiguration` parameter yang menyertakan satu atau beberapa subnet untuk menempatkan tugas Anda dan satu atau lebih grup keamanan untuk dilampirkan ke ENI. Untuk informasi selengkapnya, lihat Konfigurasi jaringan. Tugas ditempatkan pada EC2 instans Amazon yang kompatibel di Availability Zone yang sama dengan subnet tersebut, dan grup keamanan yang ditentukan dikaitkan dengan ENI yang disediakan untuk tugas tersebut.

Pertimbangan Linux

Pertimbangkan hal berikut saat menggunakan sistem operasi Linux.

- Jika Anda menggunakan instance p5.48xlarge dalam `awsvpc` mode, Anda tidak dapat menjalankan lebih dari 1 tugas pada instance tersebut.

- Tugas dan layanan yang menggunakan mode `awsvpc` jaringan memerlukan peran terkait layanan Amazon ECS untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat klaster atau jika Anda membuat atau memperbarui layanan, di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut: AWS CLI

  ```
  aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
  ```

- Instans Amazon EC2 Linux Anda memerlukan versi `1.15.0` atau yang lebih baru dari agen penampung untuk menjalankan tugas yang menggunakan mode `awsvpc` jaringan. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memerlukan setidaknya `1.15.0-4` versi `ecs-init` paket juga.
- Amazon ECS mengisi nama host tugas dengan nama host DNS (internal) yang disediakan Amazon saat opsi dan `enableDnsHostnames` opsi `enableDnsSupport` diaktifkan di VPC Anda. Jika opsi ini tidak diaktifkan, nama host DNS tugas disetel ke nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.
- Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) sendiri, yang dilampirkan ke EC2 instance Amazon yang menghostingnya. Ada kuota default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke instance Amazon EC2 Linux. Antarmuka jaringan utama dihitung sebagai satu terhadap kuota itu. Misalnya, secara default, sebuah `c5.large` instance mungkin hanya memiliki hingga tiga ENIs yang dapat dilampirkan padanya. Antarmuka jaringan utama untuk instance dihitung sebagai satu. Anda dapat melampirkan dua tambahan ENIs ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukan ENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini. Untuk informasi selengkapnya tentang batas ENI default untuk setiap jenis instans, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan EC2 Pengguna Amazon.
- Amazon ECS mendukung peluncuran instans Amazon EC2 Linux yang menggunakan tipe instans yang didukung dengan peningkatan kepadatan ENI. Saat Anda memilih setelan `awsvpcTrunking` akun dan mendaftarkan instans Amazon EC2 Linux yang menggunakan jenis instans ini ke klaster Anda, instans ini memiliki kuota ENI yang lebih tinggi. Menggunakan instans ini dengan kuota yang lebih tinggi ini berarti Anda dapat menempatkan lebih banyak tugas di setiap instans Amazon EC2 Linux. Untuk menggunakan peningkatan kepadatan ENI dengan fitur trunking, EC2 instans Amazon Anda harus menggunakan agen penampung versi `1.28.1` atau yang lebih baru.

Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda juga memerlukan setidaknya `1.28.1-2` versi paket. `ecs-init` Untuk informasi lebih lanjut tentang penyertaan pada pengaturan akun `awsvpcTrunking`, lihat Akses fitur Amazon ECS dengan pengaturan akun. Untuk informasi lebih lanjut tentang trunking ENI, lihat. Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux

- Saat menghosting tugas yang menggunakan mode `awsvpc` jaringan di instans Amazon EC2 Linux, tugas Anda ENIs tidak diberikan alamat IP publik. Untuk mengakses internet, tugas harus diluncurkan di subnet pribadi yang dikonfigurasi untuk menggunakan gateway NAT. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC. Akses jaringan masuk harus dari dalam VPC yang menggunakan alamat IP pribadi atau dirutekan melalui penyeimbang beban dari dalam VPC. Tugas yang diluncurkan dalam subnet publik tidak memiliki akses ke internet.
- Amazon ECS hanya mengenali ENIs yang melekat pada instans Amazon EC2 Linux Anda. Jika Anda ENIs melampirkan instans secara manual, Amazon ECS mungkin mencoba menambahkan tugas ke instans yang tidak memiliki cukup adaptor jaringan. Hal ini dapat mengakibatkan waktu tugas habis dan pindah ke status deprovisioning dan kemudian status berhenti. Kami menyarankan Anda untuk tidak ENIs melampirkan instans Anda secara manual.
- Instans Amazon EC2 Linux harus terdaftar dengan `ecs.capability.task-eni` kemampuan yang harus dipertimbangkan untuk penempatan tugas dengan mode `awsvpc` jaringan. Instans yang menjalankan versi `1.15.0-4` atau yang lebih baru `ecs-init` terdaftar dengan atribut ini secara otomatis.
- ENIs Yang dibuat dan dilampirkan ke instans Amazon EC2 Linux Anda tidak dapat dilepaskan secara manual atau dimodifikasi oleh akun Anda. Hal ini untuk mencegah menghapus ENI disengaja yang berhubungan dengan tugas yang sedang berjalan. Untuk melepaskan tugas, hentikan tugas. ENIs
- Ada batas 16 subnet dan 5 grup keamanan yang dapat ditentukan dalam `awsVpcConfiguration` saat menjalankan tugas atau membuat layanan yang menggunakan mode jaringan `awsvpc`. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.
- Saat tugas dimulai dengan mode `awsvpc` jaringan, agen penampung Amazon ECS membuat `pause` wadah tambahan untuk setiap tugas sebelum memulai kontainer dalam definisi tugas. Kemudian mengkonfigurasi namespace jaringan `pause` wadah dengan menjalankan plugin CNI. amazon-ecs-cni-plugins Agen kemudian memulai sisa kontainer dalam tugas sehingga mereka berbagi tumpukan jaringan kontainer `pause`. Ini berarti bahwa semua kontainer dalam tugas yang

dialamatkan oleh alamat IP dari ENI, dan mereka dapat berkomunikasi satu sama lain melalui antarmuka `localhost`.

- Layanan dengan tugas yang menggunakan mode `awsvpc` jaringan hanya mendukung Application Load Balancer dan Network Load Balancer. Saat Anda membuat grup target apa pun untuk layanan ini, Anda harus memilih `ip` sebagai jenis target. Jangan gunakan `instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan ENI, bukan dengan instance Amazon EC2 Linux. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.
- Jika VPC Anda diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda tidak dapat menerapkan perubahan ini ke tugas yang ada. Mulai tugas baru dengan perubahan ini diterapkan padanya, verifikasi bahwa mereka berfungsi dengan benar, dan kemudian hentikan tugas yang ada untuk mengubah konfigurasi jaringan ini dengan aman.

Pertimbangan Windows

Berikut ini adalah pertimbangan ketika Anda menggunakan sistem operasi Windows:

- Instans kontainer menggunakan Amazon ECS yang dioptimalkan Windows Server 2016 AMI tidak dapat meng-host tugas yang menggunakan mode `awsvpc` jaringan. Jika Anda memiliki klaster yang berisi Amazon ECS yang dioptimalkan Windows Server 2016 AMIs dan Windows AMIs yang mendukung mode `awsvpc` jaringan, tugas yang menggunakan mode `awsvpc` jaringan tidak diluncurkan pada instance Windows 2016 Server. Sebaliknya, mereka diluncurkan pada instance yang mendukung mode `awsvpc` jaringan.
- Instans Amazon EC2 Windows Anda memerlukan versi `1.57.1` atau yang lebih baru dari agen penampung untuk menggunakan CloudWatch metrik untuk kontainer Windows yang menggunakan mode `awsvpc` jaringan.
- Tugas dan layanan yang menggunakan mode `awsvpc` jaringan memerlukan peran terkait layanan Amazon ECS untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat klaster, atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut AWS CLI .

  ```
  aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
  ```

- Instans Amazon EC2 Windows Anda memerlukan versi `1.54.0` atau yang lebih baru dari agen penampung untuk menjalankan tugas yang menggunakan mode `awsvpc` jaringan. Saat Anda

mem-bootstrap instance, Anda harus mengonfigurasi opsi yang diperlukan untuk mode `awsvpc` jaringan. Untuk informasi selengkapnya, lihat the section called “Contoh wadah bootstrapping”.

- Amazon ECS mengisi nama host tugas dengan nama host DNS (internal) yang disediakan Amazon saat `enableDnsSupport` opsi `enableDnsHostnames` dan opsi diaktifkan di VPC Anda. Jika opsi ini tidak diaktifkan, nama host DNS tugas adalah nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.
- Setiap tugas Amazon ECS yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) sendiri, yang dilampirkan ke instance Amazon EC2 Windows yang menghostingnya. Ada kuota default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke instance Amazon EC2 Windows. Antarmuka jaringan utama dihitung sebagai satu terhadap kuota ini. Misalnya, secara default sebuah `c5.large` instance mungkin hanya memiliki hingga tiga yang ENIs melekat padanya. Antarmuka jaringan utama untuk instance dihitung sebagai salah satunya. Anda dapat melampirkan dua tambahan ENIs ke instance. Karena setiap tugas menggunakan mode jaringan `awsvpc` memerlukan ENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada Tipe instans ini. Untuk informasi selengkapnya tentang batas ENI default untuk setiap jenis instans, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan EC2 Pengguna Amazon.
- Saat menghosting tugas yang menggunakan mode `awsvpc` jaringan di instans Amazon EC2 Windows, tugas Anda ENIs tidak diberikan alamat IP publik. Untuk mengakses internet, luncurkan tugas di subnet pribadi yang dikonfigurasi untuk menggunakan gateway NAT. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC. Akses jaringan masuk harus dari dalam VPC yang menggunakan alamat IP pribadi atau dirutekan melalui penyeimbang beban dari dalam VPC. Tugas yang diluncurkan dalam subnet publik tidak memiliki akses ke internet.
- Amazon ECS hanya mengenali ENIs yang telah dilampirkan ke instans Amazon EC2 Windows Anda. Jika Anda ENIs melampirkan instans secara manual, Amazon ECS mungkin mencoba menambahkan tugas ke instans yang tidak memiliki cukup adaptor jaringan. Hal ini dapat mengakibatkan waktu tugas habis dan pindah ke status deprovisioning dan kemudian status berhenti. Kami menyarankan Anda untuk tidak ENIs melampirkan instans Anda secara manual.
- Instans Amazon EC2 Windows harus terdaftar dengan `ecs.capability.task-eni` kemampuan yang harus dipertimbangkan untuk penempatan tugas dengan mode `awsvpc` jaringan.
- Anda tidak dapat memodifikasi atau melepaskan secara manual ENIs yang dibuat dan dilampirkan ke instans Amazon EC2 Windows Anda. Ini untuk mencegah Anda secara tidak sengaja menghapus ENI yang terkait dengan tugas yang sedang berjalan. Untuk melepaskan tugas, hentikan tugas. ENIs

- Anda hanya dapat menentukan hingga 16 subnet dan 5 grup keamanan `awsVpcConfiguration` ketika Anda menjalankan tugas atau membuat layanan yang menggunakan mode `awsvpc` jaringan. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.
- Saat tugas dimulai dengan mode `awsvpc` jaringan, agen penampung Amazon ECS membuat `pause` wadah tambahan untuk setiap tugas sebelum memulai kontainer dalam definisi tugas. Kemudian mengkonfigurasi namespace jaringan `pause` wadah dengan menjalankan plugin CNI. amazon-ecs-cni-plugins Agen kemudian memulai sisa kontainer dalam tugas sehingga mereka berbagi tumpukan jaringan kontainer `pause`. Ini berarti bahwa semua kontainer dalam tugas yang dialamatkan oleh alamat IP dari ENI, dan mereka dapat berkomunikasi satu sama lain melalui antarmuka `localhost`.
- Layanan dengan tugas yang menggunakan mode `awsvpc` jaringan hanya mendukung Application Load Balancer dan Network Load Balancer. Saat Anda membuat grup target apa pun untuk layanan ini, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan ENI, bukan dengan instance Amazon EC2 Windows. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.
- Jika VPC Anda diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda tidak dapat menerapkan perubahan ini ke tugas yang ada. Mulai tugas baru dengan perubahan ini diterapkan padanya, verifikasi bahwa mereka berfungsi dengan benar, dan kemudian hentikan tugas yang ada untuk mengubah konfigurasi jaringan ini dengan aman.
- Berikut ini tidak didukung saat Anda menggunakan mode `awsvpc` jaringan dalam konfigurasi EC2 Windows:
  - Konfigurasikan tumpukan dobel
  - IPv6
  - ENI trunking

Menggunakan VPC dalam mode tumpukan dobel

Saat menggunakan VPC dalam mode dual-stack, tugas Anda dapat berkomunikasi melalui, atau IPv4, atau IPv6 keduanya. IPv4 dan IPv6 alamat independen satu sama lain. Oleh karena itu Anda harus mengonfigurasi perutean dan keamanan di VPC Anda secara terpisah IPv4 untuk dan. IPv6 Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode tumpukan ganda, lihat Memigrasi ke dalam Panduan Pengguna VPC IPv6 Amazon.

Jika Anda mengonfigurasi VPC Anda dengan gateway internet atau gateway internet khusus keluar, Anda dapat menggunakan VPC Anda dalam mode dual-stack. Dengan melakukan ini, tugas yang diberi IPv6 alamat dapat mengakses internet melalui gateway internet atau gateway internet khusus egres. Gateway NAT bersifat opsional. Untuk informasi selengkapnya, lihat gateway Internet dan gateway internet khusus eGress di Panduan Pengguna Amazon VPC.

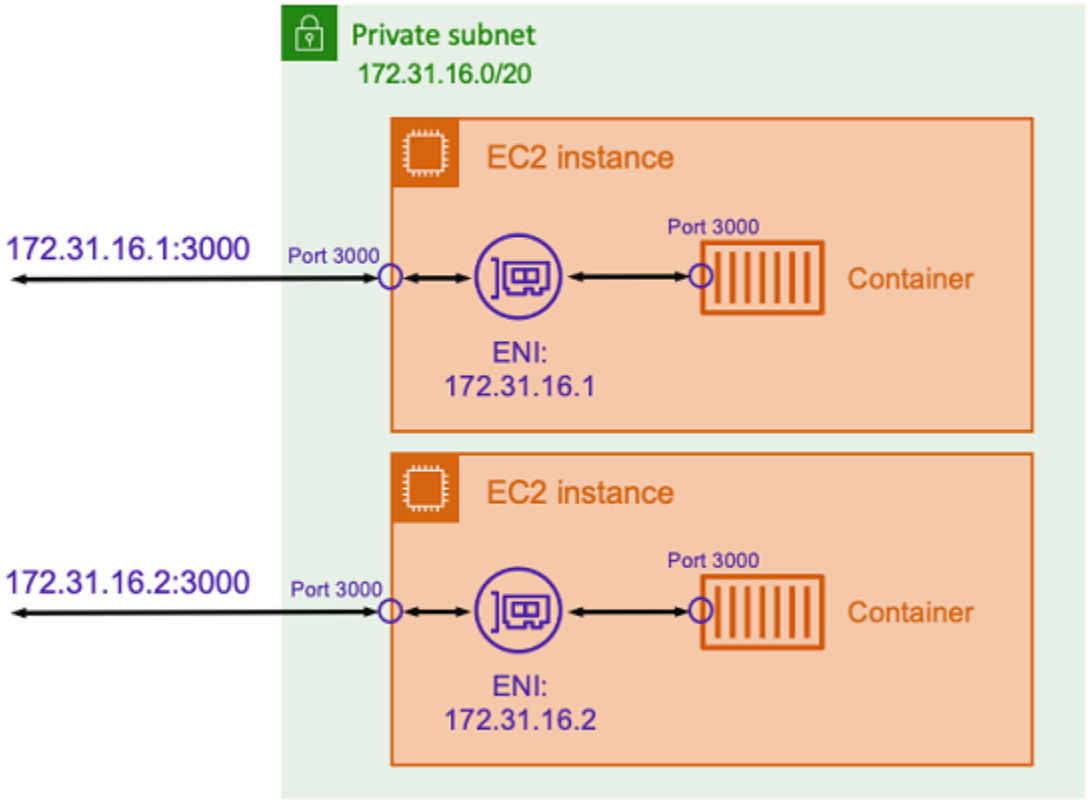
Tugas Amazon ECS diberikan IPv6 alamat jika kondisi berikut terpenuhi:

- Instance Amazon EC2 Linux yang menghosting tugas menggunakan versi `1.45.0` atau yang lebih baru dari agen kontainer. Untuk informasi tentang cara memeriksa versi agen yang digunakan instans Anda, dan memperbaruinya jika diperlukan, lihatMemperbarui agen kontainer Amazon ECS.
- Pengaturan akun `dualStackIPv6` diaktifkan. Untuk informasi selengkapnya, lihat Akses fitur Amazon ECS dengan pengaturan akun.
- Tugas Anda adalah menggunakan mode jaringan `awsvpc`.
- VPC dan subnet Anda dikonfigurasi untuk. IPv6 Konfigurasi mencakup antarmuka jaringan yang dibuat di subnet yang ditentukan. Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode dual-stack, lihat Memigrasi ke IPv6 dan Memodifikasi atribut IPv6 pengalamatan untuk subnet Anda di Panduan Pengguna Amazon VPC.

Petakan port kontainer Amazon ECS ke antarmuka jaringan EC2 instans

Mode `host` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di EC2 instans Amazon. Ini tidak didukung saat menggunakan Amazon ECS di Fargate.

Mode `host` jaringan adalah mode jaringan paling dasar yang didukung di Amazon ECS. Menggunakan mode host, jaringan kontainer terikat langsung ke host yang mendasari yang menjalankan wadah.



Asumsikan bahwa Anda menjalankan kontainer Node.js dengan aplikasi Express yang mendengarkan pada port yang `3000` mirip dengan yang diilustrasikan dalam diagram sebelumnya. Ketika mode `host` jaringan digunakan, kontainer menerima lalu lintas pada port 3000 menggunakan alamat IP dari EC2 instance Amazon host yang mendasarinya. Kami tidak menyarankan menggunakan mode ini.

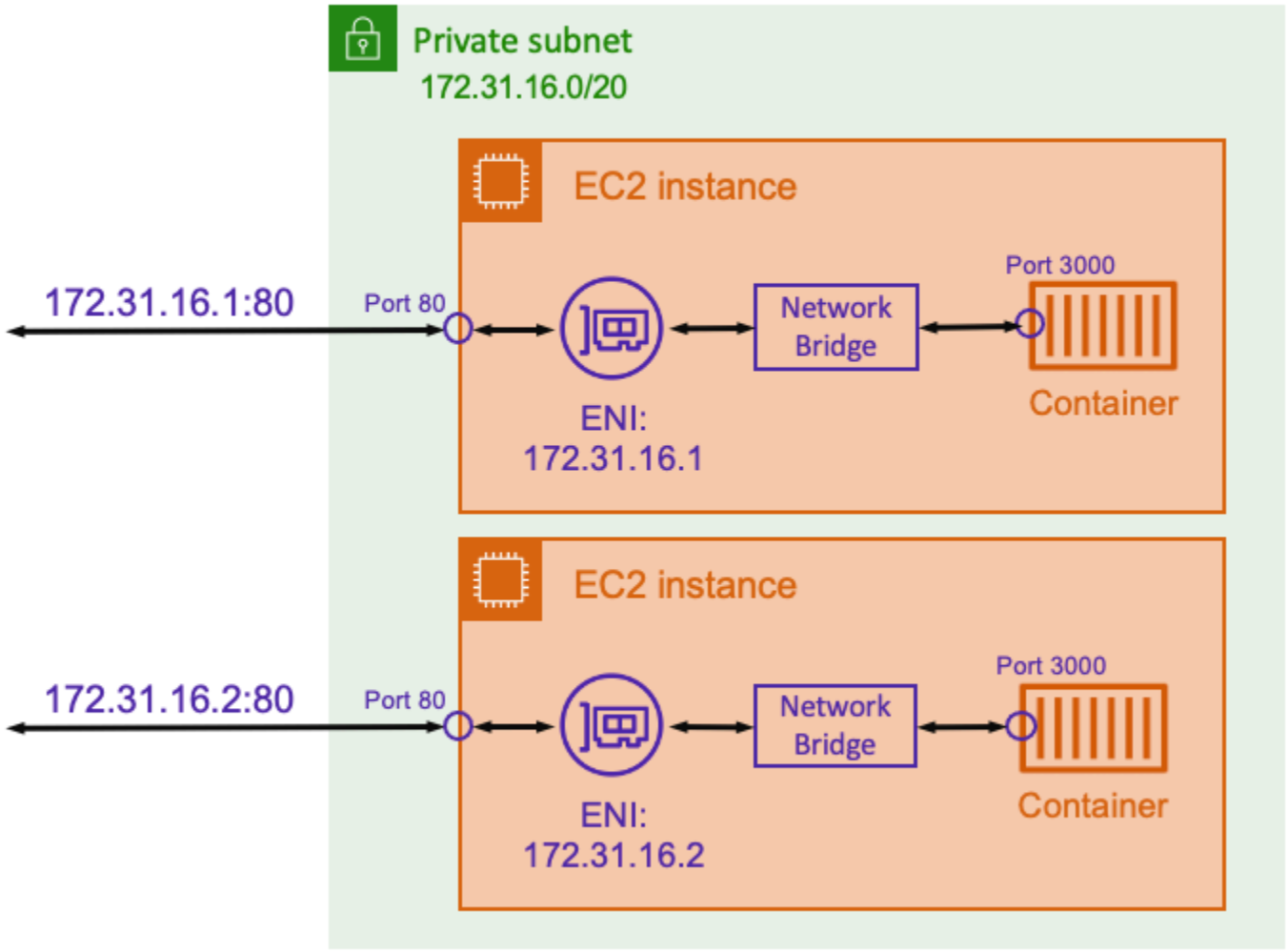
Ada kelemahan signifikan untuk menggunakan mode jaringan ini. Anda tidak dapat menjalankan lebih dari satu instantiasi tugas pada setiap host. Ini karena hanya tugas pertama yang dapat mengikat ke port yang diperlukan pada EC2 instance Amazon. Juga tidak ada cara untuk memetakan ulang port kontainer saat menggunakan mode `host` jaringan. Misalnya, jika aplikasi perlu mendengarkan nomor port tertentu, Anda tidak dapat memetakan ulang nomor port secara langsung. Sebagai gantinya, Anda harus mengelola konflik port apa pun dengan mengubah konfigurasi aplikasi.

Ada juga implikasi keamanan saat menggunakan mode `host` jaringan. Mode ini memungkinkan kontainer untuk meniru host, dan memungkinkan kontainer untuk terhubung ke layanan jaringan loopback pribadi pada host.

Gunakan jaringan virtual Docker untuk tugas-tugas Amazon ECS Linux

Mode `bridge` jaringan hanya didukung untuk tugas Amazon ECS yang dihosting di EC2 instans Amazon.

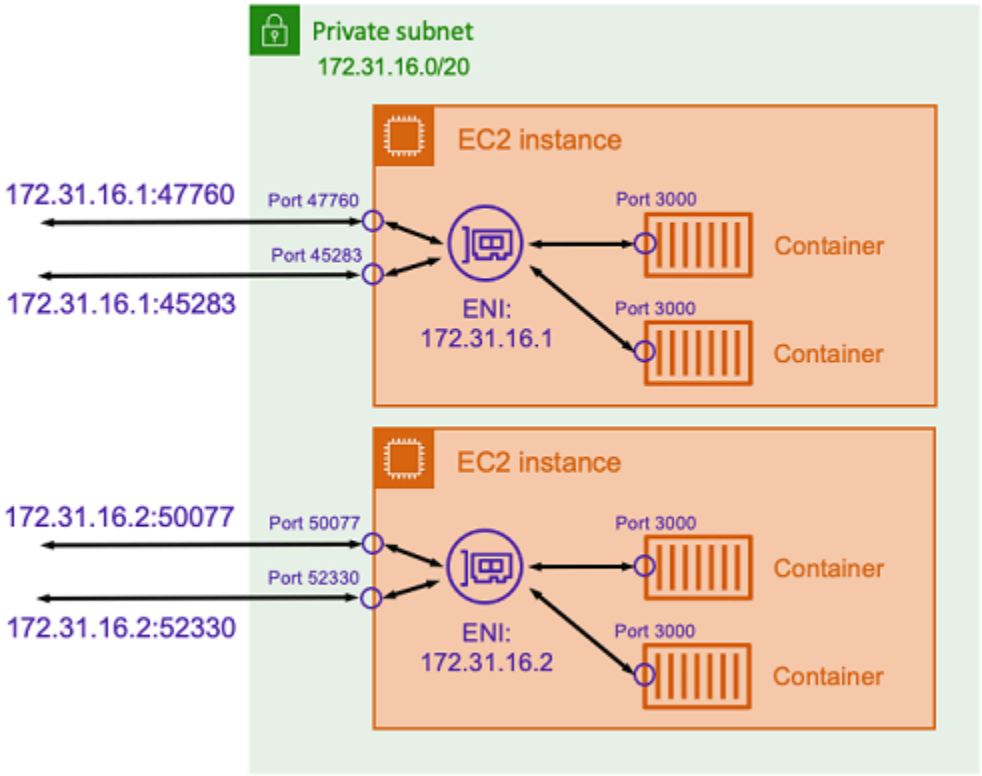
Dengan `bridge` mode, Anda menggunakan jembatan jaringan virtual untuk membuat lapisan antara host dan jaringan wadah. Dengan cara ini, Anda dapat membuat pemetaan port yang memetakan ulang port host ke port kontainer. Pemetaan dapat berupa statis atau dinamis.



Dengan pemetaan port statis, Anda dapat secara eksplisit menentukan port host mana yang ingin Anda petakan ke port kontainer. Menggunakan contoh di atas, port `80` pada host sedang dipetakan ke port `3000` pada container. Untuk berkomunikasi dengan aplikasi kontainer, Anda mengirim lalu lintas ke port `80` ke alamat IP EC2 instans Amazon. Dari perspektif aplikasi kontainer, ia melihat lalu lintas masuk di port. `3000`

Jika Anda hanya ingin mengubah port lalu lintas, maka pemetaan port statis cocok. Namun, ini masih memiliki kelemahan yang sama dengan menggunakan mode `host` jaringan. Anda tidak dapat menjalankan lebih dari satu instantiasi tugas pada setiap host. Ini karena pemetaan port statis hanya memungkinkan satu kontainer untuk dipetakan ke port 80.

Untuk mengatasi masalah ini, pertimbangkan untuk menggunakan mode `bridge` jaringan dengan pemetaan port dinamis seperti yang ditunjukkan pada diagram berikut.



Dengan tidak menentukan port host dalam pemetaan port, Anda dapat meminta Docker memilih port acak yang tidak digunakan dari rentang port sementara dan menetapkannya sebagai port host publik untuk wadah. Misalnya, aplikasi Node.js yang mendengarkan pada port `3000` pada kontainer mungkin diberi port nomor tinggi acak seperti `47760` pada EC2 host Amazon. Melakukan ini berarti Anda dapat menjalankan beberapa salinan wadah itu di host. Selain itu, setiap kontainer dapat diberi port sendiri di host. Setiap salinan kontainer menerima lalu lintas di pelabuhan`3000`. Namun, klien yang mengirim lalu lintas ke kontainer ini menggunakan port host yang ditetapkan secara acak.

Amazon ECS membantu Anda melacak port yang ditetapkan secara acak untuk setiap tugas. Hal ini dilakukan dengan secara otomatis memperbarui kelompok target load balancer dan penemuan AWS Cloud Map layanan untuk memiliki daftar alamat IP tugas dan port. Ini membuatnya lebih mudah untuk menggunakan layanan yang beroperasi menggunakan `bridge` mode dengan port dinamis.

Namun, salah satu kelemahan menggunakan mode `bridge` jaringan adalah sulit untuk mengunci komunikasi layanan ke layanan. Karena layanan mungkin ditetapkan ke port acak dan tidak terpakai, maka perlu untuk membuka rentang port yang luas antar host. Namun, tidak mudah untuk membuat aturan khusus sehingga layanan tertentu hanya dapat berkomunikasi dengan satu layanan spesifik lainnya. Layanan tidak memiliki port khusus untuk digunakan untuk aturan jaringan grup keamanan.

Opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate

Secara default, setiap tugas Amazon ECS di Fargate disediakan elastic network interface (ENI) dengan alamat IP pribadi utama. Saat menggunakan subnet publik, Anda dapat secara opsional menetapkan alamat IP publik ke ENI tugas. Jika VPC Anda dikonfigurasi untuk mode dual-stack dan Anda menggunakan subnet dengan blok IPv6 CIDR, ENI tugas Anda juga menerima alamat. IPv6 Sebuah tugas hanya dapat memiliki satu ENI yang terkait dengannya pada suatu waktu. Wadah yang termasuk dalam tugas yang sama juga dapat berkomunikasi melalui `localhost` antarmuka. Untuk informasi selengkapnya tentang VPCs dan subnet, lihat Cara kerja Amazon VPC di Panduan Pengguna Amazon VPC.

Untuk tugas di Fargate untuk menarik gambar kontainer, tugas harus memiliki rute ke internet. Berikut ini menjelaskan bagaimana Anda dapat memverifikasi bahwa tugas Anda memiliki rute ke internet.

- Saat menggunakan subnet publik, Anda dapat menetapkan alamat IP publik ke ENI tugas.
- Saat menggunakan subnet privat, subnet bisa memiliki lampiran gateway NAT.
- Saat menggunakan gambar kontainer yang di-host di Amazon ECR, Anda dapat mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka dan penarikan gambar terjadi di atas alamat pribadi tugas. IPv4 Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.

Karena setiap tugas mendapatkan ENI sendiri, Anda dapat menggunakan fitur jaringan seperti VPC Flow Logs, yang dapat Anda gunakan untuk memantau lalu lintas ke dan dari tugas Anda. Untuk informasi selengkapnya, lihat Log Alur VPC di Panduan Pengguna Amazon VPC.

Anda juga bisa memanfaatkannya AWS PrivateLink. Anda dapat mengonfigurasi titik akhir antarmuka VPC sehingga Anda dapat mengakses Amazon ECS APIs melalui alamat IP pribadi. AWS PrivateLink membatasi semua lalu lintas jaringan antara VPC Anda dan Amazon ECS ke jaringan Amazon. Anda tidak memerlukan sebuah gateway internet, perangkat NAT, atau gateway privat virtual. Untuk informasi selengkapnya, lihat Titik akhir AWS PrivateLink VPC antarmuka Amazon ECS ().

Untuk contoh cara menggunakan `NetworkConfiguration` sumber daya AWS CloudFormation, lihatthe section called “Contoh AWS CloudFormation template”.

ENIs Yang dibuat sepenuhnya dikelola oleh AWS Fargate. Selain itu, ada kebijakan IAM terkait yang digunakan untuk memberikan izin untuk Fargate. Untuk tugas yang menggunakan versi platform Fargate `1.4.0` atau yang lebih baru, tugas menerima ENI tunggal (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda. Lalu lintas ini direkam dalam log aliran VPC Anda. Untuk tugas yang menggunakan versi platform Fargate `1.3.0` dan sebelumnya, selain tugas ENI, tugas juga menerima ENI milik Fargate terpisah, yang digunakan untuk beberapa lalu lintas jaringan yang tidak terlihat di log aliran VPC. Tabel berikut menjelaskan perilaku lalu lintas jaringan dan kebijakan IAM yang diperlukan untuk setiap versi platform.

Tindakan	Arus lalu lintas dengan versi platform Linux **`1.3.0`** dan sebelumnya	Arus lalu lintas dengan versi platform Linux **`1.4.0`**	Arus lalu lintas dengan versi platform Windows **`1.0.0`**	Izin IAM
Mengambil kredensyal login Amazon ECR	Fargate dimiliki ENI	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Tarikan citra	ENI tugas	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Mengirim log melalui driver log	ENI tugas	ENI tugas	ENI tugas	Eksekusi tugas peran IAM
Mengirim log melalui FireLens Amazon ECS	ENI tugas	ENI tugas	ENI tugas	Tugas peran IAM
Mengambil rahasia dari Secrets Manager atau Systems Manager	Fargate dimiliki ENI	ENI tugas	ENI tugas	Eksekusi tugas peran IAM

Tindakan	Arus lalu lintas dengan versi platform Linux **1.3.0** dan sebelumnya	Arus lalu lintas dengan versi platform Linux **1.4.0**	Arus lalu lintas dengan versi platform Windows **1.0.0**	Izin IAM
Lalu lintas sistem file Amazon EFS	Tidak tersedia	ENI tugas	ENI tugas	Tugas peran IAM
Lalu lintas aplikasi	ENI tugas	ENI tugas	ENI tugas	Tugas peran IAM

Pertimbangan

Pertimbangkan hal berikut saat menggunakan jaringan tugas.

- Peran terkait layanan Amazon ECS diperlukan untuk memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda ketika Anda membuat klaster atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan menggunakan perintah berikut AWS CLI .

  ```
  aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
  ```

- Amazon ECS mengisi nama host tugas dengan nama host DNS yang disediakan Amazon saat `enableDnsSupport` opsi `enableDnsHostnames` dan opsi diaktifkan di VPC Anda. Jika opsi ini tidak diaktifkan, nama host DNS tugas disetel ke nama host acak. Untuk informasi selengkapnya tentang pengaturan DNS untuk VPC, lihat Menggunakan DNS dengan VPC Anda di Panduan Pengguna Amazon VPC.
- Anda hanya dapat menentukan hingga 16 subnet dan 5 grup keamanan untuk`awsVpcConfiguration`. Untuk informasi selengkapnya, lihat AwsVpcConfigurationdi Referensi API Amazon Elastic Container Service.
- Anda tidak dapat secara manual melepaskan atau memodifikasi ENIs yang dibuat dan dilampirkan oleh Fargate. Ini untuk mencegah penghapusan ENI yang tidak disengaja yang terkait dengan tugas yang sedang berjalan. Untuk melepaskan tugas, hentikan tugas. ENIs
- Jika subnet VPC diperbarui untuk mengubah set opsi DHCP yang digunakannya, Anda juga tidak dapat menerapkan perubahan ini ke tugas yang ada yang menggunakan VPC. Mulai tugas baru,

yang akan menerima pengaturan baru untuk bermigrasi dengan lancar saat menguji perubahan baru dan kemudian menghentikan yang lama, jika tidak diperlukan rollback.

- Berikut ini berlaku untuk tugas yang dijalankan pada versi platform Fargate `1.4.0` atau yang lebih baru untuk Linux atau `1.0.0` untuk Windows. Tugas yang diluncurkan dalam subnet dual-stack menerima IPv4 alamat dan alamat. IPv6
- Untuk tugas yang menggunakan versi platform `1.4.0` atau yang lebih baru untuk Linux atau `1.0.0` Windows, tugas ENIs mendukung bingkai jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Mendukung bingkai jumbo mengurangi overhead saat jalur jaringan antara tugas Anda dan tujuan mendukung bingkai jumbo.
- Layanan dengan tugas yang menggunakan tipe peluncuran Fargate hanya mendukung Application Load Balancer dan Network Load Balancer. Classic Load Balancer tidak didukung. Saat Anda membuat grup target apa pun, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Menggunakan VPC dalam mode tumpukan dobel

Saat menggunakan VPC dalam mode dual-stack, tugas Anda dapat berkomunikasi melalui IPv4 atau, atau IPv6 keduanya. IPv4 dan IPv6 alamat independen satu sama lain dan Anda harus mengonfigurasi perutean dan keamanan di VPC Anda secara terpisah IPv4 untuk dan. IPv6 Untuk informasi selengkapnya tentang mengonfigurasi VPC Anda untuk mode tumpukan ganda, lihat Memigrasi ke IPv6 dalam Panduan Pengguna VPC Amazon.

Jika kondisi berikut terpenuhi, tugas Amazon ECS di Fargate diberi IPv6 alamat:

- Setelan `dualStackIPv6` akun Amazon ECS Anda diaktifkan (`enabled`) untuk prinsipal IAM yang meluncurkan tugas Anda di Wilayah tempat Anda meluncurkan tugas. Pengaturan ini hanya dapat dimodifikasi menggunakan API atau AWS CLI. Anda memiliki opsi untuk mengaktifkan pengaturan ini untuk prinsipal IAM tertentu di akun Anda atau untuk seluruh akun Anda dengan menetapkan pengaturan default akun Anda. Untuk informasi selengkapnya, lihat Akses fitur Amazon ECS dengan pengaturan akun.
- VPC dan subnet Anda diaktifkan untuk. IPv6 Untuk informasi selengkapnya tentang cara mengonfigurasi VPC Anda untuk mode tumpukan ganda, lihat Memigrasi ke dalam Panduan Pengguna VPC IPv6 Amazon.

- Subnet Anda diaktifkan untuk alamat penetapan otomatis IPv6 . Untuk informasi selengkapnya tentang cara mengonfigurasi subnet, lihat Memodifikasi atribut IPv6 pengalamatan untuk subnet Anda di Panduan Pengguna Amazon VPC.
- Tugas atau layanan menggunakan versi platform Fargate `1.4.0` atau yang lebih baru untuk Linux.

Jika Anda mengonfigurasi VPC Anda dengan gateway internet atau gateway internet khusus keluar, tugas Amazon ECS di Fargate yang diberi alamat dapat mengakses internet. IPv6 Gateway NAT tidak diperlukan. Untuk informasi selengkapnya, lihat gateway Internet dan gateway internet khusus eGress di Panduan Pengguna Amazon VPC.

Opsi penyimpanan untuk tugas Amazon ECS

Amazon ECS memberi Anda opsi penyimpanan easy-to-use data yang fleksibel, hemat biaya, dan tergantung pada kebutuhan Anda. Amazon ECS mendukung opsi volume data berikut untuk kontainer:

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
Amazon Elastic Block Store (Amazon EBS)	Fargate, Amazon EC2	Linux	Dapat dipertahankan saat dilampirkan ke tugas mandiri. Fana ketika dilampirkan pada tugas yang dikelola oleh layanan.	Volume Amazon EBS menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Kasus penggunaan umum termasuk beban kerja transaksional seperti database, desktop virtual dan volume

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
				root, dan beban kerja intensif throughput seperti pemrosesan log dan beban kerja ETL. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EBS dengan Amazon ECS.

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
Amazon Elastic File System (Amazon EFS)	Fargate, Amazon EC2	Linux	Persisten	Volume Amazon EFS menyediakan penyimpanan file bersama yang sederhana, dapat diskalakan, dan persisten untuk digunakan dengan tugas Amazon ECS Anda yang tumbuh dan menyusut secara otomatis saat Anda menambahkan dan menghapus file. Volume Amazon EFS mendukung konkurensi dan berguna untuk aplikasi kontainer yang menskalakan secara horizontal dan membutuhkan fungsionalitas penyimpanan seperti latensi rendah, throughput tinggi, dan

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
				konsistensi. read-after-write Kasus penggunaan umum termasuk beban kerja seperti analisis data, pemrosesan media, manajemen konten, dan penayangan web. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
Amazon FSx untuk Server File Windows	Amazon EC2	Windows	Persisten	FSx untuk Windows File Server volume menyediakan server file Windows yang dikelola sepenuhnya yang dapat Anda gunakan untuk menyediakan tugas Windows Anda yang memerlukan penyimpanan file persisten, didistribusikan, dibagikan, dan statis. Kasus penggunaan umum termasuk aplikasi.NET yang mungkin memerlukan folder lokal sebagai penyimpanan persisten untuk menyimpan output aplikasi. Amazon FSx untuk Windows File Server

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
				menawarkan folder lokal dalam wadah yang memungkinkan beberapa kontainer untuk membaca-tulis pada sistem file yang sama yang didukung oleh SMB Share. Untuk informasi selengkapnya, lihat Gunakan FSx untuk volume Server File Windows dengan Amazon ECS.

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
Volume Docker	Amazon EC2	Windows, Linux	Persisten	Volume Docker adalah fitur runtime kontainer Docker yang memungkinkan kontainer menyimpan data dengan memasang direktori dari sistem file host. Driver volume Docker (juga disebut sebagai plugin) digunakan untuk menginteg rasikan volume kontainer dengan sistem penyimpan an eksternal. Volume Docker dapat dikelola oleh driver pihak ketiga atau oleh `local` driver bawaan. Kasus penggunaan umum untuk volume Docker termasuk menyediak

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
				an volume data persisten atau volume berbagi di lokasi yang berbeda pada container yang berbeda pada instance container yang sama. Untuk informasi selengkapnya, lihat Gunakan volume Docker dengan Amazon ECS.

Volume data	Jenis peluncuran yang didukung	Sistem operasi yang didukung	Kegigihan penyimpanan	Kasus penggunaan
Pemasangan terikat	Fargate, Amazon EC2	Windows, Linux	Fana	Bind mount terdiri dari file atau direktori pada host, seperti EC2 instance Amazon atau AWS Fargate, yang dipasang ke wadah. Kasus penggunaan umum untuk pemasanga n pengikat termasuk berbagi volume dari wadah sumber dengan wadah lain dalam tugas yang sama, atau memasang volume host atau volume kosong dalam satu atau lebih wadah. Untuk informasi selengkapnya, lihat Gunakan bind mount dengan Amazon ECS.

Gunakan volume Amazon EBS dengan Amazon ECS

Volume Amazon Elastic Block Store (Amazon EBS) menyediakan penyimpanan blok berkinerja tinggi yang sangat tersedia, hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja intensif data. Volume Amazon EBS dapat digunakan dengan tugas Amazon ECS untuk throughput tinggi dan aplikasi intensif transaksi.

Selama peluncuran tugas mandiri, Anda dapat memberikan konfigurasi yang akan digunakan untuk melampirkan satu volume EBS ke tugas. Selama pembuatan atau pembaruan layanan, Anda dapat memberikan konfigurasi yang akan digunakan untuk melampirkan satu volume EBS per tugas ke setiap tugas yang dikelola oleh layanan Amazon ECS.

Dengan menyediakan konfigurasi volume pada waktu peluncuran dan bukan dalam definisi tugas, Anda membuat definisi tugas yang tidak dibatasi untuk tipe volume data tertentu atau pengaturan volume EBS tertentu. Anda kemudian dapat menggunakan kembali definisi tugas Anda di lingkungan runtime yang berbeda. Misalnya, Anda dapat memberikan lebih banyak throughput selama penerapan untuk beban kerja produksi Anda daripada lingkungan pra-prod Anda.

Volume Amazon EBS yang dilampirkan ke tugas Amazon ECS dikelola oleh Amazon ECS atas nama Anda. Volume dapat dienkripsi dengan kunci AWS Key Management Service (AWS KMS) untuk melindungi data Anda. Anda dapat mengonfigurasi volume baru yang kosong untuk lampiran, atau Anda dapat menggunakan snapshot untuk memuat data dari volume yang ada. Saat Anda menggunakan snapshot untuk mengonfigurasi volume, Anda dapat menentukan`volumeInitializationRate`, dalam MIB/s, di mana data diambil dari snapshot untuk membuat volume yang sepenuhnya diinisialisasi dalam jumlah waktu yang dapat diprediksi. Untuk informasi selengkapnya tentang inisialisasi volume, lihat Menginisialisasi volume Amazon EBS di Panduan Pengguna Amazon EBS. Untuk informasi selengkapnya tentang mengonfigurasi volume Amazon EBS, lihat Tunda konfigurasi volume untuk waktu peluncuran dalam definisi tugas Amazon ECS dan. Tentukan konfigurasi volume Amazon EBS pada penerapan Amazon ECS

Untuk memantau kinerja volume, Anda juga dapat menggunakan CloudWatch metrik Amazon. Untuk informasi selengkapnya tentang metrik Amazon ECS untuk volume Amazon EBS, lihat dan metrik Metrik Amazon ECS CloudWatch Amazon ECS Container Insights.

Melampirkan volume Amazon EBS ke tugas didukung di semua komersial dan China yang Wilayah AWSmendukung Amazon ECS.

Untuk informasi selengkapnya tentang volume Amazon EBS, lihat volume Amazon EBS di Panduan Pengguna Amazon EBS.

Sistem operasi yang didukung dan jenis peluncuran

Tabel berikut menyediakan sistem operasi yang didukung dan konfigurasi tipe peluncuran.

Jenis peluncuran	Linux	Windows
Fargate	Volume Amazon EBS didukung pada versi platform `1.4.0` atau yang lebih baru (Linux). Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.	Tidak didukung
EC2	Tugas yang dihosting pada instans Linux Nitro berbasis dengan Amazon Machine Images AMIs () yang dioptimal kan Amazon ECS. Untuk informasi selengkapnya tentang jenis instans, lihat Jenis instans di Panduan EC2 Pengguna Amazon. Volume Amazon EBS didukung pada `20231219` AMI yang dioptimalkan ECS atau versi lebih baru. Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.	Tugas yang dihosting pada instans Linux Nitro berbasis dengan Amazon Machine Images AMIs () yang dioptimal kan Amazon ECS. Untuk informasi selengkapnya tentang jenis instans, lihat Jenis instans di Panduan EC2 Pengguna Amazon. Volume Amazon EBS didukung pada `20241017` AMI yang dioptimalkan ECS atau versi lebih baru. Untuk informasi selengkapnya, lihat Mengambil metadata Windows AMI yang dioptimal kan Amazon ECS.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan volume Amazon EBS:

- Anda tidak dapat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS jenis peluncuran Fargate di `use1-az3` Availability Zone.
- Jenis volume Amazon EBS magnetik (`standard`) tidak didukung untuk tugas yang dihosting di Fargate. Untuk informasi selengkapnya tentang jenis volume Amazon EBS, lihat Volume Amazon EBS di EC2 Panduan Pengguna Amazon.
- Peran IAM infrastruktur Amazon ECS diperlukan saat membuat layanan atau tugas mandiri yang mengonfigurasi volume saat penerapan. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` IAM AWS terkelola ke peran, atau Anda dapat menggunakan kebijakan terkelola sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.
- Anda dapat melampirkan paling banyak satu volume Amazon EBS ke setiap tugas Amazon ECS, dan itu harus berupa volume baru. Anda tidak dapat melampirkan volume Amazon EBS yang ada ke tugas. Namun, Anda dapat mengonfigurasi volume Amazon EBS baru saat penerapan menggunakan snapshot dari volume yang ada.
- Untuk menggunakan volume Amazon EBS dengan layanan Amazon ECS, pengontrol penerapan harus 'ECS'. Strategi rolling dan blue/green deployment didukung saat menggunakan pengontrol penerapan ini.
- Agar wadah dalam tugas Anda dapat menulis ke volume Amazon EBS yang dipasang, Anda harus menjalankan penampung sebagai pengguna root.
- Amazon ECS secara otomatis menambahkan tag yang dicadangkan `AmazonECSCreated` dan `AmazonECSManaged` ke volume terlampir. Jika Anda menghapus tag ini dari volume, Amazon ECS tidak akan dapat mengelola volume atas nama Anda. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS. Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat Menandai sumber daya Amazon ECS Anda.
- Penyediaan volume dari snapshot volume Amazon EBS yang berisi partisi tidak didukung.
- Volume yang dilampirkan ke tugas yang dikelola oleh layanan tidak dipertahankan dan selalu dihapus setelah pemutusan tugas.
- Anda tidak dapat mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS yang sedang berjalan. AWS Outposts

Tunda konfigurasi volume untuk waktu peluncuran dalam definisi tugas Amazon ECS

Untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Anda, Anda harus menentukan konfigurasi titik pemasangan dalam definisi tugas dan memberi nama volume. Anda juga harus mengatur `configuredAtLaunch` ke `true` karena volume Amazon EBS tidak dapat dikonfigurasi untuk lampiran dalam definisi tugas. Sebagai gantinya, volume Amazon EBS dikonfigurasi untuk lampiran selama penerapan.

Untuk mendaftarkan definisi tugas dengan menggunakan AWS Command Line Interface (AWS CLI), simpan template sebagai file JSON, lalu berikan file sebagai input untuk `register-task-definition` perintah.

Untuk membuat dan mendaftarkan definisi tugas menggunakan AWS Management Console, lihatMembuat definisi tugas Amazon ECS menggunakan konsol.

Definisi tugas berikut menunjukkan sintaks untuk `mountPoints` dan `volumes` objek dalam definisi tugas. Untuk informasi selengkapnya tentang parameter definisi tugas, lihatParameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate. Untuk menggunakan contoh ini, ganti *`user input placeholders`*dengan informasi Anda sendiri.

Linux

```
{
    "family": "mytaskdef",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/nginx/nginx:latest",
            "networkMode": "awsvpc",
           "portMappings": [
                {
                    "name": "nginx-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp",
                    "appProtocol": "http"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "/mount/ebs",
```

```
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
}
```

Windows

```
{
    "family": "mytaskdef",
     "memory": "4096",
     "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
    "containerDefinitions": [
        {
             "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
```

```
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "drive:\ebs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
}
```

`mountPoints`

Tipe: Array objek

Wajib: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` dalam create-container Docker API dan opsi `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive. Anda harus menentukan titik pemasangan untuk melampirkan volume Amazon EBS langsung ke tugas Amazon ECS.

`sourceVolume`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Untuk tugas yang berjalan pada EC2 instance yang menjalankan sistem operasi Windows, biarkan nilainya sebagai default. `false`

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`configuredAtLaunch`

Jenis: Boolean

Wajib: Ya, saat Anda ingin menggunakan lampirkan volume EBS langsung ke tugas.

Menentukan apakah volume dikonfigurasi saat peluncuran. Saat disetel ke`true`, Anda dapat mengonfigurasi volume saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Saat disetel ke`false`, Anda tidak akan dapat memberikan konfigurasi volume lain dalam definisi tugas. Parameter ini harus disediakan dan diatur `true` untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas.

Enkripsi data yang disimpan dalam volume Amazon EBS yang dilampirkan ke tugas Amazon ECS

Anda dapat menggunakan AWS Key Management Service (AWS KMS) untuk membuat dan mengelola kunci kriptografi yang melindungi data Anda. Volume Amazon EBS dienkripsi saat istirahat dengan menggunakan. AWS KMS keys Jenis data berikut dienkripsi:

- Data disimpan saat istirahat pada volume
- Disk I/O
- Snapshot dibuat dari volume
- Volume baru dibuat dari snapshot terenkripsi

Volume Amazon EBS yang dilampirkan ke tugas dapat dienkripsi dengan menggunakan default Kunci yang dikelola AWS dengan aliasa`lias/aws/ebs`, atau kunci terkelola pelanggan simetris yang ditentukan dalam konfigurasi volume. Default Kunci yang dikelola AWS unik untuk masing-masing Akun AWS per Wilayah AWS dan dibuat secara otomatis. Untuk membuat kunci terkelola pelanggan simetris, ikuti langkah-langkah dalam Membuat kunci KMS enkripsi simetris di Panduan PengembangAWS KMS .

Anda dapat mengonfigurasi enkripsi Amazon EBS secara default sehingga semua volume baru yang dibuat dan dilampirkan ke tugas tertentu Wilayah AWS dienkripsi dengan menggunakan kunci KMS yang Anda tentukan untuk akun Anda. Untuk informasi selengkapnya tentang enkripsi dan enkripsi Amazon EBS secara default, lihat enkripsi Amazon EBS di Panduan Pengguna Amazon EBS.

Anda juga dapat mengatur enkripsi tingkat klaster Amazon ECS untuk penyimpanan terkelola Amazon ECS saat membuat atau memperbarui klaster. Enkripsi tingkat cluster dapat digunakan untuk mengenkripsi semua volume Amazon EBS yang dilampirkan ke tugas yang berjalan di klaster tertentu dengan menggunakan kunci KMS yang ditentukan pada tingkat cluster. Untuk informasi selengkapnya tentang mengonfigurasi enkripsi di tingkat klaster, lihat ManagedStorageConfigurationdi referensi Amazon ECS API.

Anda dapat mengonfigurasi kombinasi tombol-tombol ini. Urutan prioritas kunci KMS adalah sebagai berikut:

1. Kunci KMS ditentukan dalam konfigurasi volume. Saat Anda menentukan kunci KMS dalam konfigurasi volume, kunci ini akan mengganti default Amazon EBS dan kunci KMS apa pun yang ditentukan pada tingkat cluster.
2. Kunci KMS ditentukan pada tingkat cluster. Saat Anda menentukan kunci KMS untuk enkripsi tingkat klaster penyimpanan terkelola Amazon ECS, kunci tersebut akan mengganti enkripsi default Amazon EBS tetapi tidak mengganti kunci KMS apa pun yang ditentukan dalam konfigurasi volume.
3. Enkripsi default Amazon EBS. Enkripsi default berlaku ketika Anda tidak menentukan kunci KMS tingkat cluster atau kunci dalam konfigurasi volume. Jika Anda mengaktifkan enkripsi Amazon EBS secara default, default adalah kunci KMS yang Anda tentukan untuk enkripsi secara default. Jika tidak, defaultnya adalah Kunci yang dikelola AWS dengan alias`alias/aws/ebs`.

> **Note**
>
> Jika Anda menyetel `encrypted` ke `false` dalam konfigurasi volume, tentukan tidak ada kunci KMS tingkat cluster, dan aktifkan enkripsi Amazon EBS secara default, volume akan tetap dienkripsi dengan kunci yang ditentukan untuk enkripsi Amazon EBS secara default.

Kebijakan kunci KMS yang dikelola pelanggan

Untuk mengenkripsi volume EBS yang dilampirkan ke tugas Anda dengan menggunakan kunci yang dikelola pelanggan, Anda harus mengonfigurasi kebijakan kunci KMS untuk memastikan bahwa peran IAM yang Anda gunakan untuk konfigurasi volume memiliki izin yang diperlukan untuk menggunakan kunci tersebut. Kebijakan utama harus menyertakan `kms:CreateGrant` dan `kms:GenerateDataKey*` izin. `kms:ReEncryptFrom`Izin `kms:ReEncryptTo` dan diperlukan untuk mengenkripsi volume yang dibuat menggunakan snapshot. Jika Anda ingin mengkonfigurasi dan mengenkripsi hanya volume baru yang kosong untuk lampiran, Anda dapat mengecualikan `kms:ReEncryptTo` dan `kms:ReEncryptFrom` izin.

Cuplikan JSON berikut menunjukkan pernyataan kebijakan utama yang dapat Anda lampirkan ke kebijakan kunci KMS Anda. Menggunakan pernyataan ini akan memberikan akses bagi Amazon ECS untuk menggunakan kunci untuk mengenkripsi volume EBS. Untuk menggunakan contoh pernyataan kebijakan, ganti *`user input placeholders`* dengan informasi Anda sendiri. Seperti biasa, hanya konfigurasikan izin yang Anda butuhkan.

```
{
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:DescribeKey",
      "Resource":"*"
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": [
      "kms:GenerateDataKey*",
      "kms:ReEncryptTo",
      "kms:ReEncryptFrom"
      ],
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:CreateGrant",
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        },
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
```

Untuk informasi selengkapnya tentang kebijakan dan izin utama, lihat Kebijakan kunci AWS KMS dan AWS KMS izin di Panduan AWS KMS Pengembang. Untuk mengatasi masalah lampiran volume EBS yang terkait dengan izin kunci, lihat. Memecahkan masalah lampiran volume Amazon EBS ke tugas Amazon ECS

Tentukan konfigurasi volume Amazon EBS pada penerapan Amazon ECS

Setelah mendaftarkan definisi tugas dengan `configuredAtLaunch` parameter yang disetel ke`true`, Anda dapat mengonfigurasi volume Amazon EBS saat penerapan saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan.

Untuk mengonfigurasi volume, Anda dapat menggunakan Amazon ECS APIs, atau Anda dapat meneruskan file JSON sebagai input untuk perintah berikut: AWS CLI

- `run-task`untuk menjalankan tugas ECS mandiri.
- `start-task`untuk menjalankan tugas ECS mandiri dalam instance kontainer tertentu. Perintah ini tidak berlaku untuk tugas jenis peluncuran Fargate.
- `create-service`untuk membuat layanan ECS baru.
- `update-service`untuk memperbarui layanan yang ada.

> **Note**
>
> Agar wadah dalam tugas Anda dapat menulis ke volume Amazon EBS yang dipasang, Anda harus menjalankan penampung sebagai pengguna root.

Anda juga dapat mengonfigurasi volume Amazon EBS dengan menggunakan file. AWS Management Console Lihat informasi selengkapnya di Menjalankan aplikasi sebagai tugas Amazon ECS, Membuat penyebaran pembaruan bergulir Amazon ECS, dan Memperbarui layanan Amazon ECS.

Cuplikan JSON berikut menunjukkan semua parameter volume Amazon EBS yang dapat dikonfigurasi saat penerapan. Untuk menggunakan parameter ini untuk konfigurasi volume, ganti *`user input placeholders`* dengan informasi Anda sendiri. Untuk informasi selengkapnya tentang parameter ini, lihat Konfigurasi volume.

```
"volumeConfigurations": [
        {
            "name": "ebs-volume",
```

```
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "volumeType": "gp3",
                "sizeInGiB": 10,
                "snapshotId": "snap-12345",
                "volumeInitializationRate":100,
                "iops": 3000,
                "throughput": 125,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "arn:aws:iam::1111222333:role/ecsInfrastructureRole",
                 "terminationPolicy": {
                    "deleteOnTermination": true//can't be configured for service-
managed tasks, always true
                },
                "filesystemType": "ext4"
            }
        }
    ]
```

> **Important**
>
> Pastikan bahwa yang `volumeName` Anda tentukan dalam konfigurasi sama dengan yang `volumeName` Anda tentukan dalam definisi tugas Anda.

Untuk informasi tentang memeriksa status lampiran volume, lihatMemecahkan masalah lampiran volume Amazon EBS ke tugas Amazon ECS . Untuk informasi tentang peran infrastruktur Amazon ECS AWS Identity and Access Management (IAM) yang diperlukan untuk lampiran volume EBS, lihat. Peran IAM infrastruktur Amazon ECS

Berikut ini adalah contoh cuplikan JSON yang menunjukkan konfigurasi volume Amazon EBS. Contoh-contoh ini dapat digunakan dengan menyimpan cuplikan dalam file JSON dan meneruskan file sebagai parameter (menggunakan `--cli-input-json file://filename` parameter) untuk perintah. AWS CLI Ganti *user input placeholders* dengan informasi Anda sendiri.

Konfigurasikan volume untuk tugas mandiri

Cuplikan berikut menunjukkan sintaks untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas mandiri. Cuplikan JSON berikut menunjukkan sintaks untuk mengkonfigurasi`volumeType`,,, `sizeInGiB` dan pengaturan. `encrypted` `kmsKeyId` Konfigurasi yang ditentukan dalam file JSON digunakan untuk membuat dan melampirkan volume EBS ke tugas mandiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "roleArn":"arn:aws:iam::1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

Konfigurasikan volume pada pembuatan layanan

Cuplikan berikut menunjukkan sintaks untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas yang dikelola oleh layanan. Volume bersumber dari snapshot yang ditentukan menggunakan `snapshotId` parameter pada tingkat 200 MiB/s. Konfigurasi yang ditentukan dalam file JSON digunakan untuk membuat dan melampirkan volume EBS ke setiap tugas yang dikelola oleh layanan.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
```

```
    "desiredCount": 2,
    "volumeConfigurations": [
         {
             "name": "myEbsVolume",
             "managedEBSVolume": {
               "roleArn":"arn:aws:iam::1111222333:role/ecsInfrastructureRole",
               "snapshotId": "snap-12345",
               "volumeInitializationRate": 200
             }
         }
    ]
}
```

Konfigurasikan volume pada pembaruan layanan

Cuplikan JSON berikut menunjukkan sintaks untuk memperbarui layanan yang sebelumnya tidak memiliki volume Amazon EBS yang dikonfigurasi untuk lampiran ke tugas. Anda harus memberikan ARN dari revisi definisi tugas dengan `configuredAtLaunch` set to. `true` Cuplikan JSON berikut menunjukkan sintaks untuk mengkonfigurasi`volumeType`,,, dan `sizeInGiBthroughput`, dan pengaturan. `iops` `filesystemType` Konfigurasi ini digunakan untuk membuat dan melampirkan volume EBS ke setiap tugas yang dikelola oleh layanan.

```
{
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "service": "mysvc",
    "desiredCount": 2,
    "volumeConfigurations": [
         {
             "name": "myEbsVolume",
             "managedEBSVolume": {
               "roleArn":"arn:aws:iam::1111222333:role/ecsInfrastructureRole",
                "volumeType": "gp3",
                 "sizeInGiB": 100,
                  "iops": 3000,
                 "throughput": 125,
                 "filesystemType": "ext4"
             }
         }
    ]
}
```

Konfigurasikan layanan agar tidak lagi menggunakan volume Amazon EBS

Cuplikan JSON berikut menunjukkan sintaks untuk memperbarui layanan agar tidak lagi menggunakan volume Amazon EBS. Anda harus memberikan ARN definisi tugas dengan `configuredAtLaunch` set to`false`, atau definisi tugas tanpa parameter. `configuredAtLaunch` Anda juga harus menyediakan `volumeConfigurations` objek kosong.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "service": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": []
}
```

Kebijakan penghentian untuk volume Amazon EBS

Saat tugas Amazon ECS dihentikan, Amazon ECS menggunakan `deleteOnTermination` nilainya untuk menentukan apakah volume Amazon EBS yang terkait dengan tugas yang dihentikan harus dihapus. Secara default, volume EBS yang dilampirkan ke tugas dihapus saat tugas dihentikan. Untuk tugas mandiri, Anda dapat mengubah setelan ini untuk mempertahankan volume saat penghentian tugas.

> **Note**
>
> Volume yang dilampirkan ke tugas yang dikelola oleh layanan tidak dipertahankan dan selalu dihapus pada saat pemutusan tugas.

Tandai volume Amazon EBS

Anda dapat menandai volume Amazon EBS dengan menggunakan `tagSpecifications` objek. Menggunakan objek, Anda dapat memberikan tag Anda sendiri dan mengatur propagasi tag dari definisi tugas atau layanan, tergantung pada apakah volume dilampirkan ke tugas mandiri atau tugas dalam layanan. Jumlah maksimum tag yang dapat dilampirkan ke volume adalah 50.

> **Important**
>
> Amazon ECS secara otomatis melampirkan tag `AmazonECSCreated` dan `AmazonECSManaged` cadangan ke volume Amazon EBS. Ini berarti Anda dapat mengontrol

lampiran maksimal 48 tag tambahan ke volume. Tag tambahan ini dapat berupa tag yang ditentukan pengguna, dikelola ECS, atau disebarkan.

Jika Anda ingin menambahkan tag Amazon ECS yang dikelola ke volume Anda, Anda harus menyetel `enableECSManagedTags` ke `true` dalam`UpdateService`,`CreateService`, `RunTask` atau `StartTask` panggilan. Jika Anda mengaktifkan tag yang dikelola Amazon ECS, Amazon ECS akan menandai volume secara otomatis dengan informasi kluster dan layanan (`aws:ecs:`*`clusterName`*dan). `aws:ecs:`*`serviceName`* Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat Menandai sumber daya Amazon ECS Anda.

Cuplikan JSON berikut menunjukkan sintaks untuk menandai setiap volume Amazon EBS yang dilampirkan ke setiap tugas dalam layanan dengan tag yang ditentukan pengguna. Untuk menggunakan contoh ini untuk membuat layanan, ganti *`user input placeholders`* dengan informasi Anda sendiri.

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "enableECSManagedTags": true,
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                 "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
```

```
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
    ]
}
```

> **Important**
>
> Anda harus menentukan jenis `volume` sumber daya untuk menandai volume Amazon EBS.

Kinerja volume Amazon EBS untuk tugas sesuai permintaan Fargate

Volume Amazon EBS dasar IOPS dan throughput yang tersedia untuk tugas sesuai permintaan Fargate bergantung pada total unit CPU yang Anda minta untuk tugas tersebut. Jika Anda meminta 0,25, 0,5, atau 1 unit CPU virtual (vCPU) untuk tugas Fargate Anda, kami sarankan Anda mengonfigurasi volume SSD Tujuan Umum gp2 (gp3atau) atau volume Hard Disk Drive (HDD) (atau). `st1` `sc1` Jika Anda meminta lebih dari 1 vCPU untuk tugas Fargate Anda, batas kinerja dasar berikut berlaku untuk volume Amazon EBS yang dilampirkan ke tugas. Anda mungkin untuk sementara mendapatkan kinerja EBS yang lebih tinggi daripada batas berikut. Namun, kami menyarankan Anda merencanakan beban kerja Anda berdasarkan batas-batas ini.

Unit CPU diminta (dalam vCPUs)	Dasar Amazon EBS IOPS (16 KiB I/O)	Throughput Amazon EBS Dasar (dalam, MiBps 128 KiB I/O)	Bandwidth dasar (dalam Mbps)
2	3.000	75	360
4	5.000	120	1.150
8	10.000	250	2.300
16	15.000	500	4,500

> **Note**
>
> Saat Anda mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Fargate, batas kinerja Amazon EBS untuk tugas Fargate dibagi antara penyimpanan sementara tugas dan volume terlampir.

Kinerja volume Amazon EBS untuk tugas EC2

Amazon EBS menyediakan jenis volume, yang berbeda dalam karakteristik kinerja dan harga, sehingga Anda dapat menyesuaikan kinerja penyimpanan dan biaya dengan kebutuhan aplikasi Anda. Untuk informasi tentang performa, termasuk IOPS per volume dan throughput per volume, lihat Jenis volume Amazon EBS di Panduan Pengguna Amazon Elastic Block Store.

Memecahkan masalah lampiran volume Amazon EBS ke tugas Amazon ECS

Anda mungkin perlu memecahkan masalah atau memverifikasi lampiran volume Amazon EBS ke tugas Amazon ECS.

Periksa status lampiran volume

Anda dapat menggunakan AWS Management Console untuk melihat status lampiran volume Amazon EBS ke tugas Amazon ECS. Jika tugas dimulai dan lampiran gagal, Anda juga akan melihat alasan status yang dapat Anda gunakan untuk memecahkan masalah. Volume yang dibuat akan dihapus dan tugas akan dihentikan. Untuk informasi selengkapnya tentang alasan status, lihatAlasan status untuk lampiran volume Amazon EBS ke tugas Amazon ECS.

Untuk melihat status lampiran volume dan alasan status menggunakan konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih klaster tempat tugas Anda berjalan. Halaman detail cluster muncul.
3. Pada halaman detail klaster, pilih tab Tugas.
4. Pilih tugas yang Anda inginkan untuk melihat status lampiran volume. Anda mungkin perlu menggunakan Filter status yang diinginkan dan memilih Berhenti jika tugas yang ingin Anda periksa telah berhenti.
5. Pada halaman detail tugas, pilih tab Volume. Anda akan dapat melihat status lampiran volume Amazon EBS di bawah status Lampiran. Jika volume gagal dilampirkan ke tugas, Anda dapat memilih status di bawah Status lampiran untuk menampilkan penyebab kegagalan.

Anda juga dapat melihat status lampiran volume tugas dan alasan status terkait dengan menggunakan DescribeTasksAPI.

Kegagalan layanan dan tugas

Anda mungkin mengalami kegagalan layanan atau tugas yang tidak spesifik untuk volume Amazon EBS yang dapat memengaruhi lampiran volume. Untuk informasi selengkapnya, silakan lihat

- Pesan acara layanan
- Menghentikan kode kesalahan tugas
- Alasan kegagalan API

Alasan status untuk lampiran volume Amazon EBS ke tugas Amazon ECS

Gunakan referensi berikut untuk memperbaiki masalah yang mungkin Anda temui dalam bentuk alasan status AWS Management Console saat Anda mengonfigurasi volume Amazon EBS untuk lampiran ke tugas Amazon ECS. Untuk informasi selengkapnya tentang menemukan alasan status ini di konsol, lihatPeriksa status lampiran volume.

ECS tidak dapat mengasumsikan Peran Infrastruktur ECS yang dikonfigurasi 'arn:aws:iam: ::role/ '. *111122223333* *ecsInfrastructureRole* Harap verifikasi bahwa peran yang dilewati memiliki hubungan kepercayaan yang tepat dengan Amazon ECS

Alasan status ini muncul dalam skenario berikut.

- Anda memberikan peran IAM tanpa kebijakan kepercayaan yang diperlukan. Amazon ECS tidak dapat mengakses peran IAM infrastruktur Amazon ECS yang Anda berikan jika peran tersebut tidak memiliki kebijakan kepercayaan yang diperlukan. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk informasi lebih lanjut tentang kebijakan kepercayaan yang diperlukan, lihatPeran IAM infrastruktur Amazon ECS.
- Pengguna IAM Anda tidak memiliki izin untuk meneruskan peran infrastruktur Amazon ECS ke Amazon ECS. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk menghindari masalah ini, Anda dapat melampirkan `PassRole` izin ke pengguna Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.
- Peran IAM Anda tidak memiliki izin yang diperlukan untuk lampiran volume Amazon EBS. Tugas itu bisa macet di `DEPROVISIONING` negara bagian. Untuk informasi selengkapnya tentang izin khusus yang diperlukan untuk melampirkan volume Amazon EBS ke tugas, lihat. Peran IAM infrastruktur Amazon ECS

> **Note**
>
> Anda juga dapat melihat pesan kesalahan ini karena penundaan dalam propagasi peran. Jika mencoba kembali menggunakan peran setelah menunggu beberapa menit tidak memperbaiki masalah, Anda mungkin telah salah mengonfigurasi kebijakan kepercayaan untuk peran tersebut.

ECS gagal mengatur volume EBS. Ditemui IdempotentParameterMismatch “; “Token klien yang Anda berikan dikaitkan dengan sumber daya yang sudah dihapus. Silakan gunakan token klien yang berbeda.”

Skenario AWS KMS utama berikut dapat menyebabkan `IdempotentParameterMismatch` pesan muncul:

- Anda menentukan kunci KMS ARN, ID, atau alias yang tidak valid. Dalam skenario ini, tugas mungkin tampak berhasil diluncurkan, tetapi tugas akhirnya gagal karena AWS mengautentikasi kunci KMS secara asinkron. Untuk informasi selengkapnya, lihat enkripsi Amazon EBS di Panduan EC2 Pengguna Amazon.
- Anda memberikan kunci terkelola pelanggan yang tidak memiliki izin yang memungkinkan peran IAM infrastruktur Amazon ECS menggunakan kunci untuk enkripsi. Untuk menghindari masalah izin kebijakan kunci, lihat contoh kebijakan AWS KMS kunci dalam enkripsi data untuk volume Amazon EBS.

Anda dapat mengatur Amazon EventBridge untuk mengirim peristiwa volume Amazon EBS dan peristiwa perubahan status tugas Amazon ECS ke target, seperti grup Amazon CloudWatch . Anda kemudian dapat menggunakan peristiwa ini untuk mengidentifikasi masalah terkait kunci yang dikelola pelanggan tertentu yang memengaruhi lampiran volume. Untuk informasi selengkapnya, silakan lihat

- Bagaimana saya bisa membuat grup CloudWatch log untuk digunakan sebagai target EventBridge aturan? di AWS re:post.
- Acara perubahan status tugas.
- EventBridge Acara Amazon untuk Amazon EBS di Panduan Pengguna Amazon EBS.

Waktu ECS habis saat mengonfigurasi lampiran volume EBS ke Tugas Anda.

Skenario format sistem file berikut menghasilkan pesan ini.

- Format sistem file yang Anda tentukan selama konfigurasi tidak kompatibel dengan sistem operasi tugas.

- Anda mengonfigurasi volume Amazon EBS yang akan dibuat dari snapshot, dan format sistem file snapshot tidak kompatibel dengan sistem operasi tugas. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat.

Anda dapat menggunakan log agen penampung Amazon ECS untuk memecahkan masalah pesan ini untuk tugas jenis peluncuran Amazon EC2 . Untuk informasi selengkapnya, lihat lokasi file log Amazon ECS dan pengumpul log Amazon ECS.

Gunakan volume Amazon EFS dengan Amazon ECS

Amazon Elastic File System (Amazon EFS) menyediakan penyimpanan file yang sederhana dan dapat diskalakan untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan dan kapan mereka membutuhkannya.

Anda dapat menggunakan sistem file Amazon EFS dengan Amazon ECS untuk mengekspor data sistem file di seluruh armada instans kontainer Anda. Dengan begitu, tugas Anda memiliki akses ke penyimpanan tetap yang sama, terlepas dari instans tempat mereka mendarat. Definisi tugas Anda harus mereferensikan pemasangan volume pada instance container untuk menggunakan sistem file.

Untuk tutorial, lihat Mengonfigurasi sistem file Amazon EFS untuk Amazon ECS menggunakan konsol.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan volume Amazon EFS:

- Untuk tugas yang menggunakan tipe EC2 peluncuran, dukungan sistem file Amazon EFS ditambahkan sebagai pratinjau publik dengan versi AMI Amazon ECS yang dioptimalkan dengan agen kontainer versi `20191212` 1.35.0. Namun, dukungan sistem file Amazon EFS memasuki ketersediaan umum dengan versi AMI Amazon ECS yang dioptimalkan `20200319` dengan agen kontainer versi 1.38.0, yang berisi jalur akses Amazon EFS dan fitur otorisasi IAM. Kami menyarankan Anda menggunakan versi AMI Amazon ECS yang dioptimalkan `20200319` atau yang lebih baru untuk menggunakan fitur-fitur ini. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

> **Note**
>
> Jika Anda membuat AMI sendiri, Anda harus menggunakan agen kontainer 1.38.0 atau yang lebih baru, `ecs-init` versi 1.38.0-1 atau yang lebih baru, dan jalankan perintah berikut di instans Amazon Anda EC2 untuk mengaktifkan plugin volume Amazon ECS. Perintah akan tergantung pada apakah Anda menggunakan Amazon Linux 2 atau Amazon Linux sebagai citra dasar Anda.
>
> Amazon Linux 2
>
> ```
> yum install amazon-efs-utils
> systemctl enable --now amazon-ecs-volume-plugin
> ```
>
> Amazon Linux
>
> ```
> yum install amazon-efs-utils
> sudo shutdown -r now
> ```

- Untuk tugas yang di-host di Fargate, sistem file Amazon EFS didukung pada platform versi 1.4.0 atau yang lebih baru (Linux). Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.
- Saat menggunakan volume Amazon EFS untuk tugas yang di-host di Fargate, Fargate membuat wadah pengawas yang bertanggung jawab untuk mengelola volume Amazon EFS. Wadah supervisor menggunakan sejumlah kecil memori tugas dan CPU. Kontainer pengawas akan terlihat saat dilakukan kueri titik akhir metadata tugas versi 4. Selain itu, ini terlihat di CloudWatch Container Insights sebagai nama `aws-fargate-supervisor` kontainer. Untuk informasi selengkapnya saat menggunakan jenis EC2 peluncuran Amazon, lihatTitik akhir metadata tugas Amazon ECS versi 4. Untuk informasi selengkapnya saat menggunakan jenis peluncuran Fargate, lihat. Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate
- Menggunakan volume Amazon EFS atau menentukan `EFSVolumeConfiguration` tidak didukung pada instans eksternal.
- Kami menyarankan Anda mengatur `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` parameter dalam file konfigurasi agen ke nilai yang kurang dari default (sekitar 1 jam). Perubahan ini membantu mencegah kedaluwarsa kredensi pemasangan EFS dan memungkinkan pembersihan dudukan yang tidak digunakan. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Gunakan titik akses Amazon EFS

Titik akses Amazon EFS adalah titik masuk khusus aplikasi ke dalam sistem file EFS untuk mengelola akses aplikasi ke kumpulan data bersama. Untuk informasi lebih lanjut tentang titik akses Amazon EFS dan cara mengontrol akses ke sana, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

Titik akses dapat menerapkan identitas pengguna, termasuk grup POSIX pengguna, pada semua permintaan sistem file yang dibuat melalui titik akses. Titik akses juga dapat menerapkan direktori root yang berbeda untuk sistem file. Ini agar klien hanya dapat mengakses data di direktori yang ditentukan atau subdirektorinya.

> **Note**
>
> Saat membuat titik akses EFS, tentukan jalur pada sistem file untuk berfungsi sebagai direktori root. Saat mereferensikan sistem file EFS dengan ID titik akses dalam definisi tugas Amazon ECS Anda, direktori root harus dihilangkan atau disetel ke`/`, yang memberlakukan jalur yang disetel pada titik akses EFS.

Anda dapat menggunakan peran IAM tugas Amazon ECS untuk menegakkan bahwa aplikasi tertentu menggunakan titik akses tertentu. Dengan menggabungkan kebijakan IAM dengan titik akses, Anda dapat memberikan akses aman ke kumpulan data tertentu untuk aplikasi Anda. Untuk informasi selengkapnya tentang cara menggunakan peran IAM tugas, lihatPeran IAM tugas Amazon ECS.

Praktik terbaik untuk menggunakan volume Amazon EFS dengan Amazon ECS

Catat rekomendasi praktik terbaik berikut saat Anda menggunakan Amazon EFS dengan Amazon ECS.

Kontrol keamanan dan akses untuk volume Amazon EFS

Amazon EFS menawarkan fitur kontrol akses yang dapat Anda gunakan untuk memastikan bahwa data yang disimpan dalam sistem file Amazon EFS aman dan hanya dapat diakses dari aplikasi yang membutuhkannya. Anda dapat mengamankan data dengan mengaktifkan enkripsi saat istirahat dan dalam perjalanan. Untuk informasi selengkapnya, lihat Enkripsi data di Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Selain enkripsi data, Anda juga dapat menggunakan Amazon EFS untuk membatasi akses ke sistem file. Ada tiga cara untuk menerapkan kontrol akses di EFS.

- Grup keamanan —Dengan target pemasangan Amazon EFS, Anda dapat mengonfigurasi grup keamanan yang digunakan untuk mengizinkan dan menolak lalu lintas jaringan. Anda dapat mengonfigurasi grup keamanan yang dilampirkan ke Amazon EFS untuk mengizinkan lalu lintas NFS (port 2049) dari grup keamanan yang dilampirkan ke instans Amazon ECS Anda atau, saat menggunakan mode `awsvpc` jaringan, tugas Amazon ECS.
- IAM —Anda dapat membatasi akses ke sistem file Amazon EFS menggunakan IAM. Saat dikonfigurasi, tugas Amazon ECS memerlukan peran IAM untuk akses sistem file untuk memasang sistem file EFS. Untuk informasi selengkapnya, lihat Menggunakan IAM untuk mengontrol akses data sistem file di Panduan Pengguna Amazon Elastic File System.

  Kebijakan IAM juga dapat menerapkan kondisi yang telah ditentukan sebelumnya seperti mewajibkan klien untuk menggunakan TLS saat menyambung ke sistem file Amazon EFS. Untuk informasi selengkapnya, lihat kunci kondisi Amazon EFS untuk klien di Panduan Pengguna Amazon Elastic File System.
- Jalur akses Amazon EFS —Jalur akses Amazon EFS adalah titik masuk khusus aplikasi ke dalam sistem file Amazon EFS. Anda dapat menggunakan titik akses untuk menegakkan identitas pengguna, termasuk grup POSIX pengguna, untuk semua permintaan sistem file yang dibuat melalui titik akses. Titik akses juga dapat menerapkan direktori root yang berbeda untuk sistem file. Ini agar klien hanya dapat mengakses data di direktori yang ditentukan atau sub-direktorinya.

Kebijakan IAM

Anda dapat menggunakan kebijakan IAM untuk mengontrol akses ke sistem file Amazon EFS.

Anda dapat menentukan tindakan berikut untuk klien yang mengakses sistem file menggunakan kebijakan sistem file.

Tindakan	Deskripsi
`elasticfilesystem:ClientMount`	Menyediakan akses read-only ke sistem file.
`elasticfilesystem:ClientWrite`	Memberikan izin menulis pada sistem file.
`elasticfilesystem:ClientRootAccess`	Menyediakan penggunaan pengguna root saat mengakses sistem file.

Anda perlu menentukan setiap tindakan dalam kebijakan. Kebijakan dapat didefinisikan dengan cara-cara berikut:

- Berbasis klien - Lampirkan kebijakan ke peran tugas

  Tetapkan opsi otorisasi IAM saat Anda membuat definisi tugas.
- Berbasis sumber daya - Lampirkan kebijakan ke sistem file Amazon EFS

  Jika kebijakan berbasis sumber daya tidak ada, secara default pada pembuatan sistem file akses diberikan kepada semua prinsipal (*).

Saat Anda menyetel opsi otorisasi IAM, kami menggabungkan kebijakan yang terkait dengan peran tugas dan berbasis sumber daya Amazon EFS. Opsi otorisasi IAM meneruskan identitas tugas (peran tugas) dengan kebijakan ke Amazon EFS. Hal ini memungkinkan kebijakan berbasis sumber daya Amazon EFS memiliki konteks untuk pengguna IAM atau peran yang ditentukan dalam kebijakan. Jika Anda tidak menetapkan opsi, kebijakan tingkat sumber daya Amazon EFS mengidentifikasi pengguna IAM sebagai “anonim”.

Pertimbangkan untuk menerapkan ketiga kontrol akses pada sistem file Amazon EFS untuk keamanan maksimum. Misalnya, Anda dapat mengonfigurasi grup keamanan yang dilampirkan ke titik pemasangan Amazon EFS untuk hanya mengizinkan lalu lintas NFS masuk dari grup keamanan yang terkait dengan instance container atau tugas Amazon ECS Anda. Selain itu, Anda dapat mengonfigurasi Amazon EFS agar memerlukan peran IAM untuk mengakses sistem file, meskipun koneksi berasal dari grup keamanan yang diizinkan. Terakhir, Anda dapat menggunakan jalur akses Amazon EFS untuk menerapkan izin pengguna POSIX dan menentukan direktori root untuk aplikasi.

Cuplikan definisi tugas berikut menunjukkan cara memasang sistem file Amazon EFS menggunakan titik akses.

```
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "authorizationConfig": {
          "accessPointId": "fsap-1234",
          "iam": "ENABLED"
        },
        "transitEncryption": "ENABLED",
        "rootDirectory": ""
      },
```

```
        "name": "my-filesystem"
    }
]
```

Performa volume Amazon EFS

Amazon EFS menawarkan dua mode kinerja: Tujuan Umum dan sistem I/O. General Purpose is suitable for latency-sensitive applications such as content management systems and CI/CD tools. In contrast, Max I/O file Max cocok untuk beban kerja seperti analisis data, pemrosesan media, dan pembelajaran mesin. Beban kerja ini perlu melakukan operasi paralel dari ratusan atau bahkan ribuan kontainer dan membutuhkan throughput agregat dan IOPS setinggi mungkin. Untuk informasi selengkapnya, lihat mode kinerja Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Beberapa beban kerja sensitif latensi memerlukan I/O tingkat yang lebih tinggi yang disediakan oleh mode I/O kinerja Maks dan latensi yang lebih rendah yang disediakan oleh mode kinerja Tujuan Umum. Untuk jenis beban kerja ini, kami sarankan untuk membuat beberapa sistem file mode kinerja Tujuan Umum. Dengan begitu, Anda dapat menyebarkan beban kerja aplikasi Anda di semua sistem file ini, selama beban kerja dan aplikasi dapat mendukungnya.

Throughput volume Amazon EFS

Semua sistem file Amazon EFS memiliki throughput terukur terkait yang ditentukan oleh jumlah throughput yang disediakan untuk sistem file yang menggunakan Provisioned Throughput atau jumlah data yang disimpan dalam kelas penyimpanan EFS Standard atau One Zone untuk sistem file yang menggunakan Bursting Throughput. Untuk informasi selengkapnya, lihat Memahami throughput terukur di Panduan Pengguna Amazon Elastic File System.

Mode throughput default untuk sistem file Amazon EFS adalah mode bursting. Dengan mode bursting, throughput yang tersedia untuk sistem file masuk atau keluar saat sistem file tumbuh. Karena beban kerja berbasis file biasanya melonjak, membutuhkan tingkat throughput yang tinggi untuk periode waktu dan tingkat throughput yang lebih rendah sepanjang waktu, Amazon EFS dirancang untuk meledak untuk memungkinkan tingkat throughput yang tinggi untuk periode waktu tertentu. Selain itu, karena banyak beban kerja yang berat baca, operasi baca diukur pada rasio 1:3 terhadap operasi NFS lainnya (seperti menulis).

Semua sistem file Amazon EFS memberikan kinerja dasar yang konsisten sebesar 50 MB/s untuk setiap TB penyimpanan Amazon EFS Standard atau Amazon EFS One Zone. Semua sistem file (terlepas dari ukurannya) dapat meledak hingga 100 MB/s. File systems with more than 1TB of EFS Standard or EFS One Zone storage can burst to 100 MB/s for each TB. Because read operations

are metered at a 1:3 ratio, you can drive up to 300 MiBs/s untuk setiap TiB throughput baca. Saat Anda menambahkan data ke sistem file, throughput maksimum yang tersedia untuk sistem file akan diskalakan secara linier dan otomatis dengan penyimpanan Anda di kelas penyimpanan Amazon EFS Standard. Jika Anda membutuhkan lebih banyak throughput daripada yang dapat Anda capai dengan jumlah data yang disimpan, Anda dapat mengonfigurasi Throughput yang Disediakan ke jumlah tertentu yang dibutuhkan beban kerja Anda.

Throughput sistem file dibagikan di semua EC2 instans Amazon yang terhubung ke sistem file. Misalnya, sistem file 1TB yang dapat meledak hingga 100 MB/s throughput dapat mendorong 100 MB/s dari satu EC2 instans Amazon dapat masing-masing drive 10 MB/s. Untuk informasi selengkapnya, lihat Performa Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Mengoptimalkan biaya untuk volume Amazon EFS

Amazon EFS menyederhanakan penyimpanan penskalaan untuk Anda. Sistem file Amazon EFS tumbuh secara otomatis saat Anda menambahkan lebih banyak data. Terutama dengan mode Amazon EFS Bursting Throughput, throughput di Amazon EFS meningkat seiring dengan bertambahnya ukuran sistem file Anda di kelas penyimpanan standar. Untuk meningkatkan throughput tanpa membayar biaya tambahan untuk throughput yang disediakan pada sistem file EFS, Anda dapat berbagi sistem file Amazon EFS dengan beberapa aplikasi. Menggunakan jalur akses Amazon EFS, Anda dapat menerapkan isolasi penyimpanan dalam sistem file Amazon EFS bersama. Dengan demikian, meskipun aplikasi masih berbagi sistem file yang sama, mereka tidak dapat mengakses data kecuali Anda mengotorisasi itu.

Seiring pertumbuhan data Anda, Amazon EFS membantu Anda memindahkan file yang jarang diakses secara otomatis ke kelas penyimpanan yang lebih rendah. Kelas penyimpanan Amazon EFS Standard-Infrequent Access (IA) mengurangi biaya penyimpanan untuk file yang tidak diakses setiap hari. Ini dilakukan tanpa mengorbankan ketersediaan tinggi, daya tahan tinggi, elastisitas, dan akses sistem file POSIX yang disediakan Amazon EFS. Untuk informasi selengkapnya, lihat kelas penyimpanan EFS di Panduan Pengguna Amazon Elastic File System.

Pertimbangkan untuk menggunakan kebijakan siklus hidup Amazon EFS untuk menghemat uang secara otomatis dengan memindahkan file yang jarang diakses ke penyimpanan Amazon EFS IA. Untuk informasi selengkapnya, lihat Manajemen siklus hidup Amazon EFS di Panduan Pengguna Amazon Elastic File System.

Saat membuat sistem file Amazon EFS, Anda dapat memilih apakah Amazon EFS mereplikasi data Anda di beberapa Availability Zone (Standar) atau menyimpan data Anda secara berlebihan dalam satu Availability Zone. Kelas penyimpanan Amazon EFS One Zone dapat mengurangi biaya

penyimpanan dengan margin yang signifikan dibandingkan dengan kelas penyimpanan Amazon EFS Standard. Pertimbangkan untuk menggunakan kelas penyimpanan Amazon EFS One Zone untuk beban kerja yang tidak memerlukan ketahanan Multi-AZ. Anda dapat mengurangi biaya penyimpanan Amazon EFS One Zone dengan memindahkan file yang jarang diakses ke Amazon EFS One Zone-Infrequent Access. Untuk informasi selengkapnya, lihat Amazon EFS Infrequent Access.

Perlindungan data volume Amazon EFS

Amazon EFS menyimpan data Anda secara berlebihan di beberapa Availability Zone untuk sistem file yang menggunakan kelas penyimpanan Standar. Jika Anda memilih kelas penyimpanan Amazon EFS One Zone, data Anda disimpan secara berlebihan dalam satu Availability Zone. Selain itu, Amazon EFS dirancang untuk memberikan daya tahan 99,999999999% (11 9) selama tahun tertentu.

Seperti halnya lingkungan apa pun, ini adalah praktik terbaik untuk memiliki cadangan dan membangun perlindungan terhadap penghapusan yang tidak disengaja. Untuk data Amazon EFS, praktik terbaik tersebut mencakup penggunaan cadangan yang berfungsi dan diuji secara teratur AWS Backup. Sistem file yang menggunakan kelas penyimpanan Amazon EFS One Zone dikonfigurasi untuk secara otomatis mencadangkan file secara default pada pembuatan sistem file kecuali Anda memilih untuk menonaktifkan fungsi ini. Untuk informasi selengkapnya, lihat Mencadangkan sistem file EFS di Panduan Pengguna Amazon Elastic File System.

Menentukan sistem file Amazon EFS dalam definisi tugas Amazon ECS

Untuk menggunakan volume sistem file Amazon EFS untuk container Anda, Anda harus menentukan konfigurasi volume dan titik mount dalam definisi tugas Anda. Definisi tugas berikut JSON cuplikan menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
            "name": "container-using-efs",
            "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/efs"
            ],
            "mountPoints": [
                {
```

```
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEfsVolume",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1234",
                "rootDirectory": "/path/to/my/data",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": integer,
                "authorizationConfig": {
                    "accessPointId": "fsap-1234",
                    "iam": "ENABLED"
                }
            }
        }
    ]
}
```

`efsVolumeConfiguration`

Tipe: Objek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Amazon EFS.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Amazon EFS yang akan digunakan.

`rootDirectory`

Tipe: String

Wajib: Tidak

Direktori dalam sistem file Amazon EFS untuk pemasangan sebagai direktori asal di dalam host. Jika parameter ini diabaikan, asal volume Amazon EFS akan digunakan. Menentukan `/` memiliki efek yang sama seperti mengabaikan parameter ini.

> **Important**
>
> Jika titik akses EFS ditentukan dalam `authorizationConfig`, parameter direktori root harus dihilangkan atau diatur ke `/`, yang memberlakukan jalur yang diatur pada titik akses EFS.

`transitEncryption`

Jenis: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan mengaktifkan enkripsi untuk data Amazon EFS dalam perjalanan antara host Amazon ECS dan server Amazon EFS. Jika otorisasi Amazon EFS IAM digunakan, enkripsi transit harus diaktifkan. Jika parameter ini diabaikan, nilai default `DISABLED` akan digunakan. Untuk informasi lebih lanjut, lihat Pengenkripsian Data Saat Transit di Panduan Pengguna Amazon Elastic File System.

`transitEncryptionPort`

Tipe: Integer

Wajib: Tidak

Port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, strategi pemilihan port yang digunakan oleh pembantu pemasangan Amazon EFS akan digunakan. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS dalam Panduan Pengguna Amazon Elastic File System.

`authorizationConfig`

Tipe: Objek

Wajib: Tidak

Detail konfigurasi otorisasi untuk sistem file Amazon EFS.

`accessPointId`

Tipe: String

Wajib: Tidak

ID titik akses yang akan digunakan. Jika titik akses ditentukan, nilai direktori root di dalam `efsVolumeConfiguration` harus dihilangkan atau disetel ke`/`, yang memberlakukan jalur yang ditetapkan pada titik akses EFS. Jika titik akses digunakan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Untuk informasi lebih lanjut, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

`iam`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat memasang sistem file Amazon EFS. Jika diaktifkan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Jika Anda menghilangkan parameter ini, nilai default `DISABLED` akan digunakan. Untuk informasi selengkapnya, lihat IAM Role pada Tugas.

Mengonfigurasi sistem file Amazon EFS untuk Amazon ECS menggunakan konsol

Pelajari cara menggunakan sistem file Amazon Elastic File System (Amazon EFS) dengan Amazon ECS.

Langkah 1: Buat cluster Amazon ECS

Gunakan langkah-langkah berikut untuk membuat cluster Amazon ECS.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan `EFS-tutorial` untuk nama cluster.
6. (Opsional) Untuk mengubah VPC dan subnet tempat tugas dan layanan Anda diluncurkan, di bawah Jaringan, lakukan salah satu operasi berikut:
   - Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
   - Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, lalu di bawah Subnet, pilih setiap subnet.
7. Untuk menambahkan EC2 instans Amazon ke klaster Anda, perluas Infrastruktur, lalu pilih EC2instans Amazon. Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:
   - Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:
     - Untuk Sistem Operasi/Arsitektur, pilih Amazon Linux 2.
     - EC2 Misalnya jenis, pilih`t2.micro`.

       Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
     - Untuk Kapasitas, masukkan`1`.
8. Pilih Buat.

Langkah 2: Buat grup keamanan untuk EC2 instans Amazon dan sistem file Amazon EFS

Pada langkah ini, Anda membuat grup keamanan untuk EC2 instans Amazon Anda yang memungkinkan lalu lintas jaringan masuk pada port 80 dan sistem file Amazon EFS Anda yang memungkinkan akses masuk dari instans penampung Anda.

Buat grup keamanan untuk EC2 instans Amazon Anda dengan opsi berikut:

- Nama grup keamanan - nama unik untuk grup keamanan Anda.

- VPC - VPC yang Anda identifikasi sebelumnya untuk cluster Anda.
- Aturan ke dalam
  - Jenis - HTTP
  - Sumber - 0.0.0.0/0.

Buat grup keamanan untuk sistem file Amazon EFS Anda dengan opsi berikut:

- Nama grup keamanan - nama unik untuk grup keamanan Anda. Misalnya, `EFS-access-for-sg-dc025fa2`.
- VPC - VPC yang Anda identifikasi sebelumnya untuk cluster Anda.
- Aturan ke dalam
  - Jenis - NFS
  - Sumber - Kustom dengan ID grup keamanan yang Anda buat untuk instance Anda.

Untuk informasi tentang cara membuat grup keamanan, lihat Membuat grup keamanan untuk EC2 instans Amazon Anda di Panduan EC2 Pengguna Amazon.

Langkah 3: Buat sistem file Amazon EFS

Pada langkah ini, Anda membuat sistem file Amazon EFS.

Untuk membuat sistem file Amazon EFS untuk tugas Amazon ECS.

1. Buka konsol Amazon Elastic File System di https://console.aws.amazon.com/efs/.
2. Pilih Buat sistem file.
3. Masukkan nama untuk sistem file Anda dan kemudian pilih VPC tempat instance container Anda di-host. Secara default, setiap subnet di VPC tertentu menerima target pemasangan yang menggunakan grup keamanan default untuk VPC tersebut. Kemudian, pilih Sesuaikan.

> **Note**
>
> Tutorial ini mengasumsikan bahwa sistem file Amazon EFS, cluster Amazon ECS, instance container, dan tugas Anda berada dalam VPC yang sama. Untuk informasi selengkapnya tentang memasang sistem file dari VPC yang berbeda, lihat Panduan: Memasang sistem file dari VPC yang berbeda di Panduan Pengguna Amazon EFS.

4. Pada halaman Pengaturan sistem file, konfigurasikan pengaturan opsional dan kemudian di bawah Pengaturan kinerja, pilih mode throughput meledak untuk sistem file Anda. Setelah Anda mengonfigurasi pengaturan, pilih Berikutnya.

   a. (Opsional) Tambahkan untuk menandai untuk sistem file Anda. Sebagai contoh, Anda dapat menentukan nama unik untuk sistem file dengan memasukkan nama tersebut di kolom Nilai di samping kunci Nama.

   b. (Opsional) Aktifkan manajemen siklus hidup untuk menghemat uang pada penyimpanan yang jarang diakses. Untuk informasi selengkapnya, lihat EFS Lifecycle Management di Panduan Pengguna Amazon Elastic File System.

   c. (Opsional) Aktifkan enkripsi. Pilih kotak centang untuk mengaktifkan enkripsi sistem file Amazon EFS Anda saat istirahat.

5. Pada halaman Akses jaringan, di bawah Target Mount, ganti konfigurasi grup keamanan yang ada untuk setiap zona ketersediaan dengan grup keamanan yang Anda buat untuk sistem file, Langkah 2: Buat grup keamanan untuk EC2 instans Amazon dan sistem file Amazon EFS lalu pilih Berikutnya.
6. Anda tidak perlu mengkonfigurasi kebijakan sistem File untuk tutorial ini, sehingga Anda dapat melewati bagian dengan memilih Berikutnya.
7. Tinjau opsi sistem file Anda dan pilih Buat untuk menyelesaikan proses.
8. Dari layar Sistem file, rekam ID sistem File. Pada langkah berikutnya, Anda akan mereferensikan nilai ini dalam definisi tugas Amazon ECS Anda.

Langkah 4: Tambahkan konten ke sistem file Amazon EFS

Pada langkah ini, Anda memasang sistem file Amazon EFS ke EC2 instans Amazon dan menambahkan konten ke dalamnya. Proses ini bertujuan untuk menguji tutorial ini, untuk menggambarkan sifat persisten data. Saat menggunakan fitur ini, Anda biasanya memiliki aplikasi atau metode lain untuk menulis data ke sistem file Amazon EFS Anda.

Untuk membuat EC2 instans Amazon dan memasang sistem file Amazon EFS

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan Instans.
3. Di bawah Gambar Aplikasi dan OS (Gambar Mesin Amazon), pilih Amazon Linux 2 AMI (HVM).
4. Di bawah tipe Instance, pertahankan tipe instance default,`t2.micro`.
5. Di bawah Key pair (login), pilih key pair untuk akses SSH ke instance.

6. Di bawah Pengaturan jaringan, pilih VPC yang Anda tentukan untuk sistem file Amazon EFS dan cluster Amazon ECS. Pilih subnet dan grup keamanan instance yang dibuat diLangkah 2: Buat grup keamanan untuk EC2 instans Amazon dan sistem file Amazon EFS. Konfigurasikan grup keamanan instans. Pastikan bahwa Auto-assign IP publik diaktifkan.
7. Di bawah Konfigurasi penyimpanan, pilih tombol Edit untuk sistem file dan kemudian pilih EFS. Pilih sistem file yang Anda buatLangkah 3: Buat sistem file Amazon EFS. Anda dapat secara opsional mengubah titik pemasangan atau meninggalkan nilai default.

   > **Important**
   >
   > Anda harus memilih subnet sebelum Anda dapat menambahkan sistem file ke instance.

8. Hapus secara otomatis membuat dan melampirkan grup keamanan. Biarkan kotak centang lainnya dipilih. Pilih Tambahkan sistem file bersama.
9. Di bawah Detail Lanjutan, pastikan bahwa skrip data pengguna diisi secara otomatis dengan langkah-langkah pemasangan sistem file Amazon EFS.
10. Di bawah Ringkasan, pastikan Jumlah instans adalah 1. Pilih Luncurkan instans.
11. Pada halaman Luncurkan instans, pilih Lihat semua instance untuk melihat status instance Anda. Awalnya, status status Instance adalah`PENDING`. Setelah status berubah `RUNNING` dan instance melewati semua pemeriksaan status, instance siap digunakan.

Sekarang, Anda terhubung ke EC2 instans Amazon dan menambahkan konten ke sistem file Amazon EFS.

Untuk terhubung ke EC2 instans Amazon dan menambahkan konten ke sistem file Amazon EFS

1. SSH ke EC2 instans Amazon yang Anda buat. Untuk informasi selengkapnya, lihat Connect ke instans Linux menggunakan SSH di Panduan EC2 Pengguna Amazon.
2. Dari jendela terminal, jalankan df -T perintah untuk memverifikasi bahwa sistem file Amazon EFS sudah terpasang. Pada output berikut, kami telah menyoroti pemasangan sistem file Amazon EFS.

```
$ df -T
Filesystem     Type            1K-blocks    Used        Available Use% Mounted on
devtmpfs       devtmpfs           485468       0           485468   0% /dev
tmpfs          tmpfs              503480       0           503480   0% /dev/shm
tmpfs          tmpfs              503480     424           503056   1% /run
```

```
tmpfs           tmpfs               503480       0           503480   0% /sys/fs/
cgroup
/dev/xvda1      xfs                8376300 1310952          7065348  16% /
127.0.0.1:/     nfs4     9007199254739968       0 9007199254739968   0% /mnt/efs/fs1
tmpfs           tmpfs               100700       0           100700   0% /run/
user/1000
```

3. Arahkan ke direktori tempat sistem file Amazon EFS dipasang. Dalam contoh di atas, yaitu `/mnt/efs/fs1`.
4. Buat file bernama `index.html` dengan konten berikut:

```
<html>
    <body>
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container
 storage.</p>
    </body>
</html>
```

Langkah 5: Buat ketentuan tugas

Ketentuan tugas berikut membuat volume data dengan nama `efs-html`. Pemasangan kontainer `nginx` volume data host pada root NGINX, `/usr/share/nginx/html`.

Untuk membuat definisi tugas baru menggunakan konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, salin dan tempel teks JSON berikut, ganti `fileSystemId` dengan ID sistem file Amazon EFS Anda.

```
{
    "containerDefinitions": [
        {
            "memory": 128,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
```

```
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "mountPoints": [
                {
                    "containerPath": "/usr/share/nginx/html",
                    "sourceVolume": "efs-html"
                }
            ],
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest"
        }
    ],
    "volumes": [
        {
            "name": "efs-html",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1324abcd",
                "transitEncryption": "ENABLED"
            }
        }
    ],
    "family": "efs-tutorial",
    "executionRoleArn":"arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

> **Note**
>
> Peran IAM eksekusi tugas Amazon ECS tidak memerlukan izin khusus terkait Amazon EFS untuk memasang sistem file Amazon EFS. Secara default, jika tidak ada kebijakan berbasis sumber daya Amazon EFS, akses diberikan ke semua prinsipal (*) pada pembuatan sistem file.
>
> Peran tugas Amazon ECS hanya diperlukan jika “otorisasi EFS IAM” diaktifkan dalam definisi tugas Amazon ECS. Saat diaktifkan, identitas peran tugas harus diizinkan mengakses sistem file Amazon EFS dalam kebijakan berbasis sumber daya Amazon EFS, dan akses anonim harus dinonaktifkan.

5. Pilih Buat.

Langkah 6: Jalankan tugas dan lihat hasilnya

Sekarang setelah sistem file Amazon EFS Anda dibuat dan ada konten web untuk penampung NGINX untuk disajikan, Anda dapat menjalankan tugas menggunakan definisi tugas yang Anda buat. Server web NGINX melayani halaman HTML sederhana Anda. Jika Anda memperbarui konten di sistem file Amazon EFS Anda, perubahan tersebut disebarkan ke wadah apa pun yang juga telah memasang sistem file tersebut.

Tugas berjalan di subnet yang Anda tentukan untuk cluster.

Untuk menjalankan tugas dan melihat hasilnya menggunakan konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman klaster, pilih klaster tempat tugas mandiri dijalankan.

   Tentukan sumber daya dari tempat Anda meluncurkan layanan.

Untuk memulai layanan dari	Langkah-langkah
Klaster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Tugas, pilih Jalankan tugas baru.
Jenis peluncuran	a. Pada halaman Tugas, pilih definisi tugas. b. Jika ada lebih dari satu revisi, pilih revisi. c. Pilih Buat, Jalankan tugas.

3. (Opsional) Pilih bagaimana tugas terjadwal Anda didistribusikan di seluruh infrastruktur klaster Anda. Perluas konfigurasi Compute, lalu lakukan hal berikut:

Metode distribusi	Langkah-langkah
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type.

Metode distribusi	Langkah-langkah
	b. Untuk jenis Peluncuran, pilih EC2.

4. Untuk Jenis aplikasi, pilih Tugas.
5. Untuk definisi Tugas, pilih definisi `efs-tutorial` tugas yang Anda buat sebelumnya.
6. Untuk tugas yang diinginkan, masukkan1.
7. Pilih Buat.
8. Pada halaman Cluster, pilih Infrastruktur.
9. Di bawah Instans Kontainer, pilih instance kontainer untuk dihubungkan.
10. Pada halaman Container Instance, di bawah Networking, rekam IP Publik untuk instans Anda.
11. Buka browser dan masukkan alamat IP publik. Anda akan melihat pesan berikut:

```
It works!
You are using an Amazon EFS file system for persistent container storage.
```

> **Note**
>
> Jika Anda tidak melihat pesan, pastikan bahwa grup keamanan untuk instance container Anda mengizinkan lalu lintas jaringan masuk pada port 80 dan grup keamanan untuk sistem file Anda memungkinkan akses masuk dari instance container.

Gunakan FSx untuk volume Server File Windows dengan Amazon ECS

FSx untuk Windows File Server menyediakan server file Windows yang dikelola sepenuhnya, yang didukung oleh sistem file Windows. Saat menggunakan FSx untuk Windows File Server bersama dengan ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file statis yang persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Apa itu FSx untuk Windows File Server? .

> **Note**
>
> EC2 instance yang menggunakan AMI Penuh Windows Server 2016 yang dioptimalkan Amazon ECS tidak mendukung volume tugas FSx Windows File Server ECS.

Anda tidak dapat menggunakan FSx volume Windows File Server dalam wadah Windows pada konfigurasi Fargate. Sebagai gantinya, Anda dapat memodifikasi kontainer untuk memasangnya saat startup.

Anda dapat menggunakan Windows File Server FSx untuk menyebarkan beban kerja Windows yang memerlukan akses ke penyimpanan eksternal bersama, penyimpanan Regional yang sangat tersedia, atau penyimpanan throughput tinggi. Anda dapat memasang satu atau beberapa FSx volume sistem file Windows File Server ke wadah Amazon ECS yang berjalan pada instans Amazon ECS Windows. Anda dapat berbagi FSx volume sistem file Windows File Server antara beberapa wadah Amazon ECS dalam satu tugas Amazon ECS.

Untuk mengaktifkan penggunaan FSx untuk Windows File Server dengan ECS, sertakan FSx untuk Windows File Server ID sistem file dan informasi terkait dalam definisi tugas. Ini ada dalam contoh cuplikan JSON definisi tugas berikut. Sebelum Anda membuat dan menjalankan definisi tugas, Anda memerlukan yang berikut ini.

- EC2 Instans ECS Windows yang bergabung ke domain yang valid. Ini dapat di-host oleh Active Directory lokal atau Direktori Aktif yang dihosting sendiri di Amazon. AWS Directory Service for Microsoft Active Directory EC2
- Parameter AWS Secrets Manager rahasia atau Systems Manager yang berisi kredensyal yang digunakan untuk bergabung dengan domain Active Directory dan melampirkan sistem file FSx untuk Windows File Server. Nilai kredensial adalah kredensial nama dan kata sandi yang Anda masukkan saat membuat Direktori Aktif.

Untuk tutorial terkait, lihatPelajari cara mengonfigurasi FSx sistem file Windows File Server untuk Amazon ECS.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan FSx volume Windows File Server:

- FSx untuk Windows File Server dengan Amazon ECS hanya mendukung EC2 instance Windows Amazon. EC2Instans Linux Amazon tidak didukung.
- FSx untuk Windows File Server dengan Amazon ECS tidak mendukung AWS Fargate.
- FSx untuk Windows File Server dengan Amazon ECS dengan mode `awsvpc` jaringan memerlukan versi `1.54.0` atau yang lebih baru dari agen kontainer.

- Jumlah maksimum huruf drive yang dapat digunakan untuk tugas Amazon ECS adalah 23. Setiap tugas dengan volume FSx untuk Windows File Server mendapat huruf drive yang ditetapkan untuk itu.
- Secara default, waktu pembersihan sumber daya tugas adalah tiga jam setelah tugas berakhir. Bahkan jika tidak ada tugas yang menggunakannya, pemetaan file yang dibuat oleh tugas tetap ada selama tiga jam. Waktu pembersihan default dapat dikonfigurasi dengan menggunakan variabel lingkungan Amazon ECS. `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Tugas biasanya hanya berjalan di VPC yang sama dengan sistem file FSx untuk Windows File Server. Namun, dimungkinkan untuk memiliki dukungan lintas-VPC jika ada konektivitas jaringan yang mapan antara VPC cluster Amazon ECS dan sistem file FSx untuk Windows File Server melalui VPC peering.
- Anda mengontrol akses ke sistem file Windows File Server FSx untuk tingkat jaringan dengan mengonfigurasi grup keamanan VPC. Hanya tugas yang di-host pada EC2 instance yang digabungkan ke domain Active Directory dengan grup keamanan Active Directory yang dikonfigurasi dengan benar yang dapat mengakses berbagi file FSx untuk Windows File Server. Jika grup keamanan salah dikonfigurasi, Amazon ECS gagal meluncurkan tugas dengan pesan galat berikut: `unable to mount file system` *`fs-id`*
- FSx untuk Windows File Server terintegrasi dengan AWS Identity and Access Management (IAM) untuk mengontrol tindakan yang dapat dilakukan oleh pengguna dan grup IAM Anda secara spesifik FSx untuk sumber daya Windows File Server. Dengan otorisasi klien, pelanggan dapat menentukan peran IAM yang memungkinkan atau menolak akses ke spesifik FSx untuk sistem file Windows File Server, secara opsional memerlukan akses hanya-baca, dan secara opsional mengizinkan atau melarang akses root ke sistem file dari klien. Untuk informasi selengkapnya, lihat Keamanan di Panduan Pengguna Amazon FSx Windows.

Praktik terbaik FSx untuk menggunakan Windows File Server dengan Amazon ECS

Catat rekomendasi praktik terbaik berikut saat Anda menggunakan FSx Windows File Server dengan Amazon ECS.

Kontrol keamanan dan akses FSx untuk Windows File Server

FSx untuk Windows File Server menawarkan fitur kontrol akses berikut yang dapat Anda gunakan untuk memastikan bahwa data yang FSx disimpan dalam sistem file Windows File Server aman dan hanya dapat diakses dari aplikasi yang membutuhkannya.

Enkripsi data FSx untuk volume Windows File Server

FSx untuk Windows File Server mendukung dua bentuk enkripsi untuk sistem file. Mereka adalah enkripsi data dalam perjalanan dan enkripsi saat istirahat. Enkripsi data dalam perjalanan didukung pada berbagi file yang dipetakan pada instance kontainer yang mendukung protokol SMB 3.0 atau yang lebih baru. Enkripsi data saat istirahat diaktifkan secara otomatis saat membuat sistem FSx file Amazon. Amazon FSx secara otomatis mengenkripsi data dalam perjalanan menggunakan enkripsi SMB saat Anda mengakses sistem file Anda tanpa perlu memodifikasi aplikasi Anda. Untuk informasi selengkapnya, lihat Enkripsi data FSx di Amazon di Panduan Pengguna Server File Amazon FSx untuk Windows.

Gunakan Windows ACLs untuk kontrol akses tingkat folder

EC2 Instans Windows Amazon mengakses berbagi FSx file Amazon menggunakan kredensyal Direktori Aktif. Ini menggunakan daftar kontrol akses Windows standar (ACLs) untuk kontrol akses tingkat file dan tingkat folder berbutir halus. Anda dapat membuat beberapa kredensyal, masing-masing untuk folder tertentu dalam berbagi yang memetakan ke tugas tertentu.

Dalam contoh berikut, tugas memiliki akses ke folder `App01` menggunakan kredensi yang disimpan di Secrets Manager. Nama Sumber Daya Amazon (ARN) adalah. `1234`

```
"rootDirectory": "\\path\\to\\my\\data\App01",
"credentialsParameter": "arn-1234",
"domain": "corp.fullyqualified.com",
```

Dalam contoh lain, tugas memiliki akses ke folder `App02` menggunakan kredensi yang disimpan di Secrets Manager. ARN-nya adalah. 6789

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
"domain": "corp.fullyqualified.com",
```

Tentukan sistem file Windows File Server FSx untuk Windows dalam definisi tugas Amazon ECS

FSx Untuk menggunakan volume sistem file Windows File Server untuk container Anda, tentukan konfigurasi volume dan mount point dalam definisi tugas Anda. Definisi tugas berikut JSON cuplikan menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
```

```
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>It Works!</h2> <p>You are using Amazon
 FSx for Windows File Server file system for persistent container storage.</p>' -
Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force; Start-
Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -Destination C:\
\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
```

```
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

`FSxWindowsFileServerVolumeConfiguration`

Tipe: Objek

Wajib: Tidak

Parameter ini ditentukan saat Anda menggunakan FSx sistem file Windows File Server untuk penyimpanan tugas.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Windows File Server untuk digunakan. FSx

`rootDirectory`

Tipe: String

Diperlukan: Ya

Direktori dalam sistem file FSx untuk Windows File Server untuk dipasang sebagai direktori root di dalam host.

`authorizationConfig`

`credentialsParameter`

Tipe: String

Wajib: Ya

Opsi kredensial otorisasi:

- Nama Sumber Daya Amazon (ARN) dari rahasia Secrets Manager.
- Nama Sumber Daya Amazon (ARN) dari parameter Systems Manager.

`domain`

Tipe: String

Diperlukan: Ya

Nama domain yang sepenuhnya memenuhi syarat yang dihosting oleh direktori AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) atau Direktori EC2 Aktif yang dihosting sendiri.

Metode FSx untuk menyimpan kredensyal volume Windows File Server

Ada dua metode yang berbeda untuk menyimpan kredensial untuk digunakan dengan parameter kredensial.

- AWS Secrets Manager rahasia

  Kredensi ini dapat dibuat di AWS Secrets Manager konsol dengan menggunakan kategori rahasia jenis lain. Anda menambahkan baris untuk setiap key/value pasangan, username/admin dan kata sandi/*`password`*.
- Parameter Systems Manager

Kredensyal ini dapat dibuat di konsol parameter Systems Manager dengan memasukkan teks dalam formulir yang ada di cuplikan kode contoh berikut.

```
{
  "username": "admin",
  "password": "password"
}
```

`FSxWindowsFileServerVolumeConfiguration`Parameter `credentialsParameter` dalam definisi tugas memegang ARN rahasia atau parameter Systems Manager ARN. Untuk informasi selengkapnya, lihat Apa itu AWS Secrets Manager di Panduan Pengguna Secrets Manager dan Parameter Store Systems Manager dari Panduan Pengguna Systems Manager.

Pelajari cara mengonfigurasi FSx sistem file Windows File Server untuk Amazon ECS

Pelajari cara meluncurkan instans Windows Amazon ECS yang dioptimalkan FSx untuk menghosting sistem file Windows File Server dan container yang dapat mengakses sistem file. Untuk melakukan ini, pertama-tama Anda membuat Direktori Aktif Microsoft AWS Directory Service AWS Terkelola. Kemudian, Anda membuat sistem file Server File Server File Server FSx untuk Windows dan cluster dengan EC2 instance Amazon dan definisi tugas. Anda mengonfigurasi definisi tugas untuk wadah Anda untuk menggunakan sistem file Windows File Server. FSx Akhirnya, Anda menguji sistem file.

Dibutuhkan 20 hingga 45 menit setiap kali Anda meluncurkan atau menghapus baik Active Directory atau FSx untuk sistem file Windows File Server. Bersiaplah untuk menyimpan setidaknya 90 menit untuk menyelesaikan tutorial atau menyelesaikan tutorial selama beberapa sesi.

Prasyarat untuk tutorial

- Pengguna administratif. Lihat Siapkan untuk menggunakan Amazon ECS.
- (Opsional) Sebuah `PEM` key pair untuk menghubungkan ke instance EC2 Windows Anda melalui akses RDP. Untuk informasi tentang cara membuat pasangan kunci, lihat pasangan EC2 kunci Amazon dan EC2 instans Amazon di Panduan EC2 Pengguna Amazon.
- Sebuah VPC dengan setidaknya satu subnet publik dan satu subnet privat, dan satu grup keamanan. Anda dapat menggunakan VPC default Anda. Anda tidak memerlukan gateway atau perangkat NAT. AWS Directory Service tidak mendukung Network Address Translation (NAT) dengan Direktori Aktif. Agar ini berfungsi, Active Directory, FSx untuk sistem file Windows File Server, ECS Cluster, dan EC2 instance harus berada di dalam VPC Anda. Untuk informasi

selengkapnya mengenai VPCs dan Direktori Aktif, lihat Membuat VPC dan Prasyarat untuk membuat iklan Microsoft Terkelola. AWS

- Izin IAM ecsInstanceRole dan ecsTaskExecution Peran dikaitkan dengan akun Anda. Peran terkait layanan ini memungkinkan layanan melakukan panggilan API dan mengakses kontainer, rahasia, direktori, dan server file atas nama Anda.

Langkah 1: Buat peran akses IAM

Buat klaster dengan AWS Management Console.

1. Lihat Peran IAM instance wadah Amazon ECS untuk memeriksa apakah Anda memiliki ecsInstanceRole dan untuk melihat bagaimana Anda dapat membuatnya jika Anda tidak memilikinya.
2. Kami merekomendasikan bahwa kebijakan peran disesuaikan untuk izin minimum di lingkungan produksi aktual. Untuk tujuan mengerjakan tutorial ini, verifikasi bahwa kebijakan AWS terkelola berikut ini melekat pada AndaecsInstanceRole. Lampirkan kebijakan jika belum dilampirkan.

   - EC2ContainerServiceforEC2Peran Amazon
   - Amazon SSMManaged InstanceCore
   - Amazon SSMDirectory ServiceAccess

   Untuk melampirkan kebijakan AWS terkelola.

   a. Buka konsol IAM.
   b. Di panel navigasi, pilih Peran.
   c. Pilih peran yang AWS dikelola.
   d. Pilih Izin, Lampirkan kebijakan.
   e. Untuk mempersempit kebijakan tersedia yang akan dilampirkan, gunakan Filter.
   f. Pilih kebijakan yang sesuai dan pilih Lampirkan kebijakan.
3. Lihat Peran IAM eksekusi tugas Amazon ECS untuk memeriksa apakah Anda memiliki ecsTaskExecution Peran dan untuk melihat bagaimana Anda dapat membuatnya jika Anda tidak memilikinya.

   Kami merekomendasikan bahwa kebijakan peran disesuaikan untuk izin minimum di lingkungan produksi aktual. Untuk tujuan mengerjakan tutorial ini, verifikasi bahwa kebijakan AWS terkelola

berikut dilampirkan ke ecsTaskExecution Peran Anda. Lampirkan kebijakan jika mereka belum dilampirkan. Gunakan prosedur yang diberikan di bagian sebelumnya untuk melampirkan kebijakan yang AWS dikelola.

- SecretsManagerReadWrite
- Amazon FSx ReadOnlyAccess
- Amazon SSMRead OnlyAccess
- Amazon ECSTask ExecutionRolePolicy

Langkah 2: Membuat Direktori Aktif Windows (AD)

1. Ikuti langkah-langkah yang dijelaskan dalam Membuat Microsoft AD yang AWS Dikelola di Panduan Administrasi Layanan AWS Direktori. Gunakan VPC yang telah Anda tunjuk untuk tutorial ini. Pada Langkah 3 Membuat iklan Microsoft AWS Terkelola Anda, simpan nama pengguna dan kata sandi admin untuk digunakan dalam langkah berikut. Juga, perhatikan nama DNS direktori yang sepenuhnya memenuhi syarat untuk langkah-langkah masa depan. Anda dapat menyelesaikan langkah berikut saat Active Directory sedang dibuat.
2. Buat AWS rahasia Secrets Manager untuk digunakan dalam langkah-langkah berikut. Untuk informasi selengkapnya, lihat Memulai Secrets Manager di Panduan Pengguna AWS Secrets Manager.

   a. Buka konsol Secrets Manager.

   b. Klik Menyimpan rahasia baru.

   c. Pilih Tipe rahasia lainnya.

   d. Untuk Kunci/nilai rahasia, di baris pertama, buat kunci **`username`** dengan nilai **`admin`**. Klik pada + Tambahkan baris.

   e. Di baris baru, buat kunci **`password`**. Untuk nilai, ketik kata sandi yang Anda masukkan di Langkah 3 Buat Direktori AD AWS Terkelola Anda.

   f. Klik pada tombol Selanjutnya.

   g. Masukkan nama dan deskripsi. Klik Berikutnya.

   h. Klik Berikutnya. Klik Simpan.

   i. Dari daftar halaman Rahasia, klik pada rahasia yang baru saja Anda buat.

   j. Simpan ARN rahasia baru untuk digunakan dalam langkah-langkah berikut.

   k. Anda dapat melanjutkan ke langkah berikutnya selagi Direktori Aktif sedang dibuat.

Langkah 3: Verifikasi dan perbarui grup keamanan

Pada langkah ini, Anda memverifikasi dan memperbarui aturan untuk grup keamanan yang Anda gunakan. Untuk hal ini, Anda dapat menggunakan grup keamanan default yang dibuat untuk VPC Anda.

Verifikasi dan perbarui grup keamanan.

Anda perlu membuat atau mengedit grup keamanan untuk mengirim data dari dan ke port, yang dijelaskan dalam Grup Keamanan VPC Amazon di Panduan Pengguna Server File Windows FSx untuk Windows. Anda dapat melakukan ini dengan membuat aturan grup keamanan inbound ditampilkan di baris pertama dari tabel aturan inbound berikut. Mengizinkan lalu lintas masuk dari antarmuka jaringan (dan instans terkait-nya) yang ditugaskan ke grup keamanan. Semua sumber daya cloud yang Anda buat berada dalam VPC yang sama dan melekat pada grup keamanan yang sama. Oleh karena itu, aturan ini memungkinkan lalu lintas dikirim ke dan dari sistem file Windows File Server, Active Directory, dan ECS seperti yang diperlukan. FSx Aturan inbound lainnya mengizinkan lalu lintas untuk melayani situs web dan akses RDP untuk menghubungkan ke instans ECS Anda.

Tabel berikut menunjukkan aturan grup keamanan inbound mana yang diperlukan untuk tutorial ini.

Tipe	Protokol	Rentang port	Sumber
Semua Lalu lintas	Semua	Semua	*sg-securitygroup*
HTTPS	TCP	443	0.0.0.0/0
RDP	TCP	3389	alamat IP laptop Anda

Tabel berikut menunjukkan aturan grup keamanan inbound mana yang diperlukan untuk tutorial ini.

Tipe	Protokol	Rentang Port	Tujuan
Semua lalu lintas	Semua	Semua	0.0.0.0/0

1. Buka EC2 konsol dan pilih Grup Keamanan dari menu sebelah kiri.
2. Dari daftar grup keamanan yang sekarang ditampilkan, pilih centang pada kotak centang di sebelah kiri grup keamanan yang Anda gunakan untuk tutorial ini.

   Detail grup keamanan Anda ditampilkan.
3. Edit aturan inbound dan outbound dengan memilih tab Aturan inbound atau Aturan outbound dan memilih tombol Edit aturan inbound atau Edit aturan outbound. Edit aturan untuk mencocokkan yang ditampilkan dalam tabel sebelumnya. Setelah Anda membuat EC2 instans nanti dalam tutorial ini, edit sumber RDP aturan masuk dengan alamat IP publik EC2 instans Anda seperti yang dijelaskan dalam Connect to Windows instance menggunakan RDP dari Amazon User Guide. EC2

Langkah 4: Buat sistem file Windows File Server FSx untuk Windows

Setelah grup keamanan Anda diverifikasi dan diperbarui dan Direktori Aktif Anda dibuat dan berada dalam status aktif, buat sistem file FSx untuk Windows File Server di VPC yang sama dengan Direktori Aktif Anda. Gunakan langkah-langkah berikut FSx untuk membuat sistem file Windows File Server untuk tugas Windows Anda.

Buat sistem file pertama Anda.

1. Buka FSx konsol Amazon.
2. Pada dasbor, pilih Buat sistem file untuk memulai wizard pembuatan sistem file.
3. Pada halaman Pilih jenis sistem file, pilih FSx untuk Windows File Server, lalu pilih Berikutnya. Halaman Buat sistem file akan muncul.
4. Pada bagian Detail sistem file, berikan nama untuk sistem file Anda. Penamaan sistem file Anda membuatnya lebih mudah untuk menemukan dan mengelola sistem file tersebut. Anda dapat menggunakan hingga 256 karakter Unicode. Karakter yang diizinkan adalah huruf, angka, spasi, dan karakter khusus plus tanda (+). tanda minus (-), tanda sama dengan (=), titik (.), garis bawah (_), titik dua (:), garis miring (/).
5. Untuk Tipe deployment Pilih Single-AZ untuk men-deploy sistem file yang di-deploy di Availability Zone tunggal. Single-Az 2 adalah generasi terbaru dari sistem file Availability Zone tunggal, dan mendukung penyimpanan SSD dan HDD.
6. Untuk jenis Penyimpanan, pilih HDD.
7. Untuk kapasitas penyimpanan, masukkan kapasitas penyimpanan minimum.
8. Pertahankan Kapasitas throughput pada pengaturan default-nya.

9. Di bagian Jaringan & keamanan, pilih VPC Amazon yang sama dengan yang Anda pilih untuk direktori Anda AWS Directory Service .
10. Untuk Grup Keamanan VPC, pilih grup keamanan yang Anda verifikasi di Langkah 3: Verifikasi dan perbarui grup keamanan.
11. Untuk otentikasi Windows, pilih Direktori Aktif Microsoft AWS Terkelola, lalu pilih AWS Directory Service direktori Anda dari daftar.
12. Untuk Enkripsi, simpan default pengaturan Kunci enkripsi dari aws/fsx (default).
13. Simpan pengaturan default untuk Preferensi pemeliharaan.
14. Klik pada tombol Selanjutnya.
15. Tinjau konfigurasi sistem file yang ditampilkan pada halaman Buat sistem file. Untuk referensi Anda, perhatikan pengaturan sistem file mana yang dapat Anda modifikasi setelah sistem file dibuat. Pilih Buat sistem file.
16. Catat ID sistem file. Anda perlu menggunakannya di langkah selanjutnya.

    Anda dapat melanjutkan ke langkah berikutnya untuk membuat cluster dan EC2 instance saat sistem file FSx untuk Windows File Server sedang dibuat.

Langkah 5: Buat cluster Amazon ECS

Buat cluster menggunakan konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan windows-fsx-cluster.
6. Perluas Infrastruktur, hapus AWS Fargate (tanpa server) lalu pilih instans Amazon EC2 .
   - Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:
     - Untuk Sistem Operasi/Arsitektur, pilih Windows Server 2019 Core.
     - EC2 Misalnya jenis, pilih t2.medium atau t2.micro.
7. Pilih Buat.

Langkah 6: Buat instans Amazon Amazon ECS yang dioptimalkan EC2

Buat instance penampung Amazon ECS Windows.

Untuk membuat instans Amazon ECS

1. Gunakan `aws ssm get-parameters` perintah untuk mengambil nama AMI untuk Wilayah yang menghosting VPC Anda. Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.
2. Gunakan EC2 konsol Amazon untuk meluncurkan instance.
   a. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
   b. Dari bilah navigasi, pilih Wilayah untuk digunakan.
   c. Dari EC2 Dasbor, pilih Launch instance.
   d. Untuk Nama, masukkan nama unik.
   e. Untuk Gambar Aplikasi dan OS (Gambar Mesin Amazon), di bidang pencarian, masukkan nama AMI yang Anda ambil.
   f. Untuk jenis Instance, pilih t2.medium atau t2.micro.
   g. Untuk Key pair (login), pilih key pair. Jika Anda tidak menentukan key pair, Anda
   h. Di bawah Pengaturan jaringan, untuk VPC dan Subnet, pilih VPC Anda dan subnet publik.
   i. Di bawah Pengaturan jaringan, untuk grup Keamanan, pilih grup keamanan yang ada, atau buat yang baru. Pastikan grup keamanan yang Anda pilih memiliki aturan masuk dan keluar yang ditentukan dalam Prasyarat untuk tutorial
   j. Di bawah Pengaturan jaringan, untuk Auto-assign IP Publik, pilih Aktifkan.
   k. Perluas Detail lanjutan, lalu untuk direktori Gabung Domain, pilih ID Direktori Aktif yang Anda buat. Domain opsi ini bergabung dengan AD Anda saat EC2 instans diluncurkan.
   l. Di bawah Detail lanjutan, untuk profil instans IAM, pilih ecsInstanceRole.
   m. Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna berikut. Di bawah Detail Lanjutan, tempelkan skrip berikut ke bidang Data pengguna, ganti *cluster_name* dengan nama cluster Anda.

```
<powershell>
Initialize-ECSAgent -Cluster windows-fsx-cluster -EnableTaskIAMRole
</powershell>
```

   n. Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.

o. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.

3. Buka konsol di https://console.aws.amazon.com/ecs/v2.
4. Di panel navigasi, pilih Cluster, lalu pilih. windows-fsx-cluster
5. Pilih tab Infrastructure dan verifikasi bahwa instans Anda telah terdaftar di windows-fsx-clustercluster.

Langkah 7: Daftarkan ketentuan tugas Windows

Sebelum Anda dapat menjalankan wadah Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Contoh definisi tugas berikut menampilkan halaman web sederhana. Tugas meluncurkan dua kontainer yang memiliki akses ke sistem FSx file. Kontainer pertama menulis file HTML ke sistem file. Kontainer kedua mengunduh file HTML dari sistem file dan menyediakan halaman web.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, ganti nilai untuk peran eksekusi tugas Anda dan detail tentang sistem FSx file Anda, lalu pilih Simpan.

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType
 file -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body
 {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>It
 Works!</h2> <p>You are using Amazon FSx for Windows File Server file system for
 persistent container storage.</p>' -Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
```

```
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force;
 Start-Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -
Destination C:\\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe
 w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
```

```
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

Langkah 8: Jalankan tugas dan lihat hasilnya

Sebelum menjalankan tugas, verifikasi bahwa status sistem file Windows File Server Anda FSx tersedia. Setelah tersedia, Anda dapat menjalankan tugas menggunakan ketentuan tugas yang Anda buat. Tugas dimulai dengan membuat kontainer yang mengacak file HTML di antara mereka menggunakan sistem file. Setelah pengacakan, web server melayani halaman HTML sederhana.

> **Note**
>
> Anda mungkin tidak dapat terhubung ke situs web dari VPN.

Jalankan tugas dan lihat hasilnya dengan konsol Amazon ECS.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Cluster, lalu pilih. windows-fsx-cluster
3. Pilih tab Tugas, lalu pilih Jalankan tugas baru.
4. Untuk Jenis Peluncuran, pilih EC2.
5. Di bawah konfigurasi Deployment, untuk Task Definition, pilih fsx-windows, lalu pilih Create.
6. Saat status tugas Anda SEDANG BERJALAN, pilih ID tugas.
7. Di bawah Container, saat status container1 STOPTED, pilih container2 untuk melihat detail container.

8. Di bawah rincian Container untuk container2, pilih Network binding dan kemudian klik pada alamat IP eksternal yang terkait dengan container. Peramban Anda akan membuka dan menampilkan pesan berikut.

```
Amazon ECS Sample App
It Works!
You are using Amazon FSx for Windows File Server file system for persistent
 container storage.
```

> **Note**
>
> Mungkin perlu beberapa menit agar pesan ditampilkan. Jika Anda tidak melihat pesan ini setelah beberapa menit, periksa apakah Anda tidak menjalankan VPN dan pastikan bahwa grup keamanan untuk instance container Anda mengizinkan lalu lintas HTTP jaringan masuk pada port 443.

Langkah 9: Membersihkan

> **Note**
>
> Dibutuhkan 20 hingga 45 menit FSx untuk menghapus sistem file Windows File Server atau AD. Anda harus menunggu sampai operasi penghapusan sistem file Windows File Server selesai sebelum memulai operasi penghapusan AD. FSx

Hapus FSx untuk sistem file Windows File Server.

1. Buka FSx konsol Amazon
2. Pilih tombol radio di sebelah kiri FSx untuk sistem file Windows File Server yang baru saja Anda buat.
3. Pilih Tindakan.
4. Pilih sistem file untuk dihapus.

Hapus iklan.

1. Buka konsol AWS Directory Service.

2. Pilih tombol radio di sebelah kiri AD yang baru saja Anda buat.
3. Pilih Tindakan.
4. Pilih Hapus direktori.

Hapus klaster .

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Cluster, lalu pilih. windows-fsx-cluster
3. Pilih Hapus klaster.
4. Masukkan frasa dan kemudian pilih Hapus.

Mengakhiri EC2 contoh.

1. Buka EC2 konsol Amazon.
2. Dari menu sebelah kiri, pilih Instans.
3. Centang kotak di sebelah kiri EC2 instance yang Anda buat.
4. Klik status Instance, Terminate instance.

Hapus rahasia.

1. Buka konsol Secrets Manager.
2. Pilih rahasia yang Anda buat untuk perjalanan ini.
3. Klik Tindakan.
4. Pilih Hapus rahasia.

Gunakan volume Docker dengan Amazon ECS

Saat menggunakan volume Docker, built-in driver `local` atau driver volume pihak ke tiga dapat digunakan. Volume Docker dikelola oleh Docker dan direktori dibuat di `/var/lib/docker/volumes` pada instans kontainer yang berisi data volume.

Untuk menggunakan volume Docker, tentukan `dockerVolumeConfiguration` dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Volume dalam Docker dokumentasi.

Beberapa kasus penggunaan umum untuk volume Docker adalah sebagai berikut:

- Untuk menyediakan volume data tetap untuk digunakan dengan kontainer
- Untuk berbagi volume data yang ditetapkan di lokasi yang berbeda pada kontainer yang berbeda pada instans kontainer yang sama
- Untuk menentukan volume data kosong, tidak tetap dan memasangnya pada beberapa kontainer dalam tugas yang sama
- Untuk memberikan volume data ke tugas Anda yang dikelola oleh driver pihak ketiga

Pertimbangan untuk menggunakan volume Docker

Pertimbangkan hal berikut saat menggunakan volume Docker:

- Volume Docker hanya didukung saat menggunakan tipe EC2 peluncuran atau instance eksternal.
- Kontainer Windows hanya mendukung penggunaan driver `local`.
- Jika driver pihak ketiga digunakan, pastikan itu diinstal dan aktif pada instance kontainer sebelum agen kontainer dimulai. Jika driver pihak ketiga tidak aktif sebelum agen dimulai, Anda dapat memulai ulang agen penampung menggunakan salah satu perintah berikut:
  - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

    ```
    sudo systemctl restart ecs
    ```

  - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

    ```
    sudo stop ecs && sudo start ecs
    ```

Untuk informasi tentang cara menentukan volume Docker dalam definisi tugas, lihatMenentukan volume Docker dalam definisi tugas Amazon ECS.

Menentukan volume Docker dalam definisi tugas Amazon ECS

Sebelum kontainer Anda dapat menggunakan volume data, Anda harus menentukan konfigurasi volume dan titik pemasangan dalam ketentuan tugas Anda. Bagian ini menjelaskan konfigurasi volume untuk kontainer. Untuk tugas yang menggunakan volume Docker, tentukan `dockerVolumeConfiguration`. Untuk tugas yang menggunakan volume host pemasangan terikat, tentukan `host` dan `sourcePath` opsional.

Definisi tugas berikut JSON menunjukkan sintaks untuk `volumes` dan `mountPoints` objek untuk wadah.

```
{
    "containerDefinitions": [
        {
            "mountPoints": [
                {
                    "sourceVolume": "string",
                    "containerPath": "/path/to/mount_volume",
                    "readOnly": boolean
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "string",
            "dockerVolumeConfiguration": {
                "scope": "string",
                "autoprovision": boolean,
                "driver": "string",
                "driverOpts": {
                    "key": "value"
                },
                "labels": {
                    "key": "value"
                }
            }
        }
    ]
}
```

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`dockerVolumeConfiguration`

Jenis: DockerVolumeConfigurationObjek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Docker. Volume Docker hanya didukung saat menjalankan tugas pada EC2 instance. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan pemasangan mengikat, tentukan `host` saja.

`scope`

Tipe: String

Nilai Valid: `task` | `shared`

Wajib: Tidak

Cakupan untuk volume Docker, yang menentukan siklus hidupnya. Volume Docker yang tercakup ke `task` secara otomatis disediakan saat tugas dimulai dan dihancurkan saat tugas berhenti. Volume Docker yang tercakup sebagai `shared` dipertahankan setelah tugas berhenti.

`autoprovision`

Jenis: Boolean

Nilai default: `false`

Wajib: Tidak

Jika nilai ini`true`, volume Docker dibuat jika belum ada. Bidang ini hanya digunakan jika `scope` ada`shared`. Jika `scope` ya`task`, maka parameter ini harus dihilangkan.

`driver`

Tipe: String

Wajib: Tidak

Driver volume Docker yang digunakan. Nilai driver harus sesuai dengan nama driver yang disediakan oleh Docker karena nama ini digunakan untuk penempatan tugas. Jika driver diinstal dengan menggunakan CLI plugin Docker, `docker plugin ls` gunakan untuk mengambil nama driver dari instance container Anda. Jika driver diinstal dengan menggunakan metode lain, gunakan penemuan plugin Docker untuk mengambil nama driver.

`driverOpts`

Tipe: String

Wajib: Tidak

Peta opsi khusus driver Docker untuk dilewati. Parameter ini dipetakan ke `DriverOpts` bagian Create a volume dari Docker.

`labels`

Tipe: String

Wajib: Tidak

Metadata kustom untuk ditambahkan ke volume Docker.

`mountPoints`

Tipe: Array objek

Wajib: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` dalam create-container Docker API dan opsi `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive. Anda harus menentukan titik pemasangan untuk melampirkan volume Amazon EBS langsung ke tugas Amazon ECS.

`sourceVolume`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Untuk tugas yang berjalan pada EC2 instance yang menjalankan sistem operasi Windows, biarkan nilainya sebagai default. `false`

Contoh volume Docker untuk Amazon ECS

Contoh berikut menunjukkan cara menyediakan penyimpanan sementara untuk wadah dan cara menyediakan volume bersama untuk beberapa conatiner, dan cara menyediakan penyimpanan persisten NFS untuk sebuah wadah.

Untuk menyediakan penyimpanan sementara untuk wadah menggunakan volume Docker

Dalam contoh ini, wadah menggunakan volume data kosong yang dibuang setelah tugas selesai. Salah satu contoh kasus penggunaan adalah Anda mungkin memiliki wadah yang perlu mengakses beberapa lokasi penyimpanan file awal selama tugas. Tugas ini dapat dicapai dengan menggunakan volume Docker.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, kami menentukan ruang lingkup sebagai `task` jadi volume dihapus setelah tugas berhenti dan menggunakan built-in driver `local`.

```
"volumes": [
    {
        "name": "scratch",
        "dockerVolumeConfiguration" : {
            "scope": "task",
            "driver": "local",
            "labels": {
                "scratch": "space"
            }
        }
    }
]
```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama volume yang didefinisikan dan nilai `containerPath` untuk memasang volume pada kontainer.

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
            {
              "sourceVolume": "scratch",
              "containerPath": "/var/scratch"
            }
        ]
    }
]
```

Untuk menyediakan penyimpanan persisten untuk beberapa kontainer menggunakan volume Docker

Dalam contoh ini, Anda ingin volume bersama untuk beberapa kontainer untuk digunakan dan Anda ingin tetap ada setelah satu tugas yang menggunakannya dihentikan. `local`Driver bawaan sedang digunakan. Ini agar volume masih terikat dengan siklus hidup instance container.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, tentukan `shared` cakupan sehingga volume tetap ada, setel penyediaan otomatis ke. `true` Ini agar volume dibuat untuk digunakan. Kemudian, gunakan juga `local` driver bawaan.

   ```
   "volumes": [
       {
           "name": "database",
           "dockerVolumeConfiguration" : {
               "scope": "shared",
               "autoprovision": true,
               "driver": "local",
               "labels": {
                   "database": "database_name"
               }
           }
       }
   ]
   ```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama volume yang didefinisikan dan nilai `containerPath` untuk memasang volume pada kontainer.

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    },
    {
      "name": "container-2",
      "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    }
  ]
```

Untuk menyediakan penyimpanan persisten NFS untuk wadah menggunakan volume Docker

Dalam contoh ini, kontainer menggunakan volume data NFS yang dipasang secara otomatis saat tugas dimulai dan dilepas saat tugas berhenti. Ini menggunakan `local` driver bawaan Docker. Salah satu contoh kasus penggunaan adalah Anda mungkin memiliki penyimpanan NFS lokal dan perlu mengaksesnya dari tugas ECS Anywhere. Ini dapat dicapai dengan menggunakan volume Docker dengan opsi driver NFS.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan `name` dan nilai `DockerVolumeConfiguration`. Dalam contoh ini, tentukan `task` cakupan sehingga volume dilepas setelah tugas berhenti. Gunakan `local` driver dan konfigurasikan `driverOpts` dengan`type`,`device`, dan o opsi yang sesuai. Ganti `NFS_SERVER` dengan endpoint server NFS.

```
"volumes": [
       {
           "name": "NFS",
           "dockerVolumeConfiguration" : {
               "scope": "task",
               "driver": "local",
```

```
                "driverOpts": {
                    "type": "nfs",
                    "device": "$NFS_SERVER:/mnt/nfs",
                    "o": "addr=$NFS_SERVER"
                }
            }
        }
    ]
```

2. Di `containerDefinitions` bagian tersebut, tentukan wadah dengan `mountPoints` nilai yang mereferensikan nama volume yang ditentukan dan `containerPath` nilai untuk memasang volume pada wadah.

```
"containerDefinitions": [
        {
            "name": "container-1",
            "mountPoints": [
                {
                  "sourceVolume": "NFS",
                  "containerPath": "/var/nfsmount"
                }
            ]
        }
    ]
```

Gunakan bind mount dengan Amazon ECS

Dengan bind mount, file atau direktori pada host, seperti EC2 instance Amazon, dipasang ke dalam wadah. Bind mount didukung untuk tugas yang di-host di instance Fargate dan Amazon. EC2 Dudukan pengikat diikat ke siklus hidup wadah yang menggunakannya. Setelah semua container yang menggunakan bind mount dihentikan, seperti saat tugas dihentikan, data akan dihapus. Untuk tugas yang di-host di EC2 instans Amazon, data dapat dikaitkan dengan siklus hidup EC2 instans Amazon host dengan menentukan `sourcePath` nilai `host` dan opsional dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Bind mount di dokumentasi Docker.

Berikut ini adalah kasus penggunaan umum untuk pemasangan terikat.

- Untuk menyediakan volume data kosong untuk memasang dalam satu atau lebih kontainer.
- Untuk memasang volume data host dalam satu atau lebih kontainer.
- Untuk berbagi volume data dari kontainer sumber dengan kontainer lain dalam tugas yang sama.

- Untuk mengekspos jalur dan isinya dari Dockerfile ke satu atau lebih kontainer.

Pertimbangan saat menggunakan pemasangan terikat

Saat menggunakan bind mount, pertimbangkan hal berikut.

- Secara default, tugas yang di-host saat AWS Fargate menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows) menerima minimal 20 GiB penyimpanan sementara untuk bind mount. Anda dapat meningkatkan jumlah total penyimpanan sementara hingga maksimum 200 GiB dengan menentukan `ephemeralStorage` parameter dalam definisi tugas Anda.
- Untuk mengekspos file dari Dockerfile ke volume data saat tugas dijalankan, bidang data Amazon ECS mencari arahan. `VOLUME` Jika jalur absolut yang ditentukan dalam `VOLUME` direktif sama dengan yang ditentukan dalam definisi tugas, data di jalur `VOLUME` direktif akan disalin ke volume data. `containerPath` Dalam contoh Dockerfile berikut, file yang diberi nama `examplefile` dalam `/var/log/exported` direktori ditulis ke host dan kemudian dipasang di dalam wadah.

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN mkdir -p /var/log/exported
  RUN touch /var/log/exported/examplefile
  VOLUME ["/var/log/exported"]
  ```

  Secara default, izin volume diatur ke `0755` dan pemilik sebagai `root`. Anda dapat menyesuaikan izin ini di Dockerfile. Contoh berikut mendefinisikan pemilik direktori sebagai `node`.

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN yum install -y shadow-utils && yum clean all
  RUN useradd node
  RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
  RUN touch /var/log/exported/examplefile
  USER node
  VOLUME ["/var/log/exported"]
  ```

- Untuk tugas yang di-host di EC2 instance Amazon, saat `sourcePath` nilai `host` dan tidak ditentukan, daemon Docker mengelola pemasangan pengikat untuk Anda. Ketika tidak ada kontainer yang mereferensikan pemasangan pengikat ini, layanan pembersihan tugas agen penampung Amazon ECS akhirnya menghapusnya. Secara default, ini terjadi tiga jam setelah wadah keluar. Namun, Anda dapat mengonfigurasi durasi ini dengan variabel `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` agen. Untuk informasi selengkapnya, lihat

Konfigurasi agen kontainer Amazon ECS. Jika Anda memerlukan data ini untuk bertahan di luar siklus hidup penampung, tentukan `sourcePath` nilai untuk bind mount.

Menentukan pemasangan pengikat dalam definisi tugas Amazon ECS

Untuk tugas Amazon ECS yang di-host di instance Fargate atau EC2 Amazon, cuplikan JSON definisi tugas berikut menunjukkan sintaks untuk`mountPoints`,, dan objek untuk `volumes` definisi tugas. `ephemeralStorage`

```
{
   "family": "",
   ...
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
               "containerPath" : "/path/to/mount_volume",
               "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
          "name" : "string"
       }
    ],
    "ephemeralStorage": {
    "sizeInGiB": integer
    }
}
```

Untuk tugas Amazon ECS yang dihosting di EC2 instans Amazon, Anda dapat menggunakan `host` parameter opsional dan `sourcePath` saat menentukan detail volume tugas. Ketika ditentukan, itu mengikat mount ke siklus hidup tugas daripada wadah.

```
"volumes" : [
    {
        "host" : {
```

```
            "sourcePath" : "string"
        },
        "name" : "string"
    }
]
```

Berikut ini adalah penjelasan yang lebih detail untuk setiap parameter ketentuan tugas.

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`host`

Wajib: Tidak

`host`Parameter ini digunakan untuk mengikat siklus hidup bind mount ke EC2 instance Amazon host, bukan tugas, dan tempat penyimpanannya. Jika `host` parameternya kosong, maka daemon Docker menetapkan jalur host untuk volume data Anda, tetapi data tidak dijamin akan bertahan setelah wadah yang terkait dengannya berhenti berjalan.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`.

> **Note**
>
> `sourcePath`Parameter hanya didukung saat menggunakan tugas yang di-host di EC2 instans Amazon.

`sourcePath`

Tipe: String

Wajib: Tidak

Saat `host` parameter digunakan, tentukan a `sourcePath` untuk mendeklarasikan jalur pada EC2 instance Amazon host yang disajikan ke wadah. Jika parameter ini kosong, daemon Docker akan menetapkan jalur host untuk Anda. Jika `host` parameter berisi lokasi `sourcePath` file, maka volume data tetap ada di lokasi yang ditentukan pada EC2 instance Amazon host hingga Anda menghapusnya secara manual. Jika `sourcePath` nilai tidak ada pada EC2 instance Amazon host, daemon Docker membuatnya. Jika lokasinya memang ada, konten dari folder jalur sumber diekspor.

`mountPoints`

Tipe: Array objek

Wajib: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` dalam create-container Docker API dan opsi `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive. Anda harus menentukan titik pemasangan untuk melampirkan volume Amazon EBS langsung ke tugas Amazon ECS.

`sourceVolume`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Untuk tugas yang berjalan pada EC2 instance yang menjalankan sistem operasi Windows, biarkan nilainya sebagai default. `false`

`ephemeralStorage`

Tipe: Objek

Wajib: Tidak

Jumlah penyimpanan sementara yang dialokasikan untuk tugas. Parameter ini digunakan untuk memperluas jumlah total penyimpanan sementara yang tersedia, di luar jumlah default, untuk tugas yang dihosting AWS Fargate menggunakan versi platform atau yang lebih baru (Linux) `1.4.0` atau `1.0.0` atau yang lebih baru (Windows).

Anda dapat menggunakan Copilot CLI CloudFormation, AWS SDK atau CLI untuk menentukan penyimpanan sementara untuk pemasangan pengikat.

Contoh Bind mount untuk Amazon ECS

Contoh berikut mencakup kasus penggunaan umum untuk menggunakan dudukan pengikat untuk wadah Anda.

Untuk mengalokasikan peningkatan jumlah ruang penyimpanan sementara untuk tugas Fargate

Untuk tugas Amazon ECS yang di-host di Fargate menggunakan `1.4.0` versi platform atau yang lebih baru (Linux) `1.0.0` atau (Windows), Anda dapat mengalokasikan lebih dari jumlah penyimpanan sementara default untuk wadah dalam tugas yang akan digunakan. Contoh ini dapat dimasukkan ke dalam contoh lain untuk mengalokasikan lebih banyak penyimpanan sementara untuk tugas Fargate Anda.

- Dalam ketentuan tugas, tentukan objek `ephemeralStorage`. `sizeInGiB` harus integer antara nilai-nilai `21` dan `200` dinyatakan dalam GiB.

```
"ephemeralStorage": {
    "sizeInGiB": integer
}
```

Untuk menyediakan volume data kosong untuk satu atau lebih kontainer.

Dalam beberapa kasus, Anda ingin menyediakan kontainer dalam tugas beberapa ruang scratch. Misalnya, Anda mungkin memiliki dua wadah database yang perlu mengakses lokasi penyimpanan file awal yang sama selama tugas. Hal ini dapat dicapai dengan menggunakan pemasangan terikat.

1. Dalam ketentuan tugas bagian `volumes`, tentukan pemasangan terikat dengan nama `database_scratch`.

```
  "volumes": [
    {
      "name": "database_scratch"
    }
  ]
```

2. Di `containerDefinitions` bagian ini, buat definisi wadah database. Ini agar mereka memasang volume.

```
"containerDefinitions": [
    {
      "name": "database1",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    },
    {
      "name": "database2",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
```

```
      ]
    }
  ]
```

Untuk mengekspos jalur dan isinya dari Dockerfile ke kontainer.

Dalam contoh ini, Anda memiliki Dockerfile yang menulis data yang ingin Anda pasang di dalam kontainer. Contoh ini berfungsi untuk tugas yang di-host di instance Fargate atau Amazon EC2 .

1. Buat Dockerfile. Contoh berikut menggunakan image kontainer Amazon Linux 2 publik dan membuat file yang diberi nama `examplefile` di `/var/log/exported` direktori yang ingin kita pasang di dalam wadah. Direktif `VOLUME` harus menentukan jalur absolut.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN mkdir -p /var/log/exported
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

   Secara default, izin volume diatur ke `0755` dan pemilik sebagai `root`. Izin ini dapat diubah di Dockerfile. Contoh berikut mendefinisikan pemilik direktori `/var/log/exported` diatur ke `node`.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN yum install -y shadow-utils && yum clean all
   RUN useradd node
   RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
   USER node
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

2. Dalam ketentuan tugas bagian `volumes`, tentukan volume dengan nama `application_logs`.

   ```
   "volumes": [
     {
       "name": "application_logs"
     }
   ]
   ```

3. Di `containerDefinitions` bagian ini, buat definisi wadah aplikasi. Ini agar mereka memasang penyimpanan. `containerPath`Nilai harus cocok dengan jalur absolut yang ditentukan dalam `VOLUME` arahan dari Dockerfile.

```
"containerDefinitions": [
  {
    "name": "application1",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  },
  {
    "name": "application2",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  }
]
```

Untuk menyediakan volume data kosong untuk container yang terkait dengan siklus hidup instance Amazon host EC2

Untuk tugas yang di-host di EC2 instans Amazon, Anda dapat menggunakan mount bind dan mengikat data ke siklus hidup instans Amazon host. EC2 Anda dapat melakukan ini dengan menggunakan `host` parameter dan menentukan `sourcePath` nilai. File apa pun yang ada di `sourcePath` disajikan ke wadah pada `containerPath` nilainya. File apa pun yang ditulis dengan `containerPath` nilai ditulis ke `sourcePath` nilai pada EC2 instance Amazon host.

> **Important**
>
> Amazon ECS tidak menyinkronkan penyimpanan Anda di seluruh EC2 instans Amazon. Tugas yang menggunakan penyimpanan persisten dapat ditempatkan pada EC2 instans Amazon apa pun di klaster Anda yang memiliki kapasitas yang tersedia. Jika tugas Anda memerlukan penyimpanan persisten setelah berhenti dan memulai ulang, selalu tentukan EC2 instance Amazon yang sama pada waktu peluncuran tugas dengan perintah AWS CLI start-task. Anda juga dapat menggunakan volume Amazon EFS untuk penyimpanan persisten. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.

1. Dalam ketentuan tugas bagian `volumes`, tentukan pemasangan terikat dengan `name` dan nilai `sourcePath`. Dalam contoh berikut, EC2 instance Amazon host berisi data `/ecs/webdata` yang ingin Anda pasang di dalam wadah.

```
"volumes": [
  {
    "name": "webdata",
    "host": {
      "sourcePath": "/ecs/webdata"
    }
  }
]
```

2. Di bagian `containerDefinitions`, mendefiniskan sebuah kontainer dengan nilai-nilai `mountPoints` yang mereferensikan nama pemasangan terikat dan nilai `containerPath` untuk memasang pemasangan terikat pada kontainer.

```
"containerDefinitions": [
  {
    "name": "web",
    "image": "public.ecr.aws/docker/library/nginx:latest",
    "cpu": 99,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
```

```
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webdata",
          "containerPath": "/usr/share/nginx/html"
        }
      ]
    }
  ]
```

Untuk memasang volume yang ditetapkan pada beberapa kontainer di lokasi yang berbeda

Anda dapat menentukan volume data dalam ketentuan tugas dan memasang volume tersebut di lokasi yang berbeda pada kontainer yang berbeda. Misalnya, wadah host Anda memiliki folder data situs web di`/data/webroot`. Anda mungkin ingin me-mount volume data tersebut sebagai read-only pada dua server web berbeda yang memiliki akar dokumen yang berbeda.

1. Dalam ketentuan tugas bagian `volumes`, tentukan volume data dengan nama `webroot` dan jalur sumber `/data/webroot`.

   ```
   "volumes": [
     {
       "name": "webroot",
       "host": {
         "sourcePath": "/data/webroot"
       }
     }
   ]
   ```

2. Di bagian `containerDefinitions`, tentukan kontainer untuk setiap server web dengan nilai-nilai `mountPoints` yang mengasosiasikan volume `webroot` dengan nilai `containerPath` menunjuk ke akar dokumen untuk kontainer itu.

   ```
   "containerDefinitions": [
     {
       "name": "web-server-1",
       "image": "my-repo/ubuntu-apache",
       "cpu": 100,
       "memory": 100,
       "portMappings": [
   ```

```
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "web-server-2",
      "image": "my-repo/sles11-apache",
      "cpu": 100,
      "memory": 100,
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/srv/www/htdocs",
          "readOnly": true
        }
      ]
    }
  ]
```

Untuk memasang volume dari kontainer lain menggunakan **`volumesFrom`**.

Untuk tugas yang dihosting di EC2 instans Amazon, Anda dapat menentukan satu atau beberapa volume pada penampung, lalu menggunakan `volumesFrom` parameter dalam definisi penampung yang berbeda dalam tugas yang sama untuk memasang semua volume dari titik pemasangan

yang ditentukan semula. `sourceContainer` Parameter`volumesFrom` berlaku untuk volume yang didefinisikan dalam ketentuan tugas, dan volume yang dibangun ke dalam citra dengan Dockerfile.

1. (Opsional) Untuk berbagi volume yang dibangun ke dalam gambar, gunakan `VOLUME` instruksi di Dockerfile. Contoh berikut Dockerfile menggunakan `httpd` gambar, dan kemudian menambahkan volume dan memasangnya `dockerfile_volume` di root dokumen Apache. Ini adalah folder yang digunakan oleh server `httpd` web.

   ```
   FROM httpd
   VOLUME ["/usr/local/apache2/htdocs/dockerfile_volume"]
   ```

   Anda dapat membangun citra dengan Dockerfile ini dan mendorongnya ke repositori, seperti Docker Hub, dan menggunakannya dalam ketentuan tugas Anda. Contoh `my-repo/httpd_dockerfile_volume` gambar yang digunakan dalam langkah-langkah berikut dibuat dengan Dockerfile sebelumnya.

2. Buat ketentuan tugas yang mendefinisikan volume lain dan memasang titik untuk kontainer. Dalam contoh bagian `volumes` ini, Anda akan membuat volume kosong yang disebut `empty`, yang dikelola oleh daemon Docker. Ada juga volume host yang ditentukan yang disebut`host_etc`. Ini mengekspor `/etc` folder pada instance wadah host.

   ```
   {
     "family": "test-volumes-from",
     "volumes": [
       {
         "name": "empty",
         "host": {}
       },
       {
         "name": "host_etc",
         "host": {
           "sourcePath": "/etc"
         }
       }
     ],
   ```

   Di bagian ketentuan kontainer, buat kontainer yang memasang volume yang didefinisikan sebelumnya. Dalam contoh ini, `web` wadah memasang `host_etc` volume `empty` dan. Ini adalah wadah yang menggunakan gambar yang dibangun dengan volume di Dockerfile.

```
"containerDefinitions": [
    {
      "name": "web",
      "image": "my-repo/httpd_dockerfile_volume",
      "cpu": 100,
      "memory": 500,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "empty",
          "containerPath": "/usr/local/apache2/htdocs/empty_volume"
        },
        {
          "sourceVolume": "host_etc",
          "containerPath": "/usr/local/apache2/htdocs/host_etc"
        }
      ],
      "essential": true
    },
```

Buat kontainer lain yang menggunakan `volumesFrom` untuk memasang semua volume yang terkait dengan kontainer `web`. Semua volume pada `web` wadah juga dipasang pada `busybox` wadah. Ini termasuk volume yang ditentukan dalam Dockerfile yang digunakan untuk membangun gambar. `my-repo/httpd_dockerfile_volume`

```
    {
      "name": "busybox",
      "image": "busybox",
      "volumesFrom": [
        {
          "sourceContainer": "web"
        }
      ],
      "cpu": 100,
      "memory": 500,
      "entryPoint": [
```

```
        "sh",
        "-c"
      ],
      "command": [
        "echo $(date) > /usr/local/apache2/htdocs/empty_volume/date && echo $(date)
 > /usr/local/apache2/htdocs/host_etc/date && echo $(date) > /usr/local/apache2/
htdocs/dockerfile_volume/date"
      ],
      "essential": false
    }
  ]
}
```

Saat tugas ini dijalankan, kedua kontainer memasang volume, dan `command` dalam `busybox` wadah menulis tanggal dan waktu ke file. File ini dipanggil `date` di setiap folder volume. Folder kemudian terlihat di situs web yang ditampilkan oleh kontainer `web`.

> **Note**
>
> Karena kontainer `busybox` menjalankan perintah cepat dan kemudian keluar, itu harus ditetapkan sebagai `"essential": false` dalam ketentuan kontainer. Jika tidak, kontainer berhenti seluruh tugas ketika keluar.

Mengelola ruang memori swap kontainer di Amazon ECS

Dengan Amazon ECS, Anda dapat mengontrol penggunaan ruang memori swap pada instans EC2 Amazon berbasis Linux di tingkat penampung. Menggunakan konfigurasi swap per kontainer, setiap kontainer dalam definisi tugas dapat mengaktifkan swap atau dinonaktifkan. Bagi mereka yang mengaktifkannya, jumlah maksimum ruang swap yang digunakan dapat dibatasi. Misalnya, container latency-critical dapat menonaktifkan swap. Sebaliknya, wadah dengan permintaan memori transien yang tinggi dapat mengaktifkan swap untuk mengurangi kemungkinan out-of-memory kesalahan saat wadah sedang dimuat.

Konfigurasi swap untuk kontainer dikelola oleh parameter definisi kontainer berikut.

`maxSwap`

Jumlah total memori tukar (dalam MiB) yang dapat digunakan oleh kontainer. Parameter ini diterjemahkan ke `--memory-swap` opsi untuk docker run di mana nilainya adalah jumlah memori kontainer ditambah nilainya. `maxSwap`

Jika nilai `maxSwap` sebesar `0` ditentukan, kontainer tidak menggunakan swap. Nilai yang diterima adalah `0` atau bilangan bulat positif. Jika `maxSwap` parameter diabaikan, kontainer menggunakan konfigurasi swap untuk instans kontainer tempatnya berjalan. Nilai `maxSwap` harus ditetapkan untuk parameter `swappiness` yang akan digunakan.

`swappiness`

Anda dapat menggunakan ini untuk menyetel perilaku swappiness memori kontainer. `swappiness`Nilai `0` penyebab pertukaran tidak terjadi kecuali diperlukan. `swappiness`Nilai `100` penyebab halaman ditukar secara agresif. Nilai yang diterima adalah bilangan bulat antara `0` dan `100`. Jika parameter `swappiness` tidak ditentukan, nilai default sebesar `60` akan digunakan. Jika nilai tidak ditentukan untuk`maxSwap`, parameter ini diabaikan. Parameter ini memetakan ke `--memory-swappiness` opsi untuk docker run.

Dalam contoh berikut, sintaks JSON disediakan.

```
"containerDefinitions": [{
        ...
        "linuxParameters": {
            "maxSwap": integer,
            "swappiness": integer
        },
        ...
}]
```

Pertimbangan

Pertimbangkan hal berikut ketika Anda menggunakan konfigurasi swap per kontainer.

- Ruang swap harus diaktifkan dan dialokasikan pada EC2 instans Amazon yang menghosting tugas Anda agar kontainer dapat digunakan. Secara default, Amazon ECS yang dioptimalkan AMIs tidak mengaktifkan swap. Anda harus mengaktifkan swap di instans untuk menggunakan fitur ini. Untuk informasi selengkapnya, lihat Volume Swap Toko Instance di Panduan EC2 Pengguna Amazon

atau Bagaimana cara mengalokasikan memori agar berfungsi sebagai ruang swap di EC2 instans Amazon? .

- Parameter definisi wadah ruang swap hanya didukung untuk definisi tugas yang menentukan jenis EC2 peluncuran. Parameter ini tidak didukung untuk definisi tugas yang ditujukan hanya untuk Amazon ECS pada penggunaan Fargate.
- Fitur ini hanya didukung untuk kontainer Linux. Kontainer Windows tidak didukung saat ini.
- Jika parameter definisi `maxSwap` dan `swappiness` kontainer dihilangkan dari definisi tugas, setiap kontainer memiliki `swappiness` nilai default. `60` Selain itu, total penggunaan swap dibatasi hingga dua kali memori kontainer.
- Jika Anda menggunakan tugas di Amazon Linux 2023, `swappiness` parameter tidak didukung.

Perbedaan definisi tugas Amazon ECS untuk jenis peluncuran Fargate

Untuk menggunakan Fargate, Anda harus mengonfigurasi definisi tugas Anda untuk menggunakan jenis peluncuran Fargate. Ada pertimbangan tambahan saat menggunakan Fargate.

Parameter ketentuan tugas

Tugas yang menggunakan tipe peluncuran Fargate tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda untuk tugas Fargate.

Parameter definisi tugas berikut tidak valid dalam tugas Fargate:

- `disableNetworking`
- `dnsSearchDomains`
- `dnsServers`
- `dockerSecurityOptions`
- `extraHosts`
- `gpu`
- `ipcMode`
- `links`
- `placementConstraints`
- `privileged`

- `maxSwap`
- `swappiness`

Parameter definisi tugas berikut valid dalam tugas Fargate, tetapi memiliki batasan yang harus diperhatikan:

- `linuxParameters`— Saat menentukan opsi khusus Linux yang diterapkan ke wadah, satu-satunya kemampuan yang dapat Anda tambahkan adalah. `capabilities` `CAP_SYS_PTRACE` Parameter `devices`, `sharedMemorySize`, dan `tmpfs` tidak didukung. Untuk informasi selengkapnya, lihat Parameter Linux.
- `volumes`— Tugas Fargate hanya mendukung volume host bind mount, sehingga `dockerVolumeConfiguration` parameternya tidak didukung. Untuk informasi selengkapnya, lihat Volume.
- `cpu`- Untuk wadah Windows aktifAWS Fargate, nilainya tidak boleh kurang dari 1 vCPU.
- `networkConfiguration`- Tugas Fargate selalu menggunakan mode `awsvpc` jaringan.

Untuk memastikan bahwa definisi tugas Anda memvalidasi untuk digunakan dengan Fargate, Anda dapat menentukan hal berikut saat mendaftarkan definisi tugas:

- Di bidang AWS Management Console, untuk Memerlukan Kompatibilitas, tentukan. `FARGATE`
- Dalam AWS CLI, tentukan `--requires-compatibilities` opsi.
- Di Amazon ECS API, tentukan `requiresCompatibilities` flag.

Sistem Operasi dan Arsitektur

Saat Anda mengonfigurasi definisi tugas dan kontainerAWS Fargate, Anda harus menentukan Sistem Operasi tempat penampung berjalan. Sistem Operasi berikut didukung untukAWS Fargate:

- Amazon Linux 2

> **Note**
>
> Kontainer Linux hanya menggunakan konfigurasi kernel dan kernel dari Sistem Operasi host. Misalnya, konfigurasi kernel mencakup kontrol `sysctl` sistem. Gambar kontainer Linux dapat dibuat dari gambar dasar yang berisi file dan program dari distribusi Linux apa

pun. Jika arsitektur CPU cocok, Anda dapat menjalankan container dari image container Linux apa pun di Sistem Operasi apa pun.

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Lengkap
- Windows Server 2022 Inti

Ketika Anda menjalankan wadah WindowsAWS Fargate, Anda harus memiliki arsitektur CPU X86_64.

Saat menjalankan container LinuxAWS Fargate, Anda dapat menggunakan arsitektur CPU X86_64, atau ARM64 arsitektur untuk aplikasi berbasis ARM Anda. Untuk informasi selengkapnya, lihat the section called “Definisi tugas untuk beban kerja ARM 64-bit”.

CPU dan memori tugas

Definisi tugas Amazon ECS untuk AWS Fargate mengharuskan Anda menentukan CPU dan memori pada tingkat tugas. Sebagian besar kasus penggunaan terpenuhi dengan hanya menentukan sumber daya ini pada tingkat tugas. Tabel berikut menunjukkan kombinasi yang valid antara CPU dan memori tingkat tugas. Anda dapat menentukan nilai memori dalam definisi tugas sebagai string di MiB atau GB. Misalnya, Anda dapat menentukan nilai memori baik seperti `3072` di MiB atau `3 GB` GB. Anda dapat menentukan nilai CPU dalam file JSON sebagai string dalam unit CPU atau virtual CPUs (vCPUs). Misalnya, Anda dapat menentukan nilai CPU baik seperti `1024` dalam unit CPU atau `1 vCPU` di vCPUs.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Jaringan tugas

Tugas Amazon ECS untuk AWS Fargate memerlukan mode `awsvpc` jaringan, yang menyediakan setiap tugas dengan antarmuka network elastis. Ketika Anda menjalankan tugas atau membuat layanan dengan mode jaringan ini, Anda harus menentukan satu subnet atau lebih untuk melampirkan antarmuka jaringan dan satu grup keamanan atau lebih untuk diterapkan ke antarmuka jaringan.

Jika Anda menggunakan subnet publik, putuskan apakah Anda ingin untuk memberikan alamat IP publik untuk antarmuka jaringan atau tidak. Untuk tugas Fargate di subnet publik untuk menarik gambar kontainer, alamat IP publik perlu ditetapkan ke antarmuka network elastis tugas, dengan rute ke internet atau gateway NAT yang dapat merutekan permintaan ke internet. Untuk tugas Fargate di subnet pribadi untuk menarik gambar kontainer, Anda memerlukan gateway NAT di subnet untuk merutekan permintaan ke internet. Saat Anda meng-host gambar penampung Anda di Amazon ECR,

Anda dapat mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka. Dalam hal ini, IPv4 alamat pribadi tugas digunakan untuk penarikan gambar. Untuk informasi selengkapnya tentang titik akhir antarmuka Amazon ECR, lihat Titik akhir VPC () antarmuka ECR Amazon ECR AWS PrivateLink di Panduan Pengguna Amazon Elastic Container Registry.

Berikut ini adalah contoh `networkConfiguration` bagian untuk layanan Fargate:

```
"networkConfiguration": {
   "awsvpcConfiguration": {
      "assignPublicIp": "ENABLED",
      "securityGroups": [ "sg-12345678" ],
      "subnets": [ "subnet-12345678" ]
   }
}
```

Batas sumber daya tugas

Definisi tugas Amazon ECS untuk kontainer Linux AWS Fargate mendukung `ulimits` parameter untuk menentukan batas sumber daya yang akan ditetapkan untuk wadah.

Definisi tugas Amazon ECS untuk Windows aktif AWS Fargate tidak mendukung `ulimits` parameter untuk menentukan batas sumber daya yang akan ditetapkan untuk wadah.

Tugas Amazon ECS yang dihosting di Fargate menggunakan nilai batas sumber daya default yang ditetapkan oleh sistem operasi dengan pengecualian parameter batas sumber daya. `nofile` Batas `nofile` sumber daya menetapkan batasan pada jumlah file terbuka yang dapat digunakan wadah. Di Fargate, batas `nofile` lunak default adalah `65535` dan batas keras adalah. `65535` Anda dapat mengatur nilai dari kedua batas hingga`1048576`.

Berikut ini adalah contoh potongan ketentuan tugas yang menunjukkan cara untuk menentukan batas `nofile` kustom yang telah dilipatgandakan:

```
"ulimits": [
    {
       "name": "nofile",
       "softLimit": 2048,
       "hardLimit": 8192
    }
]
```

Untuk informasi lebih lanjut tentang batas sumber daya lain yang dapat disesuaikan, lihat Batas sumber daya.

Pencatatan log

Pencatatan peristiwa

Amazon ECS mencatat tindakan yang diperlukan. EventBridge Anda dapat menggunakan acara Amazon ECS EventBridge untuk menerima notifikasi mendekati waktu nyata mengenai status klaster, layanan, dan tugas Amazon ECS Anda saat ini. Selain itu, Anda dapat mengotomatiskan tindakan untuk menanggapi peristiwa ini. Untuk informasi selengkapnya, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge.

Pencatatan siklus hidup tugas

Tugas yang berjalan di Fargate menerbitkan stempel waktu untuk melacak tugas melalui status siklus hidup tugas. Anda dapat melihat stempel waktu dalam detail tugas di AWS Management Console dan dengan menjelaskan tugas di dan. AWS CLI SDKs Misalnya, Anda dapat menggunakan stempel waktu untuk mengevaluasi berapa banyak waktu yang dihabiskan tugas untuk mengunduh gambar kontainer dan memutuskan apakah Anda harus mengoptimalkan ukuran gambar kontainer, atau menggunakan indeks OCI Seekable. Untuk informasi selengkapnya tentang praktik gambar kontainer, lihatPraktik terbaik untuk gambar kontainer Amazon ECS.

Pencatatan aplikasi

Definisi tugas Amazon ECS untuk AWS Fargate mendukung`awslogs`,`splunk`, dan driver `awsfirelens` log untuk konfigurasi log.

Driver `awslogs` log mengonfigurasi tugas Fargate Anda untuk mengirim informasi log ke Amazon CloudWatch Logs. Berikut ini menunjukkan potongan ketentuan tugas di tempat driver log `awslogs` dikonfigurasi:

```
"logConfiguration": {
   "logDriver": "awslogs",
   "options": {
      "awslogs-group" : "/ecs/fargate-task-definition",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
   }
}
```

Untuk informasi selengkapnya tentang penggunaan driver `awslogs` log dalam definisi tugas untuk mengirim log kontainer Anda ke CloudWatch Log, lihatKirim log Amazon ECS ke CloudWatch .

Untuk informasi selengkapnya tentang driver log `awsfirelens` dalam ketentuan tugas, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

Untuk informasi selengkapnya tentang penggunaan driver log `splunk` dalam ketentuan tugas, lihat `splunk`driver log.

Penyimpanan tugas

Untuk tugas Amazon ECS yang dihosting di Fargate, jenis penyimpanan berikut didukung:

- Volume Amazon EBS menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EBS dengan Amazon ECS.
- Volume Amazon EFS untuk penyimpanan persisten. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.
- Bind mount untuk penyimpanan sementara. Untuk informasi selengkapnya, lihat Gunakan bind mount dengan Amazon ECS.

Lazy loading gambar kontainer menggunakan Seekable OCI (SOCI)

Tugas Amazon ECS di Fargate yang menggunakan `1.4.0` versi platform Linux dapat menggunakan Seekable OCI (SOCI) untuk membantu memulai tugas lebih cepat. Dengan SOCI, kontainer hanya menghabiskan beberapa detik pada penarikan gambar sebelum mereka dapat memulai, menyediakan waktu untuk pengaturan lingkungan dan instantiasi aplikasi saat gambar diunduh di latar belakang. Ini disebut lazy loading. Saat Fargate memulai tugas Amazon ECS, Fargate secara otomatis mendeteksi apakah indeks SOCI ada untuk gambar dalam tugas dan memulai penampung tanpa menunggu seluruh gambar diunduh.

Untuk kontainer yang berjalan tanpa indeks SOCI, gambar kontainer diunduh sepenuhnya sebelum penampung dimulai. Perilaku ini sama pada semua versi platform Fargate lainnya dan pada AMI yang dioptimalkan Amazon ECS di instans Amazon. EC2

Seekable OCI (SOCI) adalah teknologi open source yang dikembangkan oleh AWS yang dapat meluncurkan kontainer lebih cepat dengan malas memuat gambar kontainer. SOCI bekerja dengan membuat indeks (Indeks SOCI) dari file dalam gambar kontainer yang ada. Indeks ini membantu

meluncurkan kontainer lebih cepat, memberikan kemampuan untuk mengekstrak file individual dari gambar kontainer sebelum mengunduh seluruh gambar. Indeks SOCI harus disimpan sebagai artefak di repositori yang sama dengan gambar dalam registri kontainer. Anda hanya boleh menggunakan indeks SOCI dari sumber tepercaya, karena indeks adalah sumber otoritatif untuk konten gambar. Untuk informasi selengkapnya, lihat Memperkenalkan OCI yang Dapat Dicari untuk memuat gambar kontainer yang lambat.

Pelanggan yang ingin menggunakan SOCI hanya akan dapat menggunakan manifes indeks SOCI v2. Pelanggan lama yang sebelumnya telah menggunakan SOCI di Fargate dapat terus menggunakan SOCI Index Manifest v1, namun kami sangat menyarankan pelanggan tersebut untuk bermigrasi ke SOCI Index Manifest v2. SOCI Index Manifest v2 menciptakan hubungan eksplisit antara gambar kontainer dan indeks SOCI mereka untuk memastikan penerapan yang konsisten.

Pertimbangan

Jika Anda ingin Fargate menggunakan indeks SOCI untuk memuat gambar kontainer dengan malas dalam suatu tugas, pertimbangkan hal berikut:

- Hanya tugas yang berjalan pada versi platform Linux yang `1.4.0` dapat menggunakan indeks SOCI. Tugas yang menjalankan kontainer Windows di Fargate tidak didukung.
- Tugas yang berjalan pada X86_64 atau arsitektur ARM64 CPU didukung.
- Gambar kontainer dalam definisi tugas harus disimpan dalam registri gambar yang kompatibel. Berikut ini daftar registri yang kompatibel:
  - Pendaftar pribadi Amazon ECR.
- Hanya gambar kontainer yang menggunakan gzip kompresi, atau tidak dikompresi yang didukung. Gambar kontainer yang menggunakan zstd kompresi tidak didukung.
- Untuk SOCI Index Manifest v2, membuat manifes indeks SOCI memodifikasi manifes gambar kontainer saat kami menambahkan anotasi untuk indeks SOCI. Ini menghasilkan intisari gambar kontainer baru. Isi lapisan filesystem gambar kontainer tidak berubah.
- Untuk SOCI Index Manifest v2, ketika gambar kontainer telah disimpan di repositori gambar kontainer, setelah indeks SOCI dibuat, Anda perlu mendorong ulang gambar kontainer. Mendorong kembali gambar kontainer tidak akan meningkatkan biaya penyimpanan dengan menduplikasi lapisan sistem file, itu hanya mengunggah file manifes baru.
- Kami menyarankan Anda mencoba lazy loading dengan gambar kontainer yang lebih besar dari ukuran 250 MiB terkompresi. Anda cenderung tidak melihat pengurangan waktu untuk memuat gambar yang lebih kecil.

- Karena lazy loading dapat mengubah berapa lama tugas Anda dimulai, Anda mungkin perlu mengubah berbagai timeout seperti masa tenggang pemeriksaan kesehatan untuk Elastic Load Balancing.
- Jika Anda ingin mencegah gambar kontainer dimuat lambat, gambar kontainer perlu didorong ulang tanpa indeks SOCI terpasang.

Membuat indeks OCI Seekable

Agar gambar kontainer dimuat lambat, diperlukan indeks SOCI (file metadata) yang dibuat dan disimpan di repositori gambar kontainer di sepanjang sisi gambar kontainer. Untuk membuat dan mendorong indeks SOCI, Anda dapat menggunakan alat CLI soci-snapshotter open source. GitHub Atau, Anda dapat menerapkan CloudFormation AWS SOCI Index Builder. Ini adalah solusi tanpa server yang secara otomatis membuat dan mendorong indeks SOCI saat gambar kontainer didorong ke Amazon ECR. Untuk informasi selengkapnya tentang solusi dan langkah-langkah penginstalan, lihat Pembuat Indeks CloudFormation AWS SOCI di GitHub. CloudFormation AWS SOCI Index Builder adalah cara untuk mengotomatisasi memulai dengan SOCI, sementara alat soci open source memiliki lebih banyak fleksibilitas seputar pembuatan indeks dan kemampuan untuk mengintegrasikan pembuatan indeks dalam pipeline integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) Anda.

> **Note**
>
> Agar indeks SOCI dibuat untuk gambar, gambar harus ada di penyimpanan containerd gambar di komputer yang sedang berjalan`soci-snapshotter`. Jika gambar ada di toko Docker gambar, gambar tidak dapat ditemukan.

Memverifikasi bahwa tugas menggunakan pemuatan lambat

Untuk memverifikasi bahwa tugas dimuat dengan malas menggunakan SOCI, periksa titik akhir metadata tugas dari dalam tugas. Saat Anda menanyakan titik akhir metadata tugas versi 4, ada `Snapshotter` bidang di jalur default untuk wadah yang Anda kueri. Selain itu, ada `Snapshotter` bidang untuk setiap kontainer di `/task` jalur. Nilai default untuk bidang ini adalah`overlayfs`, dan bidang ini diatur ke `soci` jika SOCI digunakan. Untuk memverifikasi bahwa gambar kontainer memiliki SOCI Index Manifest v2 yang dilampirkan, Anda dapat mengambil Image Index dari Amazon ECR menggunakan. AWS CLI

```
IMAGE_REPOSITORY=r
IMAGE_TAG=latest

aws ecr batch-get-image \
    --repository-name=$IMAGE_REPOSITORY \
    --image-ids imageTag=$IMAGE_TAG \
    --query 'images[0].imageManifest' --output text | jq -r '.manifests[] |
 select(.artifactType=="application/vnd.amazon.soci.index.v2+json")'
```

Untuk memverifikasi bahwa image container memiliki SOCI Index Manifest v1 yang terpasang, Anda dapat menggunakan OCI Referrers API.

```
ACCOUNT_ID=111222333444
AWS_REGION=us-east-1
IMAGE_REPOSITORY=nginx-demo
IMAGE_TAG=latest
IMAGE_DIGEST=$(aws ecr describe-images --repository-name $IMAGE_REPOSITORY --image-ids
 imageTag=$IMAGE_TAG --query 'imageDetails[0].imageDigest' --output text)
ECR_PASSWORD=$(aws ecr get-login-password)

curl \
    --silent \
    --user AWS:$ECR_PASSWORD \
    https://$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/v2/$IMAGE_REPOSITORY/
referrers/$IMAGE_DIGEST?artifactType=application%2Fvnd.amazon.soci.index.v1%2Bjson | jq
 -r '.'
```

Perbedaan definisi tugas Amazon ECS untuk EC2 instance yang menjalankan Windows

Tugas yang berjalan pada instance EC2 Windows tidak mendukung semua parameter definisi tugas Amazon ECS yang tersedia. Beberapa parameter tidak didukung sama sekali, dan yang lain berperilaku berbeda.

Parameter definisi tugas berikut tidak didukung untuk definisi tugas Amazon EC2 Windows:

- `containerDefinitions`
  - `disableNetworking`
  - `dnsServers`

  - `dnsSearchDomains`
  - `extraHosts`
  - `links`
  - `linuxParameters`
  - `privileged`
  - `readonlyRootFilesystem`
  - `user`
  - `ulimits`
- `volumes`
  - `dockerVolumeConfiguration`
- `cpu`

  Kami merekomendasikan agar CPU tingkat-kontainer ditentukan untuk kontainer Windows.
- `memory`

  Kami merekomendasikan agar CPU tingkat-kontainer ditentukan untuk kontainer Windows.
- `proxyConfiguration`
- `ipcMode`
- `pidMode`
- `taskRoleArn`

  Peran IAM untuk tugas pada fitur instance EC2 Windows memerlukan konfigurasi tambahan, tetapi sebagian besar konfigurasi ini mirip dengan mengonfigurasi peran IAM untuk tugas pada instance container Linux. Untuk mengetahui informasi selengkapnya, lihat the section called “ Konfigurasi tambahan instans Amazon EC2 Windows”.

Membuat definisi tugas Amazon ECS menggunakan konsol

Anda membuat definisi tugas sehingga Anda dapat menentukan aplikasi yang Anda jalankan sebagai tugas atau layanan.

Saat Anda membuat definisi tugas untuk jenis peluncuran eksternal, Anda perlu membuat definisi tugas menggunakan editor JSON dan mengatur `requireCapabilities` parameternya. `EXTERNAL`

Anda dapat membuat definisi tugas dengan menggunakan pengalaman konsol, atau dengan menentukan file JSON.

Validasi JSON

Editor JSON konsol Amazon ECS memvalidasi hal berikut dalam file JSON:

- File tersebut adalah file JSON yang valid.
- File tidak berisi kunci asing.
- File berisi `familyName` parameter.
- Setidaknya ada satu entri di bawah`containerDefinitions`.

AWS CloudFormation tumpukan

Perilaku berikut berlaku untuk definisi tugas yang dibuat di konsol Amazon ECS baru sebelum 12 Januari 2023.

Saat Anda membuat definisi tugas, konsol Amazon ECS secara otomatis membuat CloudFormation tumpukan yang memiliki nama yang dimulai`ECS-Console-V2-TaskDefinition-`. Jika Anda menggunakan AWS CLI atau AWS SDK untuk membatalkan pendaftaran definisi tugas, Anda harus menghapus tumpukan definisi tugas secara manual. Untuk informasi selengkapnya, lihat Menghapus tumpukan di Panduan AWS CloudFormation Pengguna.

Definisi tugas yang dibuat setelah 12 Januari 2023, tidak memiliki CloudFormation tumpukan yang dibuat secara otomatis untuk mereka.

Prosedur

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pada menu Create new task definition, pilih Create new task definition.
4. Untuk Keluarga penetapan tugas, tentukan nama unik untuk penetapan tugas tersebut.
5. Untuk jenis Peluncuran, pilih lingkungan aplikasi. Default konsol adalah AWS Fargate(yang tanpa server). Amazon ECS menggunakan nilai ini untuk melakukan validasi guna memastikan bahwa parameter definisi tugas valid untuk jenis infrastruktur.

6. Untuk Sistem Operasi/Arsitektur, pilih sistem operasi dan arsitektur CPU untuk tugas tersebut.

   Untuk menjalankan tugas Anda pada arsitektur ARM 64-bit, pilih ARM64Linux/. Untuk informasi selengkapnya, lihat the section called “Platform runtime”.

   Untuk menjalankan AWS Fargatetugas Anda di wadah Windows, pilih sistem operasi Windows yang didukung. Untuk informasi selengkapnya, lihat the section called “Sistem Operasi dan Arsitektur”.

7. Untuk ukuran Tugas, pilih CPU dan nilai memori yang akan dicadangkan untuk tugas tersebut. Nilai CPU ditentukan sebagai v CPUs dan memori ditentukan sebagai GB.

   Untuk tugas yang dihosting di Fargate, tabel berikut menunjukkan kombinasi CPU dan memori yang valid.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
`1.4.0` atau yang lebih baru.		
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Untuk tugas yang menggunakan tipe peluncuran EC2 atau eksternal, nilai CPU tugas yang didukung adalah antara 128 unit CPU (0,125 vCPUs) dan unit CPU 196608 (192 v). CPUs

Untuk menentukan nilai memori dalam GB, masukkan GB setelah nilai. Misalnya, untuk mengatur nilai Memori ke 3GB, masukkan 3GB.

> **Note**
>
> Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer.

8. Untuk mode Jaringan, pilih mode jaringan yang akan digunakan. Defaultnya adalah mode awsvpc. Untuk informasi selengkapnya, lihat jaringan tugas Amazon ECS.

   Jika Anda memilih jembatan, di bawah pemetaan Port, untuk port Host, masukkan nomor port pada instance kontainer untuk memesan kontainer Anda.

9. (Opsional) Perluas bagian Peran tugas untuk mengonfigurasi peran AWS Identity and Access Management (IAM) untuk tugas:

   a. Untuk peran Tugas, pilih peran IAM yang akan ditetapkan ke tugas. Peran IAM tugas memberikan izin untuk kontainer dalam tugas untuk memanggil operasi AWS API.

b. Untuk peran eksekusi tugas, pilih peran.

   Untuk informasi tentang kapan menggunakan peran eksekusi tugas, lihatthe section called “Eksekusi tugas peran IAM”. Jika Anda tidak membutuhkan peran tersebut, pilih Tidak Ada.

10. (Opsional) Perluas bagian Penempatan tugas untuk menambahkan batasan penempatan. Batasan penempatan tugas memungkinkan Anda memfilter instance kontainer yang digunakan untuk penempatan tugas menggunakan atribut bawaan atau kustom.
11. (Opsional) Perluas bagian injeksi Kesalahan untuk mengaktifkan injeksi kesalahan. Injeksi kesalahan memungkinkan Anda menguji bagaimana aplikasi Anda merespons skenario gangguan tertentu.
12. Untuk setiap kontainer untuk menentukan definisi tugas Anda, selesaikan langkah-langkah berikut.

    a. Untuk Nama, masukkan nama untuk wadah.

    b. Untuk URI Gambar, masukkan gambar yang akan digunakan untuk memulai wadah. Gambar dalam registri Galeri Publik Amazon ECR dapat ditentukan dengan menggunakan nama registri Publik Amazon ECR saja. Misalnya, jika `public.ecr.aws/ecs/amazon-ecs-agent:latest` ditentukan, wadah Amazon Linux yang dihosting di Galeri Publik Amazon ECR digunakan. Untuk semua repositori lainnya, tentukan repositori dengan menggunakan format atau. `repository-url/image:tag` `repository-url/image@digest`

    c. Jika gambar Anda berada di registri pribadi di luar Amazon ECR, di bawah Registri pribadi, aktifkan otentikasi registri pribadi. Kemudian, di Secrets Manager ARN atau nama, masukkan Amazon Resource Name (ARN) dari rahasia tersebut.

    d. Untuk wadah Essential, jika definisi tugas Anda memiliki dua atau lebih kontainer yang ditentukan, Anda dapat menentukan apakah penampung harus dianggap penting. Ketika sebuah wadah ditandai sebagai Essential, jika kontainer itu berhenti, maka tugas dihentikan. Setiap definisi tugas harus berisi setidaknya satu wadah penting.

    e. Pemetaan port memungkinkan kontainer untuk mengakses port pada host untuk mengirim atau menerima lalu lintas. Di bawah pemetaan Port, lakukan salah satu hal berikut:

       - Saat Anda menggunakan mode jaringan awsvpc, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.

- Saat Anda menggunakan mode jaringan jembatan, untuk port Container dan Protocol, pilih pemetaan port yang akan digunakan untuk wadah.

  Pilih Tambahkan lebih banyak pemetaan port untuk menentukan pemetaan port kontainer tambahan.

f. Untuk memberikan akses read-only container ke sistem file root, untuk Read only root file system, pilih Read only.

g. (Opsional) Untuk menentukan batas CPU, GPU, dan memori tingkat kontainer yang berbeda dari nilai tingkat tugas, di bawah batas alokasi sumber daya, lakukan hal berikut:

- Untuk CPU, masukkan jumlah unit CPU yang dicadangkan agen kontainer Amazon ECS untuk kontainer.
- Untuk GPU, masukkan jumlah unit GPU untuk instance container.

  EC2 Instans Amazon dengan dukungan GPU memiliki 1 unit GPU untuk setiap GPU. Untuk informasi selengkapnya, lihat the section called “Definisi tugas untuk beban kerja GPU”.
- Untuk batas keras Memori, masukkan jumlah memori, dalam GB, untuk disajikan ke wadah. Jika wadah mencoba melampaui batas keras, wadah berhenti.
- Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 mebibytes (MiB) memori untuk wadah, jadi jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

  Daemon Docker 19.03.13-ce atau sebelumnya menyimpan minimal 4 MiB memori untuk wadah, jadi jangan tentukan kurang dari 4 MiB memori untuk kontainer Anda.
- Untuk batas lunak Memori, masukkan batas lunak (dalam GB) memori untuk cadangan wadah.

  Ketika memori sistem sedang diperdebatkan, Docker upaya untuk menjaga memori kontainer ke batas lunak ini. Jika Anda tidak menentukan memori tingkat tugas, Anda harus menentukan bilangan bulat bukan nol untuk salah satu atau kedua batas keras Memori dan batas lunak Memori. Jika Anda menentukan keduanya, batas keras Memori harus lebih besar dari batas lunak Memori.

  Fitur ini tidak didukung pada wadah Windows.

h. (Opsional) Perluas bagian variabel Lingkungan untuk menentukan variabel lingkungan untuk disuntikkan ke dalam wadah. Anda dapat menentukan variabel lingkungan baik secara individual dengan menggunakan pasangan nilai kunci atau secara massal dengan menentukan file variabel lingkungan yang dihosting di bucket Amazon S3. Untuk informasi tentang cara memformat file variabel lingkungan, lihatMeneruskan variabel lingkungan individual ke wadah Amazon ECS.

   Saat Anda menentukan variabel lingkungan untuk penyimpanan rahasia, untuk Key, masukkan nama rahasia. Kemudian untuk ValueFrom, masukkan ARN lengkap dari rahasia Systems Manager Parameter Store atau rahasia Secrets Manager

i. (Opsional) Pilih opsi Gunakan koleksi log untuk menentukan konfigurasi log. Untuk setiap driver log yang tersedia, ada opsi driver log untuk ditentukan. Opsi default mengirimkan log kontainer ke Amazon CloudWatch Logs. Opsi driver log lainnya dikonfigurasi dengan menggunakan AWS FireLens. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

   Berikut ini menjelaskan setiap tujuan log kontainer secara lebih rinci.

   - Amazon CloudWatch — Konfigurasikan tugas untuk mengirim log kontainer ke CloudWatch Log. Opsi driver log default disediakan, yang membuat grup CloudWatch log atas nama Anda. Untuk menentukan nama grup log yang berbeda, ubah nilai opsi driver.
   - Ekspor log ke Splunk — Konfigurasikan tugas untuk mengirim log kontainer ke Splunk driver yang mengirim log ke layanan jarak jauh. Anda harus memasukkan URL ke layanan Splunk web Anda. SplunkToken ditetapkan sebagai opsi rahasia karena dapat diperlakukan sebagai data sensitif.
   - Ekspor log ke Amazon Data Firehose — Konfigurasikan tugas untuk mengirim log kontainer ke Firehose. Opsi driver log default disediakan, yang mengirimkan log ke aliran pengiriman Firehose. Untuk menentukan nama aliran pengiriman yang berbeda, ubah nilai opsi driver.
   - Ekspor log ke Amazon Kinesis Data Streams — Konfigurasikan tugas untuk mengirim log kontainer ke Kinesis Data Streams. Opsi driver log default disediakan, yang mengirim log ke aliran Kinesis Data Streams. Untuk menentukan nama aliran yang berbeda, ubah nilai opsi driver.
   - Ekspor log ke Amazon OpenSearch Service — Konfigurasikan tugas untuk mengirim log kontainer ke domain OpenSearch Layanan. Opsi driver log harus disediakan.

- Ekspor log ke Amazon S3 — Konfigurasikan tugas untuk mengirim log kontainer ke bucket Amazon S3. Opsi driver log default disediakan, tetapi Anda harus menentukan nama bucket Amazon S3 yang valid.

j. (Opsional) Konfigurasikan parameter wadah tambahan.

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Mulai ulang kebijakan Opsi ini menentukan kebijakan restart untuk memulai ulang kontainer saat keluar.	Perluas kebijakan Restart, lalu konfigurasikan item berikut: • Untuk mengaktifkan kebijakan restart kontainer, aktifkan Aktifkan kebijakan Restart. • Untuk kode keluar yang diabaikan, tentukan daftar kode keluar kontainer integer yang dipisahkan koma. Jika kontainer keluar dengan salah satu kode keluar yang ditentuka n, Amazon ECS tidak akan mencoba memulai ulang penampung. Jika tidak ada yang ditentuka n, Amazon ECS tidak akan mengabaikan kode keluar apa pun. • Untuk periode Reset Percobaan, tentukan periode waktu integer, dalam hitungan detik, wadah harus dijalankan sebelum	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	restart dapat dicoba jika terjadi keluar. Amazon ECS dapat mencoba memulai ulang penampung hanya sekali setiap detik periode Percobaan reset. Jika tidak ada yang ditentukan, penampung harus berjalan selama 300 detik sebelum restart dapat dicoba.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
HealthCheck Ini adalah perintah yang menentukan apakah sebuah wadah sehat. Untuk informasi selengkap nya, lihat Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer.	Perluas HealthCheck, lalu konfigurasikan item berikut: • Untuk Command, masukkan daftar perintah yang dipisahka n koma. Anda dapat memulai perintah `CMD` dengan menjalank an argumen perintah secara langsung, atau `CMD-SHELL` menjalank an perintah dengan shell default container. Jika tidak ada yang `CMD` ditentukan, digunakan. • Untuk Interval, masukkan jumlah detik antara setiap pemeriksa an kesehatan. Nilai yang valid adalah antara 5 dan 30. • Untuk Timeout, masukkan periode waktu (dalam detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal. Nilai	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	yang valid adalah antara 2 dan 60. • Untuk periode Mulai, masukkan periode waktu (dalam detik) untuk menunggu wadah untuk bootstrap sebelum perintah pemeriksaan kesehatan berjalan. Nilai yang valid adalah antara 0 dan 300. • Untuk Mencoba lagi, masukkan berapa kali untuk mencoba kembali perintah pemeriksaan kesehatan ketika ada kegagalan. Nilai yang valid adalah antara 1 dan 10.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Pemesanan ketergantungan startup Opsi ini mendefinisikan dependensi untuk startup dan shutdown container . Sebuah kontainer dapat berisi beberapa dependensi.	Perluas urutan ketergantungan Startup, lalu konfigurasikan yang berikut ini: a. Pilih Tambahkan ketergantungan kontainer. b. Untuk Kontainer, pilih wadahnya. c. Untuk Kondisi, pilih kondisi ketergantungan startup. Untuk menambahkan dependensi tambahan, pilih Tambahkan ketergantungan kontainer.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Batas waktu kontainer Opsi ini menentukan kapan harus memulai dan menghentikan wadah.	Perluas batas waktu Kontainer, lalu konfigura sikan yang berikut ini: • Untuk mengonfigurasi waktu menunggu sebelum menyerah pada penyelesaian dependensi untuk wadah, untuk Batas waktu mulai, masukkan jumlah detik. • Untuk mengonfigurasi waktu menunggu sebelum penampung dihentikan jika tidak keluar secara normal dengan sendirinya, untuk Stop timeout, masukkan jumlah detik.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Pengaturan jaringan kontainer Opsi ini menentukan apakah akan menggunakan jaringan dalam wadah.	Perluas pengaturan jaringan Container, lalu konfigurasikan yang berikut ini: • Untuk menonaktifkan jaringan kontainer, pilih Matikan jaringan. • Untuk mengkonfigurasi alamat IP server DNS yang disajikan ke wadah, di server DNS, masukkan alamat IP setiap server pada baris terpisah. • Untuk mengonfigurasi domain DNS untuk mencari nama non-fully-qualified host yang disajikan ke wadah, di domain pencarian DNS, masukkan setiap domain pada baris terpisah. Polanya adalah **^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$** . • Untuk mengonfigurasi nama host kontainer, dalam nama Host,	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	masukkan nama kambing kontainer. • Untuk menambahkan nama host dan pemetaan alamat IP yang ditambahkan ke `/etc/hosts` file pada wadah, pilih Tambahkan host tambahan, lalu untuk Hostname dan alamat IP, masukkan nama host dan alamat IP.	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Dockerkonfigurasi Ini mengesampingkan nilai-nilai di. Dockerfile	Perluas Dockerkon figurasi, lalu konfigura sikan item berikut: • Untuk Command, masukkan perintah yang dapat dieksekusi untuk sebuah wadah. Parameter ini dipetakan ke `Cmd` bagian Create a container dari Docker Remote API dan `COMMAND` opsi untuk`docker run`. Parameter ini mengesampingkan `CMD` instruksi dalam file. Dockerfile • Untuk titik Masuk, masukkan Docker ENTRYPOINT yang diteruskan ke wadah. Parameter ini dipetakan ke `Entrypoint` bagian Create a container dari Docker Remote API dan `--entrypoint` opsi untuk`docker run`. Parameter ini	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
	mengesampingkan `ENTRYPOINT` instruksi dalam file. Dockerfile • Untuk direktori Kerja, masukkan direktori bahwa wadah akan menjalankan titik masuk dan instruksi perintah yang disediakan. Parameter ini dipetakan ke `WorkingDir` bagian Create a container dari Docker Remote API dan `--workdir` opsi untuk`docker run`. Parameter ini mengesampingkan `WORKDIR` instruksi dalam file. Dockerfile	

Untuk mengkonfigurasi opsi ini	Lakukan hal berikut	
Batas sumber daya (Ulimits) Nilai-nilai ini menimpa pengaturan kuota sumber daya default untuk sistem operasi. Parameter ini memetakan ke `Ulimits` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--ulimit` untuk docker run.	Perluas batas Sumber daya (ulimits), lalu pilih Tambah ulimit. Untuk nama Limit, pilih limit. Kemudian, untuk Soft limit dan Hard limit, masukkan nilainya. Untuk menambahkan tambahanulimits, pilih Tambah ulimit.	
Dockerlabel Opsi ini menambahkan metadata ke wadah Anda. Parameter ini memetakan ke `Labels` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--label` untuk docker run.	Perluas Dockerlabel, pilih Tambahkan pasangan nilai kunci, lalu masukkan Kunci dan Nilai. Untuk menambahkan Docker label tambahan, pilih Tambah pasangan nilai kunci.	

k. (Opsional) Pilih Tambahkan lebih banyak wadah untuk menambahkan wadah tambahan ke definisi tugas.

13. (Opsional) Bagian Penyimpanan digunakan untuk memperluas jumlah penyimpanan sementara untuk tugas yang dihosting di Fargate. Anda juga dapat menggunakan bagian ini untuk menambahkan konfigurasi volume data untuk tugas tersebut.

    - Untuk memperluas penyimpanan sementara yang tersedia di luar nilai default 20 gibibytes (GiB) untuk tugas Fargate Anda, untuk Jumlah, masukkan nilai hingga. 200 GiB

14. (Opsional) Untuk menambahkan konfigurasi volume data untuk definisi tugas, pilih Tambahkan volume, lalu ikuti langkah-langkah ini.

    a. Untuk nama Volume, masukkan nama untuk volume data. Nama volume data digunakan saat membuat titik pemasangan kontainer.

    b. Untuk konfigurasi Volume, pilih apakah Anda ingin mengonfigurasi volume saat membuat definisi tugas atau selama penerapan.

    > **Note**
    >
    > Volume yang dapat dikonfigurasi saat membuat definisi tugas termasuk Bind mountDocker, Amazon EFS, dan Amazon FSx untuk Windows File Server. Volume yang dapat dikonfigurasi saat penerapan saat menjalankan tugas, atau saat membuat atau memperbarui layanan menyertakan Amazon EBS.

    c. Untuk tipe Volume, pilih tipe volume yang kompatibel dengan tipe konfigurasi yang Anda pilih, lalu konfigurasikan jenis volume.

Tipe volume	Langkah-langkah	
Bind mount	a. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. b. Untuk menambahkan titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
EFS	a. Untuk ID sistem File, pilih ID sistem file Amazon EFS. b. (Opsional) Untuk Root directory (Direktor i asal), masukkan direktori dalam sistem file Amazon EFS untuk pemasangan sebagai direktori asal di dalam host. Jika parameter ini diabaikan, asal volume Amazon EFS akan digunakan. Jika Anda berencana untuk menggunakan titik akses EFS, biarkan bidang ini kosong. c. (Opsional) Untuk titik Akses, pilih ID titik akses yang akan digunakan. d. (Opsional) Untuk mengenkripsi data antara sistem file Amazon EFS dan host Amazon ECS atau untuk menggunak an peran eksekusi tugas saat memasang volume, pilih Konfigurasi lanjutan,	

Tipe volume	Langkah-langkah	
	lalu konfigurasikan yang berikut ini: • Untuk mengenkripsi data antara sistem file Amazon EFS dan host Amazon ECS, pilih Enkripsi Transit, lalu untuk Port, masukkan port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, strategi pemilihan port yang digunakan oleh pembantu pemasangan Amazon EFS akan digunakan. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS di Panduan Pengguna Amazon Elastic File System. • Untuk menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat	

Tipe volume	Langkah-langkah	
	memasang sistem file Amazon EFS, pilih otorisasi IAM. e. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. f. Untuk menambahk an titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
Docker	a. Untuk Driver, masukkan konfigurasi Docker volume. Wadah Windows hanya mendukung penggunaan driver lokal. Untuk menggunakan bind mount, tentukan host. b. Untuk Lingkup, pilih siklus hidup volume. • Agar siklus hidup bertahan saat tugas dimulai dan berhenti, pilih Tugas. • Agar volume tetap ada setelah tugas berhenti, pilih Dibagikan. c. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah.	

Tipe volume	Langkah-langkah	
	• Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. d. Untuk menambahkan titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
FSx untuk Windows File Server	a. Untuk ID sistem File, pilih ID sistem file Windows File Server FSx untuk Windows. b. Untuk direktori Root, masukkan direktori, masukkan direktori di dalam sistem file FSx untuk Windows File Server untuk dipasang sebagai direktori root di dalam host. c. Untuk parameter Credential, pilih bagaimana kredensyal disimpan. • Untuk menggunakan AWS Secrets Manager, masukkan Nama Sumber Daya Amazon (ARN) dari rahasia Secrets Manager. • Untuk menggunakan AWS Systems Manager, masukkan Amazon Resource Name (ARN) dari parameter Systems Manager.	

Tipe volume	Langkah-langkah	
	d. Untuk Domain, masukkan nama domain yang memenuhi syarat sepenuhnya yang dihosting oleh direktori AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) atau Direktori EC2 Aktif yang dihosting sendiri. e. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume.	

Tipe volume	Langkah-langkah	
	f. Untuk menambahkan titik pemasangan tambahan, Tambahkan titik pemasangan.	

Tipe volume	Langkah-langkah	
Amazon EBS	a. Pilih Tambahkan titik pemasangan, lalu konfigurasikan yang berikut ini: • Untuk Container, pilih wadah untuk titik pemasangan. • Untuk volume Sumber, pilih volume data yang akan dipasang ke wadah. • Untuk jalur Kontainer , masukkan jalur pada wadah untuk memasang volume. • Untuk Read only, pilih apakah container memiliki akses read-only ke volume. b. Untuk menambahk an titik pemasangan tambahan, Tambahkan titik pemasangan.	

15. Untuk menambahkan volume dari wadah lain, pilih Tambahkan volume dari, lalu konfigurasikan yang berikut ini:

    • Untuk Kontainer, pilih wadahnya.

- Untuk Sumber, pilih wadah yang memiliki volume yang ingin Anda pasang.
- Untuk Read only, pilih apakah container memiliki akses read-only ke volume.

16. (Opsional) Untuk mengonfigurasi pengaturan penelusuran aplikasi dan pengumpulan metrik dengan menggunakan AWS Distro for OpenTelemetry integrasi, perluas Pemantauan, lalu pilih Gunakan koleksi metrik untuk mengumpulkan dan mengirim metrik tugas Anda ke Amazon CloudWatch atau Amazon Managed Service for Prometheus. Saat opsi ini dipilih, Amazon ECS membuat sespan AWS Distro for OpenTelemetry kontainer yang telah dikonfigurasi sebelumnya untuk mengirim metrik aplikasi. Untuk informasi selengkapnya, lihat Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi.

    a. Saat Amazon CloudWatch dipilih, metrik aplikasi kustom Anda dirutekan ke CloudWatch metrik khusus. Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon CloudWatch.

    > **Important**
    >
    > Saat mengekspor metrik aplikasi ke Amazon CloudWatch, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk AWS Distro untuk OpenTelemetry integrasi dengan Amazon CloudWatch.

    b. Saat Anda memilih Layanan Terkelola Amazon untuk Prometheus (instrumentasi pustaka Prometheus), CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda dirutekan ke Amazon Managed Service untuk Prometheus. Untuk titik akhir penulisan jarak jauh Workspace, masukkan URL titik akhir penulisan jarak jauh untuk ruang kerja Anda. Prometheus Untuk target Scraping, masukkan host dan port yang dapat digunakan AWS Distro for OpenTelemetry kolektor untuk mengikis data metrik. Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus.

    > **Important**
    >
    > Saat mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk

AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus.

c. Saat Anda memilih Amazon Managed Service for Prometheus OpenTelemetry (instrumentasi), CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda dirutekan ke Amazon Managed Service for Prometheus. Untuk titik akhir penulisan jarak jauh Workspace, masukkan URL titik akhir penulisan jarak jauh untuk ruang kerja Anda. Prometheus Untuk informasi selengkapnya, lihat Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus.

> **Important**
>
> Saat mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus, definisi tugas Anda memerlukan peran IAM tugas dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin IAM yang diperlukan untuk AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus.

17. (Opsional) Perluas bagian Tag untuk menambahkan tag, sebagai pasangan nilai kunci, ke definisi tugas.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

18. Pilih Buat untuk mendaftarkan definisi tugas.

Amazon ECS console JSON editor

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pada menu Create new task definition, pilih Create new task definition with JSON.
4. Di kotak editor JSON, edit file JSON Anda,

   JSON harus lulus pemeriksaan validasi yang ditentukan dalam. the section called “Validasi JSON”

5. Pilih Buat.

Memperbarui definisi tugas Amazon ECS menggunakan konsol

Revisi definisi tugas adalah salinan dari definisi tugas saat ini dengan nilai parameter baru menggantikan yang sudah ada. Semua parameter yang tidak Anda modifikasi ada dalam revisi baru.

Untuk memperbarui ketentuan tugas, membuat revisi ketentuan tugas. Jika ketentuan tugas yang digunakan dalam layanan, Anda harus memperbarui layanan tersebut untuk menggunakan ketentuan tugas yang diperbarui.

Saat Anda membuat revisi, Anda dapat memodifikasi properti kontainer berikut dan properti lingkungan.

- URI gambar kontainer
- Pemetaan pelabuhan
- Variabel-variabel lingkungan
- Ukuran tugas
- Ukuran kontainer
- role tugas
- Peran pelaksanaan tugas
- Volume dan titik pemasangan kontainer
- Registri pribadi

Validasi JSON

Editor JSON konsol Amazon ECS memvalidasi hal berikut dalam file JSON:

- File tersebut adalah file JSON yang valid
- File tidak berisi kunci asing
- File berisi `familyName` parameter
- Setidaknya ada satu entri di bawah `containerDefinitions`

Prosedur

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Definisi tugas.
4. Pilih definisi tugas.
5. Pilih revisi definisi tugas, lalu pilih Buat revisi baru, Buat revisi baru.
6. Pada halaman Buat revisi definisi tugas baru, buat perubahan. Misalnya, untuk mengubah definisi kontainer yang ada (seperti gambar kontainer, batas memori, atau pemetaan port), pilih wadah, lalu buat perubahan.
7. Verifikasi informasi, lalu pilih Perbarui.
8. Jika ketentuan tugas Anda digunakan dalam layanan, perbarui layanan Anda dengan ketentuan tugas yang diperbarui. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Amazon ECS console JSON editor

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat revisi baru, Buat revisi baru dengan JSON.
4. Di kotak editor JSON, edit file JSON Anda,

   JSON harus lulus pemeriksaan validasi yang ditentukan dalam. the section called “Validasi JSON”
5. Pilih Buat.

Membatalkan pendaftaran revisi definisi tugas Amazon ECS menggunakan konsol

Anda dapat membatalkan pendaftaran revisi definisi tugas sehingga tidak lagi ditampilkan dalam panggilan `ListTaskDefinition` API atau di konsol saat Anda ingin menjalankan tugas atau memperbarui layanan.

Ketika Anda membatalkan pendaftaran revisi ketentuan tugas, segera ditandai sebagai `INACTIVE`. Tugas dan layanan yang ada yang mereferensikan tugas revisi definisi `INACTIVE` terus berjalan tanpa gangguan. Layanan yang ada yang mereferensikan revisi ketentuan tugas `INACTIVE` masih dapat menaikkan atau menurunkan skala dengan mengubah jumlah layanan yang diinginkan.

Anda tidak dapat menggunakan revisi ketentuan tugas `INACTIVE` untuk menjalankan tugas baru atau membuat layanan baru. Anda juga tidak dapat memperbarui layanan yang sudah ada untuk mereferensikan revisi ketentuan tugas `INACTIVE` (meskipun mungkin ada jendela hingga 10 menit setelah pencabutan pendaftaran tempat pembatasan ini belum berlaku).

> **Note**
>
> Saat Anda membatalkan pendaftaran semua revisi dalam keluarga tugas, keluarga definisi tugas dipindahkan ke daftar. `INACTIVE` Menambahkan revisi baru definisi `INACTIVE` tugas memindahkan keluarga definisi tugas kembali ke `ACTIVE` daftar.
> Pada saat ini, revisi ketentuan tugas `INACTIVE` tetap dapat ditemukan di akun Anda tanpa batas waktu. Namun, perilaku ini dapat berubah di masa mendatang. Oleh karena itu, Anda tidak boleh mengandalkan revisi ketentuan tugas `INACTIVE` yang bertahan di luar siklus hidup tugas dan layanan terkait apa pun.

AWS CloudFormation tumpukan

Perilaku berikut berlaku untuk definisi tugas yang dibuat di konsol Amazon ECS baru sebelum 12 Januari 2023.

Saat Anda membuat definisi tugas, konsol Amazon ECS secara otomatis membuat CloudFormation tumpukan yang memiliki nama yang dimulai`ECS-Console-V2-TaskDefinition-`. Jika Anda menggunakan AWS CLI atau AWS SDK untuk membatalkan pendaftaran definisi tugas, Anda harus menghapus tumpukan definisi tugas secara manual. Untuk informasi selengkapnya, lihat Menghapus tumpukan di Panduan AWS CloudFormation Pengguna.

Definisi tugas yang dibuat setelah 12 Januari 2023, tidak memiliki CloudFormation tumpukan yang dibuat secara otomatis untuk mereka.

Prosedur

Untuk membatalkan pendaftaran definisi tugas baru (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Definisi tugas.
4. Pada halaman Definisi tugas, pilih keluarga definisi tugas yang berisi satu atau beberapa revisi yang ingin Anda deregister.
5. Pada halaman Nama definisi tugas, pilih revisi yang akan dihapus, lalu pilih Tindakan, Deregister.
6. Verifikasi informasi di jendela Deregister, lalu pilih Deregister untuk menyelesaikan.

Menghapus revisi definisi tugas Amazon ECS menggunakan konsol

Saat tidak lagi memerlukan revisi definisi tugas tertentu di Amazon ECS, Anda dapat menghapus revisi definisi tugas.

Saat Anda menghapus revisi definisi tugas, itu segera bertransisi dari ke. `INACTIVE` `DELETE_IN_PROGRESS` Tugas dan layanan yang ada yang mereferensikan revisi definisi `DELETE_IN_PROGRESS` tugas terus berjalan tanpa gangguan.

Anda tidak dapat menggunakan revisi definisi `DELETE_IN_PROGRESS` tugas untuk menjalankan tugas baru atau membuat layanan baru. Anda juga tidak dapat memperbarui layanan yang ada untuk mereferensikan revisi definisi `DELETE_IN_PROGRESS` tugas.

Saat Anda menghapus semua revisi definisi `INACTIVE` tugas, nama definisi tugas tidak ditampilkan di konsol dan tidak ditampilkan di API. Jika revisi definisi tugas dalam `DELETE_IN_PROGRESS` status, nama definisi tugas ditampilkan di konsol dan dikembalikan di API. Nama definisi tugas dipertahankan oleh Amazon ECS dan revisi bertambah saat berikutnya Anda membuat definisi tugas dengan nama itu.

Sumber daya Amazon ECS yang dapat memblokir penghapusan

Permintaan penghapusan definisi tugas tidak akan selesai ketika ada sumber daya Amazon ECS yang bergantung pada revisi definisi tugas. Sumber daya berikut mungkin mencegah definisi tugas dihapus:

- Tugas mandiri Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Tugas layanan Amazon ECS - Definisi tugas diperlukan agar tugas tetap sehat.
- Penerapan layanan Amazon ECS dan set tugas - Definisi tugas diperlukan saat peristiwa penskalaan dimulai untuk penyebaran atau set tugas Amazon ECS.

Jika definisi tugas Anda tetap dalam `DELETE_IN_PROGRESS` status, Anda dapat menggunakan konsol, atau AWS CLI untuk mengidentifikasi, dan kemudian menghentikan sumber daya yang memblokir penghapusan definisi tugas.

Penghapusan definisi tugas setelah sumber daya yang diblokir dihapus

Aturan berikut berlaku setelah Anda menghapus sumber daya yang memblokir penghapusan definisi tugas:

- Tugas Amazon ECS - Penghapusan definisi tugas dapat memakan waktu hingga 1 jam untuk diselesaikan setelah tugas dihentikan.
- Penyebaran layanan Amazon ECS dan set tugas - Penghapusan definisi tugas dapat memakan waktu hingga 24 jam untuk diselesaikan setelah penerapan atau set tugas dihapus.

Prosedur

Untuk menghapus definisi tugas (konsol Amazon ECS)

Anda harus membatalkan pendaftaran revisi definisi tugas sebelum Anda menghapusnya. Untuk informasi selengkapnya, lihat the section called “Membatalkan pendaftaran revisi definisi tugas menggunakan konsol”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Definisi tugas.

4. Pada halaman Definisi tugas, pilih keluarga definisi tugas yang berisi satu atau beberapa revisi yang ingin Anda hapus.
5. Pada halaman nama definisi tugas, pilih revisi yang akan dihapus, lalu pilih Tindakan, Hapus.

   Jika Hapus tidak tersedia, Anda harus membatalkan pendaftaran definisi tugas.
6. Verifikasi informasi di kotak Hapus konfirmasi, lalu pilih Hapus untuk menyelesaikan.

Kasus penggunaan definisi tugas Amazon ECS

Pelajari lebih lanjut tentang cara menulis definisi tugas untuk berbagai AWS layanan dan fitur.

Bergantung pada beban kerja Anda, ada parameter definisi tugas tertentu yang perlu diatur. Juga untuk jenis EC2 peluncuran, Anda harus memilih instance tertentu yang direkayasa untuk beban kerja.

Topik

- Definisi tugas Amazon ECS untuk beban kerja GPU
- Definisi tugas Amazon ECS untuk beban kerja transcoding video
- Definisi tugas Amazon ECS untuk beban kerja pembelajaran mesin AWS Neuron
- Definisi tugas Amazon ECS untuk instans pembelajaran mendalam
- Definisi tugas Amazon ECS untuk beban kerja ARM 64-bit
- Kirim log Amazon ECS ke CloudWatch
- Kirim log Amazon ECS ke AWS layanan atau AWS Partner
- Menggunakan gambar AWS non-kontainer di Amazon ECS
- Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer
- Meneruskan data sensitif ke wadah Amazon ECS

Definisi tugas Amazon ECS untuk beban kerja GPU

Amazon ECS mendukung beban kerja yang digunakan GPUs, saat Anda membuat cluster dengan instance container yang mendukung. GPUs Instans container EC2 berbasis GPU Amazon yang menggunakan tipe instans p2, p3, p5, g3, g4, dan g5 menyediakan akses ke NVIDIA. GPUs Untuk informasi selengkapnya, lihat Instans Komputasi Akselerasi Linux di panduan Jenis EC2 Instans Amazon.

Amazon ECS menyediakan AMI yang dioptimalkan untuk GPU yang dilengkapi dengan driver kernel NVIDIA yang telah dikonfigurasi sebelumnya dan runtime GPU Docker. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

Anda dapat menunjuk sejumlah definisi GPUs tugas Anda untuk pertimbangan penempatan tugas di tingkat kontainer. Amazon ECS menjadwalkan instans kontainer yang tersedia yang mendukung GPUs dan menyematkan fisik GPUs ke wadah yang tepat untuk kinerja optimal.

Jenis instans EC2 berbasis GPU Amazon berikut didukung. Untuk informasi selengkapnya, lihat Instans Amazon EC2 P2, InstansAmazon EC2 P3, Instans Amazon P4d, Instans EC2 AmazonP5, Instans EC2 Amazon G3, Instans Amazon G4, Instans EC2 Amazon G5 EC2 , Instans Amazon G6, dan Instans EC2 Amazon G6e. EC2 EC2

Jenis instans	GPUs	Memori GPU (GiB)	v CPUs	Memori (GiB)
p3.2xlarge	1	16	8	61
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048
g3s.xlarge	1	8	4	30,5
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488
g4dn.xlarge	1	16	4	16
g4dn.2xlarge	1	16	8	32
g4dn.4xlarge	1	16	16	64

Jenis instans	GPUs	Memori GPU (GiB)	v CPUs	Memori (GiB)
g4dn.8xlarge	1	16	32	128
g4dn.12xlarge	4	64	48	192
g4dn.16xlarge	1	16	64	256
g5.xlarge	1	24	4	16
g5.2xlarge	1	24	8	32
g5.4xlarge	1	24	16	64
g5.8xlarge	1	24	32	128
g5.16xlarge	1	24	64	256
g5.12xlarge	4	96	48	192
g5.24xlarge	4	96	96	384
g5.48xlarge	8	192	192	768
g6.xlarge	1	24	4	16
g6.2xlarge	1	24	8	32
g6.4xlarge	1	24	16	64
g6.8xlarge	1	24	32	128
g6.16.xlarge	1	24	64	256
g6.12xlarge	4	96	48	192
g6.24xlarge	4	96	96	384
g6.48xlarge	8	192	192	768
g6.logam	8	192	192	768

Jenis instans	GPUs	Memori GPU (GiB)	v CPUs	Memori (GiB)
gr6.4xbesar	1	24	16	128
g6e.xlarge	1	48	4	32
g6e.2xlarge	1	48	8	64
g6e.4xlarge	1	48	16	128
g6e.8xlarge	1	48	32	256
g6e16.xlarge	1	48	64	512
g6e12.xlarge	4	192	48	384
g6e24.xlarge	4	192	96	768
g6e48.xlarge	8	384	192	1536
gr6.8xbesar	1	24	32	256

Anda dapat mengambil ID Amazon Machine Image (AMI) untuk Amazon ECS yang dioptimalkan AMIs dengan menanyakan Parameter Store API. AWS Systems Manager Dengan menggunakan parameter ini, Anda tidak perlu mencari AMI yang dioptimalkan Amazon ECS secara manual. IDs Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParameter. Pengguna yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon ECS yang dioptimalkan.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

Pertimbangan

> **Note**
>
> Dukungan untuk tipe keluarga instans g2 telah usang.

Jenis keluarga instans p2 hanya didukung pada versi yang lebih awal dari AMI yang `20230912` dioptimalkan untuk GPU Amazon ECS. Jika Anda perlu terus menggunakan instance p2, lihat. Apa yang harus dilakukan jika Anda membutuhkan instance P2 Pembaruan NVIDIA/CUDA driver di tempat pada kedua tipe keluarga instans ini akan menyebabkan potensi kegagalan beban kerja GPU.

Kami menyarankan Anda mempertimbangkan hal berikut sebelum Anda mulai bekerja dengan GPUs Amazon ECS.

- Klaster Anda dapat terdiri dari campuran GPU dan instans kontainer non-GPU.
- Anda dapat menjalankan beban kerja GPU pada instance eksternal. Saat mendaftarkan instance eksternal dengan cluster Anda, pastikan `--enable-gpu` flag disertakan pada skrip instalasi. Untuk informasi selengkapnya, lihat Mendaftarkan instans eksternal ke kluster Amazon ECS.
- Anda harus mengatur `ECS_ENABLE_GPU_SUPPORT` ke `true` dalam file konfigurasi agen Anda. Untuk informasi selengkapnya, lihat the section called “Konfigurasi agen kontainer”.
- Saat menjalankan tugas atau membuat layanan, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas untuk menentukan instance kontainer tempat tugas akan diluncurkan. Dengan melakukan hal tersebut, Anda bisa lebih efektif menggunakan sumber daya Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Contoh berikut meluncurkan tugas pada instans kontainer `g4dn.xlarge` di klaster default Anda.

  ```
  aws ecs run-task --cluster default --task-definition ecs-gpu-task-def \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
    g4dn.xlarge" --region us-east-2
  ```

- Untuk setiap kontainer yang memiliki persyaratan sumber daya GPU yang ditentukan dalam definisi kontainer, Amazon ECS menetapkan runtime kontainer menjadi runtime kontainer NVIDIA.
- Runtime kontainer NVIDIA membutuhkan beberapa variabel lingkungan untuk disetel dalam wadah agar berfungsi dengan baik. Untuk daftar variabel lingkungan ini, lihat Konfigurasi Khusus dengan Docker. Amazon ECS menetapkan nilai variabel `NVIDIA_VISIBLE_DEVICES` lingkungan menjadi daftar perangkat GPU yang ditetapkan IDs Amazon ECS ke wadah. Untuk variabel lingkungan lain yang diperlukan, Amazon ECS tidak mengaturnya. Jadi, pastikan gambar kontainer Anda menyetelnya atau disetel dalam definisi kontainer.

- Keluarga tipe instans p5 didukung pada versi `20230929` dan yang lebih baru dari AMI yang dioptimalkan untuk GPU Amazon ECS.
- Keluarga tipe instans g4 didukung pada versi `20230913` dan yang lebih baru dari AMI yang dioptimalkan untuk GPU Amazon ECS. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs. Ini tidak didukung dalam alur kerja Create Cluster di konsol Amazon ECS. Untuk menggunakan jenis instance ini, Anda harus menggunakan EC2 konsol Amazon AWS CLI, atau API dan mendaftarkan instance secara manual ke klaster Anda.
- Tipe instans p4d.24xlarge hanya bekerja dengan CUDA 11 atau yang lebih baru.
- AMI yang dioptimalkan GPU Amazon ECS IPv6 telah diaktifkan, yang menyebabkan masalah saat menggunakan. `yum` Ini dapat diatasi dengan mengkonfigurasi `yum` untuk digunakan IPv4 dengan perintah berikut.

  ```
  echo "ip_resolve=4" >> /etc/yum.conf
  ```

- Saat Anda membuat image kontainer yang tidak menggunakan gambar NVIDIA/CUDA dasar, Anda harus menyetel variabel runtime `NVIDIA_DRIVER_CAPABILITIES` container ke salah satu nilai berikut:
  - `utility,compute`
  - `all`

  Untuk informasi tentang cara mengatur variabel, lihat Mengontrol Runtime Kontainer NVIDIA di situs web NVIDIA.
- GPUs tidak didukung pada wadah Windows.

Luncurkan instans kontainer GPU untuk Amazon ECS

Untuk menggunakan instance GPU di Amazon ECS, Anda perlu membuat template peluncuran, file data pengguna, dan meluncurkan instance.

Anda kemudian dapat menjalankan tugas yang menggunakan definisi tugas yang dikonfigurasi untuk GPU.

Menggunakan templat peluncuran

Anda dapat membuat template peluncuran.

- Buat template peluncuran yang menggunakan ID AMI GPU Amazon ECS yang dioptimalkan Untuk AMI. Untuk informasi tentang cara membuat templat peluncuran, lihat Membuat templat

peluncuran baru menggunakan parameter yang Anda tentukan di Panduan EC2 Pengguna Amazon.

Gunakan ID AMI dari langkah sebelumnya untuk image Amazon Machine. Untuk informasi tentang cara menentukan ID AMI dengan parameter Systems Manager, lihat Menentukan parameter Systems Manager dalam template peluncuran di Panduan EC2 Pengguna Amazon.

Tambahkan yang berikut ini ke data Pengguna di template peluncuran. Ganti *cluster-name* dengan nama klaster Anda.

```
#!/bin/bash
echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
```

Gunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk meluncurkan instance container.

1. Buat file yang dipanggil`userdata.toml`. File ini digunakan untuk data pengguna contoh. Ganti *cluster-name* dengan nama klaster Anda.

   ```
   #!/bin/bash
   echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
   echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
   ```

2. Jalankan perintah berikut untuk mendapatkan ID AMI GPU. Anda menggunakan ini pada langkah berikut.

   ```
   aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
   recommended --region us-east-1
   ```

3. Jalankan perintah berikut untuk meluncurkan instance GPU. Ingatlah untuk mengganti parameter berikut:

   - Ganti *subnet* dengan ID subnet pribadi atau publik tempat instans Anda akan diluncurkan.
   - Ganti *gpu_ami* dengan ID AMI dari langkah sebelumnya.
   - Ganti *t3.large* dengan jenis instance yang ingin Anda gunakan.
   - Ganti *region* dengan kode Region.

```
aws ec2 run-instances --key-name ecs-gpu-example \
   --subnet-id subnet \
   --image-id gpu_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications 'ResourceType=instance,Tags=[{Key=GPU,Value=example}]' \
   --user-data file://userdata.toml \
   --iam-instance-profile Name=ecsInstanceRole
```

4. Jalankan perintah berikut untuk memverifikasi bahwa instance container terdaftar ke cluster. Saat Anda menjalankan perintah ini, ingatlah untuk mengganti parameter berikut:

   - Ganti *cluster* dengan nama klaster Anda.
   - Ganti *region* dengan kode Wilayah Anda.

```
aws ecs list-container-instances --cluster cluster-name --region region
```

Menentukan GPUs dalam definisi tugas Amazon ECS

Untuk menggunakan instance GPUs on a container dan runtime GPU Docker, pastikan Anda menetapkan jumlah container yang dibutuhkan dalam definisi GPUs tugas. Saat wadah yang mendukung GPUs ditempatkan, agen kontainer Amazon ECS menyematkan jumlah fisik yang diinginkan GPUs ke wadah yang sesuai. Jumlah GPUs cadangan untuk semua kontainer dalam tugas tidak dapat melebihi jumlah yang tersedia GPUs pada instance kontainer tempat tugas diluncurkan. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

> **Important**
>
> Jika persyaratan GPU Anda tidak ditentukan dalam ketentuan tugas, tugas akan menggunakan waktu aktif Docker default.

Berikut ini menunjukkan format JSON untuk persyaratan GPU dalam ketentuan tugas:

```
{
```

```
  "containerDefinitions": [
     {
        ...
        "resourceRequirements" : [
            {
               "type" : "GPU",
               "value" : "2"
            }
        ],
     },
...
}
```

Contoh berikut menunjukkan sintaksis untuk kontainer Docker yang menentukan persyaratan GPU. Wadah ini menggunakan dua GPUs, menjalankan `nvidia-smi` utilitas, dan kemudian keluar.

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:11.0.3-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
  ],
  "family": "example-ecs-gpu"
}
```

Bagikan GPUs

Saat Anda ingin berbagi GPUs, Anda perlu mengkonfigurasi yang berikut ini.

1. Hapus persyaratan sumber daya GPU dari definisi tugas Anda sehingga Amazon ECS tidak mencadangkan semua GPUs yang harus dibagikan.
2. Tambahkan data pengguna berikut ke instance Anda saat Anda ingin berbagi GPUs. Ini akan menjadikan nvidia sebagai runtime kontainer Docker default pada instance container sehingga semua container Amazon ECS dapat menggunakan. GPUs Untuk informasi selengkapnya, lihat Menjalankan perintah saat Anda meluncurkan EC2 instance dengan input data pengguna di Panduan EC2 Pengguna Amazon.

```
const userData = ec2.UserData.forLinux();
 userData.addCommands(
 'sudo rm /etc/sysconfig/docker',
 'echo DAEMON_MAXFILES=1048576 | sudo tee -a /etc/sysconfig/docker',
 'echo OPTIONS="--default-ulimit nofile=32768:65536 --default-runtime nvidia" |
 sudo tee -a /etc/sysconfig/docker',
 'echo DAEMON_PIDFILE_TIMEOUT=10 | sudo tee -a /etc/sysconfig/docker',
 'sudo systemctl restart docker',
);
```

3. Atur variabel `NVIDIA_VISIBLE_DEVICES` lingkungan pada wadah Anda. Anda dapat melakukan ini dengan menentukan variabel lingkungan dalam definisi tugas Anda. Untuk informasi tentang nilai yang valid, lihat Pencacahan GPU di situs dokumentasi NVIDIA.

Apa yang harus dilakukan jika Anda membutuhkan instance P2

Jika Anda perlu menggunakan instance P2, Anda dapat menggunakan salah satu opsi berikut untuk terus menggunakan instance.

Anda harus memodifikasi data pengguna instance untuk kedua opsi. Untuk informasi selengkapnya, lihat Menjalankan perintah saat Anda meluncurkan EC2 instance dengan input data pengguna di Panduan EC2 Pengguna Amazon.

Gunakan AMI yang dioptimalkan untuk GPU terakhir yang didukung

Anda dapat menggunakan `20230906` versi AMI yang dioptimalkan untuk GPU, dan menambahkan yang berikut ini ke data pengguna instance.

Ganti nama cluster dengan nama cluster Anda.

```
#!/bin/bash
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
```

```
echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
```

Gunakan AMI terbaru yang dioptimalkan untuk GPU, dan perbarui data pengguna

Anda dapat menambahkan berikut ini ke data pengguna instance. Ini menghapus instalasi driver Nvidia 535/Cuda12.2, dan kemudian menginstal driver Nvidia 470/Cuda11.4 dan memperbaiki versi.

```
#!/bin/bash
yum remove -y cuda-toolkit* nvidia-driver-latest-dkms*
tmpfile=$(mktemp)
cat >$tmpfile <<EOF
[amzn2-nvidia]
name=Amazon Linux 2 Nvidia repository
mirrorlist=\$awsproto://\$amazonlinux.\$awsregion.\$awsdomain/\$releasever/amzn2-
nvidia/latest/\$basearch/mirror.list
priority=20
gpgcheck=1
gpgkey=https://developer.download.nvidia.com/compute/cuda/repos/rhel7/
x86_64/7fa2af80.pub
enabled=1
exclude=libglvnd-*
EOF

mv $tmpfile /etc/yum.repos.d/amzn2-nvidia-tmp.repo
yum install -y system-release-nvidia cuda-toolkit-11-4 nvidia-driver-latest-
dkms-470.182.03
yum install -y libnvidia-container-1.4.0 libnvidia-container-tools-1.4.0 nvidia-
container-runtime-hook-1.4.0 docker-runtime-nvidia-1

echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
nvidia-smi
```

Buat AMI yang dioptimalkan dengan GPU yang kompatibel dengan P2 Anda sendiri

Anda dapat membuat AMI yang dioptimalkan untuk GPU Amazon ECS kustom Anda sendiri yang kompatibel dengan instans P2, lalu meluncurkan instans P2 menggunakan AMI.

1. Jalankan perintah berikut untuk mengkloning file. `amazon-ecs-ami repo`

   ```
   git clone https://github.com/aws/amazon-ecs-ami
   ```

2. Tetapkan agen Amazon ECS yang diperlukan dan sumber versi AMI Amazon Linux di `release.auto.pkrvars.hcl` atau`overrides.auto.pkrvars.hcl`.
3. Jalankan perintah berikut untuk membangun EC2 AMI yang kompatibel dengan P2 pribadi.

   Ganti region dengan Region dengan instance Region.

   ```
   REGION=region make al2keplergpu
   ```

4. Gunakan AMI dengan data pengguna instans berikut untuk menyambung ke klaster Amazon ECS.

   Ganti nama cluster dengan nama cluster Anda.

   ```
   #!/bin/bash
   echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
   ```

Definisi tugas Amazon ECS untuk beban kerja transcoding video

Untuk menggunakan beban kerja transcoding video di Amazon ECS, daftarkan instans Amazon. EC2 VT1 Setelah mendaftarkan instans ini, Anda dapat menjalankan beban kerja transcoding video langsung dan pra-render sebagai tugas di Amazon ECS. EC2 VT1 Instans Amazon menggunakan kartu transcoding media Xilinx U30 untuk mempercepat beban kerja transcoding video langsung dan pra-render.

> **Note**
>
> Untuk petunjuk tentang cara menjalankan beban kerja transcoding video dalam wadah selain Amazon ECS, lihat dokumentasi Xilinx.

Pertimbangan

Sebelum Anda mulai menerapkan VT1 di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran VT1 dan VT1 non-instance.
- Anda memerlukan aplikasi Linux yang menggunakan kartu transcoding media Xilinx U30 dengan codec AVC (H.264) dan HEVC (H.265) yang dipercepat.

> **Important**
>
> Aplikasi yang menggunakan codec lain mungkin tidak memiliki kinerja yang lebih baik pada VT1 instance.

- Hanya satu tugas transcoding yang dapat dijalankan pada kartu U30. Setiap kartu memiliki dua perangkat yang terkait dengannya. Anda dapat menjalankan tugas transcoding sebanyak mungkin karena ada kartu untuk setiap VT1 instance Anda.
- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas. Ini memastikan bahwa tugas diluncurkan pada instance container yang Anda tentukan. Melakukannya membantu memastikan bahwa Anda menggunakan sumber daya Anda secara efektif dan bahwa tugas Anda untuk beban kerja transcoding video ada di instans Anda VT1 . Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Dalam contoh berikut, tugas dijalankan pada `vt1.3xlarge` instance di `default` cluster Anda.

  ```
  aws ecs run-task \
       --cluster default \
       --task-definition vt1-3xlarge-xffmpeg-processor \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
   vt1.3xlarge"
  ```

- Anda mengonfigurasi kontainer untuk menggunakan kartu U30 spesifik yang tersedia pada instance wadah host. Anda dapat melakukan ini dengan menggunakan `linuxParameters` parameter dan menentukan detail perangkat. Untuk informasi selengkapnya, lihat Persyaratan ketentuan tugas.

Menggunakan VT1 AMI

Anda memiliki dua opsi untuk menjalankan AMI di Amazon EC2 untuk instans penampung Amazon ECS. Opsi pertama adalah menggunakan AMI resmi Xilinx di. AWS Marketplace Opsi kedua adalah membangun AMI Anda sendiri dari repositori sampel.

- Xilinx menawarkan AMIs pada. AWS Marketplace
- Amazon ECS menyediakan contoh repositori yang dapat Anda gunakan untuk membangun AMI untuk beban kerja transcoding video. AMI ini hadir dengan driver Xilinx U30. Anda dapat

menemukan repositori yang berisi skrip Packer di. GitHub Untuk informasi selengkapnya tentang Packer, lihat dokumentasi Packer.

Persyaratan ketentuan tugas

Untuk menjalankan kontainer transcoding video di Amazon ECS, definisi tugas Anda harus berisi aplikasi transcoding video yang menggunakan codec H.264/AVC dan H.265/HEVC yang dipercepat. Anda dapat membuat gambar kontainer dengan mengikuti langkah-langkah di GitHubXilinx.

Definisi tugas harus spesifik untuk jenis instance. Jenis instans adalah 3xlarge, 6xlarge, dan 24xlarge. Anda harus mengonfigurasi kontainer untuk menggunakan perangkat Xilinx U30 tertentu yang tersedia pada instance wadah host. Anda dapat melakukannya dengan menggunakan `linuxParameters` parameter. Tabel berikut merinci kartu dan perangkat SoCs yang spesifik untuk setiap jenis instance.

Tipe Instans	v CPUs	RAM (GiB)	Kartu akselerator U30	Perangkat XCU3 0 SoC yang dapat dialamatkan	Jalur Perangkat
vt1.3xlarge	12	24	1	2	`/dev/dri/renderD128 ,/dev/dri/renderD129`
vt1.6xlarge	24	48	2	4	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/`

Tipe Instans	v CPUs	RAM (GiB)	Kartu akselerator U30	Perangkat XCU3 0 SoC yang dapat dialamatkan	Jalur Perangkat
					`renderD131`

Tipe Instans	v CPUs	RAM (GiB)	Kartu akselerator U30	Perangkat XCU3 0 SoC yang dapat dialamatkan	Jalur Perangkat
vt1.24xlarge	96	182	8	16	/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131 ,/dev/dri/renderD132 ,/dev/dri/renderD133 ,/dev/dri/renderD134 ,/dev/dri/renderD135 ,/dev/dri/renderD136 ,/dev/dri/renderD137 ,/dev/dri/

Tipe Instans	v CPUs	RAM (GiB)	Kartu akselerator U30	Perangkat XCU3 0 SoC yang dapat dialamatkan	Jalur Perangkat
					`renderD138 ,/dev/dri/renderD139 ,/dev/dri/renderD140 ,/dev/dri/renderD141 ,/dev/dri/renderD142 ,/dev/dri/renderD143`

> **Important**
>
> Jika definisi tugas mencantumkan perangkat yang tidak dimiliki EC2 instance, tugas gagal dijalankan. Ketika tugas gagal, pesan kesalahan berikut muncul `distoppedReason:CannotStartContainerError: Error response from daemon: error gathering device information while adding custom device "/dev/dri/renderD130": no such file or directory.`

Menentukan transcoding video dalam definisi tugas Amazon ECS

Dalam contoh berikut, sintaks yang digunakan untuk definisi tugas dari wadah Linux di Amazon EC2 disediakan. Definisi tugas ini adalah untuk gambar kontainer yang dibangun mengikuti prosedur

yang disediakan dalam dokumentasi Xilinx. Jika Anda menggunakan contoh ini, ganti `image` dengan gambar Anda sendiri, dan salin file video Anda ke instance di `/home/ec2-user` direktori.

vt1.3xlarge

1. Buat file teks yang diberi nama `vt1-3xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-3xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.3xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
```

```
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-3xlarge-xffmpeg-processor --cli-
input-json file://vt1-3xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.6xlarge

1. Buat file teks yang diberi nama `vt1-6xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-6xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
```

```
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.6xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
```

```
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-6xlarge-xffmpeg-processor --cli-
input-json file://vt1-6xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.24xlarge

1. Buat file teks yang diberi nama `vt1-24xlarge-ffmpeg-linux.json` dengan konten berikut.

```
{
    "family": "vt1-24xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
```

```
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.24xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
```

```
                    "containerPath": "/dev/dri/renderD131",
                    "hostPath": "/dev/dri/renderD131",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD132",
                    "hostPath": "/dev/dri/renderD132",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD133",
                    "hostPath": "/dev/dri/renderD133",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD134",
                    "hostPath": "/dev/dri/renderD134",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD135",
                    "hostPath": "/dev/dri/renderD135",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD136",
                    "hostPath": "/dev/dri/renderD136",
                    "permissions": [
                        "read",
```

```
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD137",
                    "hostPath": "/dev/dri/renderD137",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD138",
                    "hostPath": "/dev/dri/renderD138",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD139",
                    "hostPath": "/dev/dri/renderD139",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD140",
                    "hostPath": "/dev/dri/renderD140",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD141",
                    "hostPath": "/dev/dri/renderD141",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
```

```
                        "containerPath": "/dev/dri/renderD142",
                        "hostPath": "/dev/dri/renderD142",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD143",
                        "hostPath": "/dev/dri/renderD143",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. Mendaftarkan ketentuan tugas.

```
aws ecs register-task-definition --family vt1-24xlarge-xffmpeg-processor --cli-
input-json file://vt1-24xlarge-xffmpeg-linux.json --region us-east-1
```

Definisi tugas Amazon ECS untuk beban kerja pembelajaran mesin AWS Neuron

Anda dapat mendaftarkan instans Amazon EC2 Trn1, Amazon EC2 Inf1, dan Amazon EC2 Inf2 ke cluster Anda untuk beban kerja pembelajaran mesin.

Instans Amazon EC2 Trn1 didukung oleh AWS chip Trainium. Instans ini memberikan pelatihan berkinerja tinggi dan biaya rendah untuk pembelajaran mesin di cloud. Anda dapat melatih model inferensi pembelajaran mesin menggunakan kerangka pembelajaran mesin dengan AWS Neuron pada instance Trn1. Kemudian, Anda dapat menjalankan model pada instance Inf1, atau instance Inf2 untuk menggunakan akselerasi chip AWS Inferentia.

Instans Amazon EC2 Inf1 dan instans Inf2 didukung oleh chip AWS Inferentia. Instans ini memberikan kinerja tinggi dan inferensi biaya terendah di cloud.

Model pembelajaran mesin digunakan ke wadah menggunakan AWS Neuron, yang merupakan Kit Pengembang Perangkat Lunak (SDK) khusus. SDK terdiri dari compiler, runtime, dan alat profiling yang mengoptimalkan kinerja machine learning chip machine learning. AWS AWS Neuron mendukung kerangka kerja pembelajaran mesin populer seperti TensorFlow, PyTorch, dan MXNet Apache.

Pertimbangan

Sebelum Anda mulai menerapkan Neuron di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran Trn1, Inf1, Inf2, dan instance lainnya.
- Anda memerlukan aplikasi Linux dalam wadah yang menggunakan kerangka pembelajaran mesin yang mendukung AWS Neuron.

> **Important**
>
> Aplikasi yang menggunakan kerangka kerja lain mungkin tidak meningkatkan kinerja pada instance Trn1, Inf1, dan Inf2.

- Hanya satu tugas inferensi atau pelatihan inferensi yang dapat dijalankan pada setiap chip AWS Trainium atau Inferentia.AWS Untuk Inf1, setiap chip memiliki 4 NeuronCores. Untuk Trn1 dan Inf2 setiap chip memiliki 2. NeuronCores Anda dapat menjalankan tugas sebanyak mungkin karena ada chip untuk setiap instans Trn1, Inf1, dan Inf2 Anda.

- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance saat mengonfigurasi batasan penempatan tugas. Ini memastikan bahwa tugas diluncurkan pada instance container yang Anda tentukan. Melakukannya dapat membantu Anda mengoptimalkan pemanfaatan sumber daya secara keseluruhan dan memastikan bahwa tugas untuk beban kerja inferensi ada di instans Trn1, Inf1, dan Inf2 Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Dalam contoh berikut, tugas dijalankan pada `Inf1.xlarge` instance di `default` cluster Anda.

  ```
  aws ecs run-task \
       --cluster default \
       --task-definition ecs-inference-task-def \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
   Inf1.xlarge"
  ```

- Persyaratan sumber daya neuron tidak dapat didefinisikan dalam definisi tugas. Sebagai gantinya, Anda mengonfigurasi wadah untuk menggunakan chip AWS Trainium atau AWS Inferentia tertentu yang tersedia di instance wadah host. Lakukan ini dengan menggunakan `linuxParameters` parameter dan menentukan detail perangkat. Untuk informasi selengkapnya, lihat Persyaratan ketentuan tugas.

Gunakan Amazon Amazon Linux 2023 (Neuron) AMI yang dioptimalkan Amazon ECS

Amazon ECS menyediakan AMI Amazon ECS yang dioptimalkan yang didasarkan pada Amazon Linux 2023 untuk beban kerja AWS Trainium dan Inferentia. AWS Muncul dengan driver AWS Neuron dan runtime untuk Docker. AMI ini membuat menjalankan beban kerja inferensi pembelajaran mesin lebih mudah di Amazon ECS.

Sebaiknya gunakan AMI Amazon Linux 2023 (Neuron) Amazon ECS yang dioptimalkan Amazon saat meluncurkan instans Amazon EC2 Trn1, Inf1, dan Inf2 Anda.

Anda dapat mengambil AMI Amazon Linux 2023 (Neuron) Amazon ECS yang dioptimalkan saat AWS CLI ini menggunakan perintah berikut.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/
recommended
```

Amazon ECS yang dioptimalkan Amazon Linux 2023 (Neuron) AMI didukung di Wilayah berikut:

- Timur AS (N. Virginia)

- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Tokyo)
- Asia Pasifik (Singapura)
- Asia Pacific (Sydney)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Stockholm)
- Amerika Selatan (Sao Paulo)

Gunakan Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI

Amazon ECS menyediakan AMI Amazon ECS yang dioptimalkan yang didasarkan pada Amazon Linux 2 untuk beban kerja AWS Trainium dan AWS Inferentia. Muncul dengan driver AWS Neuron dan runtime untuk Docker. AMI ini membuat menjalankan beban kerja inferensi pembelajaran mesin lebih mudah di Amazon ECS.

Sebaiknya gunakan AMI Amazon Linux 2 (Neuron) Amazon ECS yang dioptimalkan saat meluncurkan instans Amazon EC2 Trn1, Inf1, dan Inf2 Anda.

Anda dapat mengambil Amazon ECS saat ini yang dioptimalkan Amazon Linux 2 (Neuron) AMI menggunakan AWS CLI perintah berikut.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended
```

Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI didukung di Wilayah berikut:

- Timur AS (N. Virginia)
- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Tokyo)
- Asia Pasifik (Singapura)
- Asia Pacific (Sydney)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Stockholm)
- Amerika Selatan (Sao Paulo)

Persyaratan ketentuan tugas

Untuk menerapkan Neuron di Amazon ECS, definisi tugas Anda harus berisi definisi kontainer untuk wadah bawaan yang menyajikan model inferensi. TensorFlow Ini disediakan oleh AWS Deep Learning Containers. Wadah ini berisi runtime AWS Neuron dan aplikasi TensorFlow Serving. Saat startup, penampung ini mengambil model Anda dari Amazon S3, meluncurkan Penyajian TensorFlow Neuron dengan model yang disimpan, dan menunggu permintaan prediksi. Dalam contoh berikut, gambar kontainer memiliki TensorFlow 1.15 dan Ubuntu 18.04. Daftar lengkap Deep Learning Containers pra-bangun yang dioptimalkan untuk Neuron GitHub dipertahankan. Untuk informasi lebih lanjut, lihat Menggunakan TensorFlow Penyajian AWS Neuron.

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-neuron:1.15.4-neuron-
py37-ubuntu18.04
```

Atau, Anda dapat membangun sendiri citra kontainer Neuron sespan. Untuk informasi lebih lanjut, lihat Tutorial: TensorFlow Penyajian Neuron di Panduan AWS Deep Learning AMIs Pengembang.

Definisi tugas harus spesifik untuk satu jenis instance. Anda harus mengonfigurasi wadah untuk menggunakan perangkat AWS Trainium atau AWS Inferentia tertentu yang tersedia di instance wadah host. Anda dapat melakukannya dengan menggunakan `linuxParameters` parameter. Tabel berikut merinci chip yang spesifik untuk setiap jenis instance.

Tipe Instans	v CPUs	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
trn1.2xlarge	8	32	1	`/dev/neuron0`
trn1.32xlarge	128	512	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15`
inf1.xlarge	4	8	1	`/dev/neuron0`

Tipe Instans	v CPUs	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
inf1.2xlarge	8	16	1	`/dev/neuron0`
inf1.6xlarge	24	48	4	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3`
inf1.24xlarge	96	192	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15`

Tipe Instans	v CPUs	RAM (GiB)	AWS Chip akselerator mL	Jalur Perangkat
inf2.xlarge	8	16	1	`/dev/neuron0`
inf2.8xlarge	32	64	1	`/dev/neuron0`
inf2.24xlarge	96	384	6	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,`
inf2.48xlarge	192	768	12	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11`

Menentukan pembelajaran mesin AWS Neuron dalam definisi tugas Amazon ECS

Berikut ini adalah contoh definisi tugas Linux untuk`inf1.xlarge`, menampilkan sintaks yang akan digunakan.

```
{
    "family": "ecs-neuron",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == inf1.xlarge"
        }
    ],
    "executionRoleArn": "${YOUR_EXECUTION_ROLE}",
    "containerDefinitions": [
        {
            "entryPoint": [
                "/usr/local/bin/entrypoint.sh",
                "--port=8500",
                "--rest_api_port=9000",
                "--model_name=resnet50_neuron",
                "--model_base_path=s3://amzn-s3-demo-bucket/resnet50_neuron/"
            ],
            "portMappings": [
                {
                    "hostPort": 8500,
                    "protocol": "tcp",
                    "containerPort": 8500
                },
                {
                    "hostPort": 8501,
                    "protocol": "tcp",
                    "containerPort": 8501
                },
                {
                    "hostPort": 0,
                    "protocol": "tcp",
                    "containerPort": 80
```

```
                }
            ],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/neuron0",
                        "hostPath": "/dev/neuron0",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ],
                "capabilities": {
                    "add": [
                        "IPC_LOCK"
                    ]
                }
            },
            "cpu": 0,
            "memoryReservation": 1000,
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference-neuron:1.15.4-neuron-py37-ubuntu18.04",
            "essential": true,
            "name": "resnet50"
        }
    ]
}
```

Definisi tugas Amazon ECS untuk instans pembelajaran mendalam

Untuk menggunakan beban kerja pembelajaran mendalam di Amazon ECS, daftarkan EC2 DL1 instans Amazon ke klaster Anda. EC2DL1 Instans Amazon didukung oleh akselerator Gaudi dari Habana Labs (perusahaan Intel). Gunakan Habana SynapSeai SDK untuk terhubung ke akselerator Habana Gaudi. SDK mendukung kerangka kerja pembelajaran mesin yang populer, TensorFlow dan. PyTorch

Pertimbangan

Sebelum Anda mulai menerapkan DL1 di Amazon ECS, pertimbangkan hal berikut:

- Cluster Anda dapat berisi campuran DL1 dan DL1 non-instance.

- Saat membuat layanan atau menjalankan tugas mandiri, Anda dapat menggunakan atribut tipe instance secara khusus saat mengonfigurasi batasan penempatan tugas untuk memastikan bahwa tugas Anda diluncurkan pada instance container yang Anda tentukan. Melakukannya memastikan bahwa sumber daya Anda digunakan secara efektif dan bahwa tugas Anda untuk beban kerja pembelajaran mendalam ada di DL1 instans Anda. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

  Contoh berikut menjalankan tugas pada instans `dl1.24xlarge` di klaster `default` Anda.

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-dl1-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 dl1.24xlarge"
```

Menggunakan DL1 AMI

Anda memiliki tiga opsi untuk menjalankan AMI di EC2 DL1 instans Amazon untuk Amazon ECS:

- AWS Marketplace AMIs yang disediakan oleh Habana di sini.
- Habana Deep Learning AMIs yang disediakan oleh Amazon Web Services. Karena tidak termasuk, Anda perlu menginstal agen kontainer Amazon ECS secara terpisah.
- Gunakan Packer untuk membuat AMI khusus yang disediakan oleh GitHubrepo. Untuk informasi selengkapnya, lihat dokumentasi Packer.

Menentukan pembelajaran mendalam dalam definisi tugas Amazon ECS

Untuk menjalankan wadah pembelajaran mendalam yang dipercepat Habana Gaudi di Amazon ECS, definisi tugas Anda harus berisi definisi wadah untuk wadah pra-bangun yang menyajikan model pembelajaran mendalam untuk TensorFlow atau PyTorch menggunakan Habana SynapSeai yang disediakan oleh Deep Learning Containers. AWS

Gambar kontainer berikut memiliki TensorFlow 2.7.0 dan Ubuntu 20.04. Daftar lengkap Deep Learning Containers pra-bangun yang dioptimalkan untuk akselerator Habana Gaudi dipertahankan. GitHub Untuk informasi lebih lanjut, lihat Wadah Pelatihan Habana.

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training-habana:2.7.0-hpu-py38-
synapseai1.2.0-ubuntu20.04
```

Berikut ini adalah contoh definisi tugas untuk wadah Linux di Amazon EC2, menampilkan sintaks yang akan digunakan. Contoh ini menggunakan gambar yang berisi Habana Labs System Management Interface Tool (HL-SMI) yang ditemukan di sini: `vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/tensorflow-installer-tf-cpu-2.6.0:1.1.0-614`

```
{
    "family": "dl-test",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == dl1.24xlarge"
        }
    ],
    "networkMode": "host",
    "cpu": "10240",
    "memory": "1024",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": ["hl-smi"],
            "cpu": 8192,
            "environment": [
                {
                    "name": "HABANA_VISIBLE_DEVICES",
                    "value": "all"
                }
            ],
            "image": "vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/
tensorflow-installer-tf-cpu-2.6.0:1.1.0-614",
            "essential": true,
```

```
            "name": "tensorflow-installer-tf-hpu"
        }
    ]
}
```

Definisi tugas Amazon ECS untuk beban kerja ARM 64-bit

Amazon ECS mendukung penggunaan aplikasi ARM 64-bit. Anda dapat menjalankan aplikasi Anda di platform yang didukung oleh AWS Graviton Processors. Ini cocok untuk berbagai macam beban kerja. Ini termasuk beban kerja seperti server aplikasi, layanan mikro, komputasi kinerja tinggi, inferensi pembelajaran mesin berbasis CPU, pengkodean video, otomatisasi desain elektronik, game, database sumber terbuka, dan cache dalam memori.

Pertimbangan

Sebelum Anda mulai menerapkan definisi tugas yang menggunakan arsitektur ARM 64-bit, pertimbangkan hal berikut:

- Aplikasi dapat menggunakan Fargate atau tipe EC2 peluncuran.
- Aplikasi hanya dapat menggunakan sistem operasi Linux.
- Untuk tipe Fargate, aplikasi harus menggunakan versi platform Fargate atau yang lebih baru. `1.4.0`
- Aplikasi dapat digunakan Fluent Bit atau CloudWatch untuk pemantauan.
- Untuk jenis peluncuran Fargate, berikut ini Wilayah AWS tidak mendukung beban kerja ARM 64-bit:
  - AS Timur (Virginia N.), Zona `use1-az3` Ketersediaan
- Untuk jenis EC2 peluncuran Amazon, lihat berikut ini untuk memverifikasi bahwa Wilayah tempat Anda berada mendukung jenis instans yang ingin Anda gunakan:
  - Instans Amazon EC2 M6g
  - Instans Amazon EC2 T4G
  - Instans Amazon EC2 C6g
  - Instans Amazon EC2 R6gd
  - Instans Amazon EC2 X2GD

  Anda juga dapat menggunakan EC2 `describe-instance-type-offerings` perintah Amazon dengan filter untuk melihat penawaran instance untuk Wilayah Anda.

```
aws ec2 describe-instance-type-offerings --filters Name=instance-
type,Values=instance-type --region region
```

Contoh berikut memeriksa ketersediaan tipe instans M6 di Wilayah AS Timur (Virginia N.) (us-east-1).

```
aws ec2 describe-instance-type-offerings --filters "Name=instance-type,Values=m6*" --
region us-east-1
```

Untuk informasi selengkapnya, lihat describe-instance-type-offerings di Referensi Baris EC2 Perintah Amazon.

Menentukan arsitektur ARM dalam definisi tugas Amazon ECS

Untuk menggunakan arsitektur ARM, `ARM64` tentukan parameter definisi `cpuArchitecture` tugas.

Dalam contoh berikut, arsitektur ARM ditentukan dalam definisi tugas. Ini dalam format JSON.

```
{
    "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
    },
...
}
```

Dalam contoh berikut, definisi tugas untuk arsitektur ARM menampilkan “hello world.”

```
{
 "family": "arm64-testapp",
 "networkMode": "awsvpc",
 "containerDefinitions": [
    {
        "name": "arm-container",
        "image": "public.ecr.aws/docker/library/busybox:latest",
        "cpu": 100,
        "memory": 100,
        "essential": true,
        "command": [ "echo hello world" ],
```

```
        "entryPoint": [ "sh", "-c" ]
    }
 ],
 "requiresCompatibilities": [ "EC2" ],
 "cpu": "256",
 "memory": "512",
 "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
  },
 "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

Kirim log Amazon ECS ke CloudWatch

Anda dapat mengonfigurasi kontainer dalam tugas Anda untuk mengirim informasi CloudWatch log ke Log. Jika Anda menggunakan tipe peluncuran Fargate untuk tugas Anda, Anda dapat melihat log dari kontainer Anda. Jika Anda menggunakan tipe EC2 peluncuran, Anda dapat melihat log yang berbeda dari kontainer Anda di satu lokasi yang nyaman, dan ini mencegah log kontainer Anda mengambil ruang disk pada instance kontainer Anda.

> **Note**
>
> Jenis informasi yang dicatat oleh kontainer dalam tugas Anda sebagian besar bergantung pada perintah `ENTRYPOINT` mereka. Secara default, log yang diambil menunjukkan output perintah yang biasanya Anda lihat di terminal interaktif jika Anda menjalankan kontainer secara lokal, yang merupakan aliran `STDOUT` dan `STDERR` I/O. Driver `awslogs` log hanya meneruskan log ini dari Docker ke CloudWatch Logs. Untuk informasi selengkapnya tentang cara log Docker diproses, termasuk cara alternatif untuk menangkap berbagai pengaliran atau data file, lihat Melihat log untuk kontainer atau layanan dalam dokumentasi Docker.

Untuk mengirim log sistem dari instans penampung Amazon ECS Anda ke CloudWatch Log, lihat Memantau Kuota File Log dan CloudWatch Log di Panduan Pengguna Amazon CloudWatch Logs.

Jenis peluncuran Fargate

Jika Anda menggunakan tipe peluncuran Fargate untuk tugas Anda, Anda perlu menambahkan `logConfiguration` parameter yang diperlukan ke definisi tugas Anda untuk mengaktifkan driver

`awslogs` log. Untuk informasi selengkapnya, lihat Contoh definisi tugas Amazon ECS: Rute log ke CloudWatch.

Untuk penampung Windows di Fargate, lakukan salah satu opsi berikut ketika salah satu parameter definisi tugas Anda memiliki karakter khusus seperti,: `& \ < > ^ |`

- Tambahkan escape (\) dengan tanda kutip ganda di sekitar seluruh string parameter

  Contoh

  ```
  "awslogs-multiline-pattern": "\"^[|DEBUG|INFO|WARNING|ERROR\"",
  ```

- Tambahkan karakter escape (^) di sekitar setiap karakter khusus

  Contoh

  ```
  "awslogs-multiline-pattern": "^^[^|DEBUG^|INFO^|WARNING^|ERROR",
  ```

EC2 jenis peluncuran

Jika Anda menggunakan jenis EC2 peluncuran untuk tugas Anda dan ingin mengaktifkan driver `awslogs` log, instans penampung Amazon ECS Anda memerlukan setidaknya versi 1.9.0 dari agen penampung. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.

> **Note**
>
> Anda harus menggunakan AMI Amazon ECS yang dioptimalkan atau AMI khusus dengan setidaknya `1.9.0-1` versi paket. `ecs-init` Saat menggunakan AMI kustom, Anda harus menentukan bahwa driver `awslogs` logging tersedia di EC2 instans Amazon saat memulai agen dengan menggunakan variabel lingkungan berikut dalam file variabel docker run pernyataan atau lingkungan Anda.
>
> ```
> ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
> ```

Instans penampung Amazon ECS Anda juga memerlukan `logs:CreateLogStream` dan `logs:PutLogEvents` izin pada peran IAM yang dapat digunakan untuk meluncurkan instans

penampung. Jika Anda membuat peran instans penampung Amazon ECS sebelum dukungan driver `awslogs` log diaktifkan di Amazon ECS, Anda mungkin perlu menambahkan izin ini. `ecsTaskExecutionRole`Ini digunakan saat ditugaskan ke tugas dan kemungkinan berisi izin yang benar. Untuk informasi tentang peran eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS. Jika instance container Anda menggunakan kebijakan IAM terkelola untuk instance container, instance container Anda mungkin memiliki izin yang benar. Untuk informasi tentang kebijakan IAM terkelola untuk instance container, lihat. Peran IAM instance wadah Amazon ECS

Contoh definisi tugas Amazon ECS: Rute log ke CloudWatch

Sebelum kontainer Anda dapat mengirim log ke CloudWatch, Anda harus menentukan driver `awslogs` log untuk kontainer dalam definisi tugas Anda. Untuk informasi selengkapnya tentang parameter log, lihat Penyimpanan dan pencatatan

Definisi tugas JSON yang berikut memiliki `logConfiguration` objek yang ditentukan untuk setiap kontainer. Salah satunya adalah untuk WordPress wadah yang mengirim log ke grup log yang dipanggil`awslogs-wordpress`. Yang lainnya adalah untuk wadah MySQL yang mengirimkan log ke grup log yang dipanggil. `awslogs-mysql` Kedua kontainer menggunakan prefiks pengaliran log `awslogs-example`.

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "public.ecr.aws/docker/library/wordpress:latest",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
```

```
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
        {
            "environment": [
                {
                    "name": "MYSQL_ROOT_PASSWORD",
                    "value": "password"
                }
            ],
            "name": "mysql",
            "image": "public.ecr.aws/docker/library/mysql:latest",
            "cpu": 10,
            "memory": 500,
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example",
                    "mode": "non-blocking",
                    "max-buffer-size": "25m"
                }
            }
        }
    ],
    "family": "awslogs-example"
}
```

Langkah selanjutnya

- Anda dapat secara opsional menetapkan kebijakan retensi untuk grup log dengan menggunakan CloudWatch AWS CLI atau API. Untuk informasi selengkapnya, lihat put-retention-policy dalam Referensi AWS Command Line Interface .
- Setelah Anda mendaftarkan definisi tugas dengan driver `awslogs` log dalam konfigurasi log definisi kontainer, Anda dapat menjalankan tugas atau membuat layanan dengan definisi tugas tersebut untuk mulai mengirim CloudWatch log ke Log. Untuk informasi selengkapnya, lihat

Menjalankan aplikasi sebagai tugas Amazon ECS dan Membuat penyebaran pembaruan bergulir Amazon ECS.

Kirim log Amazon ECS ke AWS layanan atau AWS Partner

Anda dapat menggunakan Amazon ECS FireLens untuk menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan AWS Partner Network (APN) untuk penyimpanan log dan analitik. AWS Partner Network Ini adalah komunitas mitra global yang memanfaatkan program, keahlian, dan sumber daya untuk membangun, memasarkan, dan menjual penawaran pelanggan. Untuk informasi lebih lanjut lihat AWS Partner. FireLensbekerja dengan Fluentd danFluent Bit. Kami menyediakan AWS untuk Fluent Bit gambar atau Anda dapat menggunakan gambar Anda sendiri Fluentd atau Fluent Bit gambar.

Secara default, Amazon ECS mengonfigurasi dependensi penampung sehingga kontainer Firelens dimulai sebelum penampung apa pun yang menggunakannya. Wadah Firelens juga berhenti setelah semua wadah yang menggunakannya berhenti.

Pertimbangkan hal berikut saat menggunakan FireLens untuk Amazon ECS:

- Kami menyarankan Anda `my_service_` menambahkan nama wadah log sehingga Anda dapat dengan mudah membedakan nama kontainer di konsol.
- Amazon ECS menambahkan dependensi pesanan kontainer awal antara container aplikasi dan FireLens container secara default. Saat Anda menentukan urutan kontainer antara wadah aplikasi dan FireLens wadah, maka urutan kontainer awal default diganti.
- FireLensuntuk Amazon ECS didukung untuk tugas-tugas yang di-host AWS Fargate di Linux dan Amazon EC2 di Linux. Kontainer Windows tidak mendukungFireLens.

  Untuk informasi tentang cara mengonfigurasi pencatatan terpusat untuk kontainer Windows, lihat Pencatatan terpusat untuk kontainer Windows di Amazon ECS menggunakan Bit Lancar.
- Anda dapat menggunakan AWS CloudFormation templat FireLens untuk mengonfigurasi Amazon ECS. Untuk informasi selengkapnya, lihat AWS::ECS::TaskDefinition FirelensConfigurationdi Panduan AWS CloudFormation Pengguna
- FireLensmendengarkan di port`24224`, jadi untuk memastikan bahwa router FireLens log tidak dapat dijangkau di luar tugas, Anda tidak boleh mengizinkan lalu lintas masuk pada port `24224` dalam grup keamanan yang digunakan tugas Anda. Untuk tugas yang menggunakan mode `awsvpc` jaringan, ini adalah grup keamanan yang terkait dengan tugas tersebut. Untuk tugas yang menggunakan mode `host` jaringan, ini adalah grup keamanan yang terkait dengan EC2

instans Amazon yang menghosting tugas tersebut. Untuk tugas yang menggunakan mode `bridge` jaringan, jangan membuat pemetaan port apa pun yang menggunakan port. `24224`

- Untuk tugas yang menggunakan mode `bridge` jaringan, wadah dengan FireLens konfigurasi harus dimulai sebelum wadah aplikasi apa pun yang mengandalkannya dimulai. Untuk mengendalikan urutan mulai kontainer Anda, gunakan syarat dependensi dalam ketentuan tugas Anda. Untuk informasi selengkapnya, lihat Dependensi kontainer.

  > **Note**
  >
  > Jika Anda menggunakan parameter kondisi ketergantungan dalam definisi kontainer dengan FireLens konfigurasi, pastikan bahwa setiap kontainer memiliki persyaratan `START` atau `HEALTHY` kondisi.

- Secara default, FireLens tambahkan nama definisi klaster dan tugas serta Nama Sumber Daya Amazon (ARN) klaster sebagai kunci metadata ke log penampung Anda. stdout/stderr Berikut ini adalah contoh format metadata.

  ```
  "ecs_cluster": "cluster-name",
  "ecs_task_arn": "arn:aws:ecs:region:111122223333:task/cluster-
  name/f2ad7dba413f45ddb4EXAMPLE",
  "ecs_task_definition": "task-def-name:revision",
  ```

  Jika Anda tidak ingin metadata di log Anda, atur `enable-ecs-log-metadata` ke `false` `firelensConfiguration` bagian definisi tugas.

  ```
  "firelensConfiguration":{
     "type":"fluentbit",
     "options":{
        "enable-ecs-log-metadata":"false",
        "config-file-type":"file",
        "config-file-value":"/extra.conf"
  }
  ```

Untuk menggunakan fitur ini, Anda harus membuat peran IAM untuk tugas Anda yang memberikan izin yang diperlukan untuk menggunakan AWS layanan apa pun yang diperlukan tugas. Misalnya, jika kontainer merutekan log ke Firehose, tugas tersebut memerlukan izin untuk memanggil `firehose:PutRecordBatch` API. Untuk informasi selengkapnya, silakan lihat Menambahkan dan Menghapus Izin Identitas IAM dalam Panduan Pengguna IAM.

Tugas Anda mungkin juga memerlukan peran eksekusi tugas Amazon ECS dalam kondisi berikut. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

- Jika tugas Anda di-host di Fargate dan Anda menarik gambar kontainer dari Amazon ECR atau mereferensikan data sensitif dari AWS Secrets Manager konfigurasi log Anda, maka Anda harus menyertakan peran IAM eksekusi tugas.
- Saat Anda menggunakan file konfigurasi khusus yang di-host di Amazon S3, peran IAM eksekusi tugas Anda harus menyertakan izin. `s3:GetObject`

Untuk informasi tentang cara menggunakan beberapa file konfigurasi dengan Amazon ECS, termasuk file yang Anda host atau file di Amazon S3, lihat Proses init untuk Bit Lancar di ECS, dukungan multi-konfigurasi.

Untuk informasi tentang konfigurasi contoh, lihat Contoh definisi tugas Amazon ECS: Rute log ke FireLens

Mengkonfigurasi log Amazon ECS untuk throughput tinggi

Saat Anda membuat definisi tugas, Anda dapat menentukan jumlah baris log yang disangga dalam memori dengan menentukan nilai dalam. `log-driver-buffer-limit` Untuk informasi selengkapnya, lihat Driver logging fluentd di dokumentasi Docker.

Gunakan opsi ini ketika ada throughput tinggi, karena Docker mungkin kehabisan memori buffer dan membuang pesan buffer, sehingga dapat menambahkan pesan baru.

Pertimbangkan hal berikut saat menggunakan FireLens Amazon ECS dengan opsi batas buffer:

- Opsi ini didukung pada jenis EC2 peluncuran Amazon dan jenis peluncuran Fargate dengan versi platform `1.4.0` atau yang lebih baru.
- Opsi ini hanya valid ketika `logDriver` diatur `keawsfirelens`.
- Batas buffer default adalah baris `1048576` log.
- Batas buffer harus lebih besar dari atau sama dengan `0` dan kurang dari garis `536870912` log.
- Jumlah maksimum memori yang digunakan untuk buffer ini adalah produk dari ukuran setiap baris log dan ukuran buffer. Misalnya, jika baris log aplikasi rata-rata 2 KiB, batas buffer 4096 akan menggunakan paling banyak 8 MiB. Jumlah total memori yang dialokasikan pada tingkat tugas harus lebih besar dari jumlah memori yang dialokasikan untuk semua wadah selain buffer memori driver log.

Saat driver `awsfirelens` log ditentukan dalam definisi tugas, agen penampung Amazon ECS menyuntikkan variabel lingkungan berikut ke dalam wadah:

`FLUENT_HOST`

Alamat IP yang ditetapkan ke FireLens wadah.

> **Note**
>
> Jika Anda menggunakan tipe EC2 peluncuran dengan mode `bridge` jaringan, variabel `FLUENT_HOST` lingkungan dalam wadah aplikasi Anda dapat menjadi tidak akurat setelah restart wadah router FireLens log (wadah dengan `firelensConfiguration` objek dalam definisi kontainer). Ini karena `FLUENT_HOST` merupakan alamat IP dinamis dan dapat berubah setelah restart. Logging langsung dari wadah aplikasi ke alamat `FLUENT_HOST` IP dapat mulai gagal setelah alamat berubah. Untuk informasi selengkapnya tentang memulai ulang kontainer individual, lihat. Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer

`FLUENT_PORT`

Port tempat protokol Fluent Forward mendengarkan.

Anda dapat menggunakan variabel `FLUENT_HOST` dan `FLUENT_PORT` lingkungan untuk log langsung ke router log dari kode alih-alih melalui`stdout`. Untuk informasi lebih lanjut, lihat fluent-logger-golangdi GitHub.

Berikut ini menunjukkan sintaks untuk menentukan. `log-driver-buffer-limit` Ganti `my_service_` dengan nama layanan Anda:

```
{
    "containerDefinitions": [
        {
            "name": "my_service_log_router",
            "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:stable",
            "cpu": 0,
            "memoryReservation": 51,
            "portMappings": [],
            "essential": true,
            "environment": [],
```

```
            "mountPoints": [],
            "volumesFrom": [],
            "user": "0",
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/ecs-aws-firelens-sidecar-container",
                    "mode": "non-blocking",
                    "awslogs-create-group": "true",
                    "max-buffer-size": "25m",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "firelens"
                },
                "secretOptions": []
            },
            "systemControls": [],
            "firelensConfiguration": {
                "type": "fluentbit"
            }
        },
        {
            "essential": true,
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "51200"
                }
            },
            "dependsOn": [
                {
                    "containerName": "log_router",
                    "condition": "START"
                }
            ],
            "memoryReservation": 100
        }
    ]
}
```

AWS untuk repositori Fluent Bit gambar untuk Amazon ECS

AWS menyediakan Fluent Bit gambar dengan plugin untuk CloudWatch Log dan Firehose. Sebaiknya gunakan Fluent Bit sebagai router log Anda karena memiliki tingkat pemanfaatan sumber daya yang lebih rendah daripadaFluentd. Untuk informasi selengkapnya, lihat CloudWatch Log untuk Bit Lancar dan Amazon Kinesis Firehose for Fluent Bit.

Gambar AWS for Fluent Bit tersedia di Amazon ECR di Galeri Publik Amazon ECR dan di repositori Amazon ECR di sebagian besar untuk ketersediaan tinggi. Wilayah AWS

Galeri Publik Amazon ECR

Fluent BitGambar AWS untuk tersedia di Galeri Publik Amazon ECR. Ini adalah lokasi yang disarankan untuk mengunduh Fluent Bit gambar AWS for karena ini adalah repositori publik dan tersedia untuk digunakan dari semua. Wilayah AWS Untuk informasi lebih lanjut, lihat aws-for-fluent-bitdi Galeri Publik Amazon ECR.

Linux

AWS Untuk Fluent Bit gambar di Galeri Publik Amazon ECR mendukung sistem operasi Amazon Linux dengan`ARM 64`, atau `x86-64` arsitektur.

Anda dapat menarik Fluent Bit gambar AWS for dari Galeri Publik Amazon ECR dengan menentukan URL repositori dengan tag gambar yang diinginkan. Tag gambar yang tersedia dapat ditemukan di tab Tag gambar di Galeri Publik Amazon ECR.

Berikut ini menunjukkan sintaksis yang digunakan untuk Docker CLI.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

Misalnya, Anda dapat menarik stable terbaru AWS untuk Fluent Bit gambar menggunakan perintah CLI Docker ini.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:stable
```

> **Note**
>
> Penarikan yang tidak diautentikasi diperbolehkan, tetapi memiliki batas kecepatan yang lebih rendah daripada tarikan yang diautentikasi. Untuk mengautentikasi menggunakan AWS akun Anda sebelum menarik, gunakan perintah berikut.

```
aws ecr-public get-login-password --region us-east-1 | docker login --username
 AWS --password-stdin public.ecr.aws
```

Windows

Fluent BitGambar AWS for di Galeri Publik Amazon ECR mendukung AMD64 arsitektur dengan sistem operasi berikut:

- Windows Server 2022 Lengkap
- Windows Server 2022 Inti
- Windows Server 2019 Full
- Windows Server 2019 Core

Wadah Windows yang ada di AWS Fargate tidak mendukung. FireLens

Anda dapat menarik Fluent Bit gambar AWS for dari Galeri Publik Amazon ECR dengan menentukan URL repositori dengan tag gambar yang diinginkan. Tag gambar yang tersedia dapat ditemukan di tab Tag gambar di Galeri Publik Amazon ECR.

Berikut ini menunjukkan sintaksis yang digunakan untuk Docker CLI.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

Misalnya, Anda dapat menarik stable terbaru AWS untuk Fluent Bit gambar menggunakan perintah CLI Docker ini.

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:windowsservercore-
stable
```

> **Note**
>
> Penarikan yang tidak diautentikasi diperbolehkan, tetapi memiliki batas kecepatan yang lebih rendah daripada tarikan yang diautentikasi. Untuk mengautentikasi menggunakan AWS akun Anda sebelum menarik, gunakan perintah berikut.

```
aws ecr-public get-login-password --region us-east-1 | docker login --username
 AWS --password-stdin public.ecr.aws
```

Amazon ECR

Gambar AWS for Fluent Bit tersedia di Amazon ECR untuk ketersediaan tinggi. Gambar-gambar ini tersedia di sebagian besar Wilayah AWS, termasuk AWS GovCloud (US).

Linux

Stabil terbaru AWS untuk URI gambar Fluent Bit dapat diambil menggunakan perintah berikut.

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/stable \
      --region us-east-1
```

Semua versi image AWS for Fluent Bit dapat dicantumkan menggunakan perintah berikut untuk menanyakan parameter Systems Manager Parameter Store.

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit \
      --region us-east-1
```

Stabil terbaru AWS untuk image Fluent Bit dapat direferensikan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager. Berikut ini adalah contohnya:

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/stable
```

Windows

Stabil terbaru AWS untuk URI gambar Fluent Bit dapat diambil menggunakan perintah berikut.

```
aws ssm get-parameters \
```

```
      --names /aws/service/aws-for-fluent-bit/windowsservercore-stable \
      --region us-east-1
```

Semua versi image AWS for Fluent Bit dapat dicantumkan menggunakan perintah berikut untuk menanyakan parameter Systems Manager Parameter Store.

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit/windowsservercore \
      --region us-east-1
```

Stabil terbaru AWS untuk image Fluent Bit dapat direferensikan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager. Berikut ini adalah contohnya:

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/windowsservercore-stable
```

Contoh definisi tugas Amazon ECS: Rute log ke FireLens

Untuk menggunakan perutean log kustom denganFireLens, Anda harus menentukan yang berikut dalam definisi tugas Anda:

- Sebuah wadah log router yang berisi FireLens konfigurasi. Kami merekomendasikan bahwa kontainer ditandai sebagai `essential`.
- Satu kontainer aplikasi atau lebih yang berisi konfigurasi log menentukan driver log `awsfirelens`.
- Peran IAM tugas Amazon Resource Name (ARN) yang berisi izin yang diperlukan untuk tugas untuk merutekan log.

Saat membuat definisi tugas baru menggunakan AWS Management Console, ada bagian FireLens integrasi yang memudahkan untuk menambahkan wadah router log. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Amazon ECS mengonversi konfigurasi log dan menghasilkan konfigurasi output Fluentd atau Fluent Bit. Konfigurasi output dipasang dalam kontainer perutean log di `/fluent-bit/etc/fluent-bit.conf` untuk Fluent Bit dan `/fluentd/etc/fluent.conf` untuk Fluentd.

> **Important**
>
> FireLensmendengarkan di port`24224`. Oleh karena itu, untuk memastikan bahwa router FireLens log tidak dapat dijangkau di luar tugas, Anda tidak boleh mengizinkan lalu lintas masuk pada port `24224` dalam grup keamanan yang digunakan tugas Anda. Untuk tugas yang menggunakan mode `awsvpc` jaringan, ini adalah grup keamanan yang terkait dengan tugas. Untuk tugas yang menggunakan mode `host` jaringan, ini adalah grup keamanan yang terkait dengan EC2 instans Amazon yang menghosting tugas tersebut. Untuk tugas yang menggunakan mode `bridge` jaringan, jangan membuat pemetaan port apa pun yang menggunakan port. `24224`

Secara default, Amazon ECS menambahkan bidang tambahan di entri log Anda yang membantu mengidentifikasi sumber log.

- `ecs_cluster`— Nama cluster yang menjadi bagian dari tugas.
- `ecs_task_arn`— Nama Sumber Daya Amazon (ARN) lengkap dari tugas yang menjadi bagian dari wadah.
- `ecs_task_definition`— Nama definisi tugas dan revisi yang digunakan tugas.
- `ec2_instance_id`— ID EC2 instans Amazon tempat penampung di-host. Bidang ini hanya berlaku untuk tugas yang menggunakan tipe EC2 peluncuran.

Anda dapat mengatur `enable-ecs-log-metadata` ke `false` jika Anda tidak ingin metadata.

Contoh definisi tugas berikut mendefinisikan wadah router log yang menggunakan Bit Lancar untuk merutekan lognya ke Log. CloudWatch Ini juga mendefinisikan wadah aplikasi yang menggunakan konfigurasi log untuk merutekan log ke Amazon Data Firehose dan menetapkan memori yang digunakan untuk buffer peristiwa ke 2 MiB.

> **Note**
>
> Untuk contoh definisi tugas lainnya, lihat FireLenscontoh Amazon ECS di GitHub.

```
{
  "family": "firelens-example-firehose",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
```

```
  "containerDefinitions": [
    {
            "name": "log_router",
            "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:stable",
            "cpu": 0,
            "memoryReservation": 51,
            "portMappings": [],
            "essential": true,
            "environment": [],
            "mountPoints": [],
            "volumesFrom": [],
            "user": "0",
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/ecs-aws-firelens-sidecar-container",
                    "mode": "non-blocking",
                    "awslogs-create-group": "true",
                    "max-buffer-size": "25m",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "firelens"
                },
                "secretOptions": []
            },
            "systemControls": [],
            "firelensConfiguration": {
                "type": "fluentbit"
            }
        },
    {
      "essential": true,
      "image": "public.ecr.aws/docker/library/httpd:latest",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "firehose",
          "region": "us-west-2",
          "delivery_stream": "my-stream",
          "log-driver-buffer-limit": "2097152"
        }
      },
      "memoryReservation": 100
    }
```

```
  ]
}
```

Pasangan kunci-nilai yang ditentukan sebagai opsi dalam `logConfiguration` objek digunakan untuk menghasilkan konfigurasi Fluentd atau Fluent Bit output. Berikut ini adalah contoh kode dari definisi Fluent Bit output.

```
[OUTPUT]
    Name   firehose
    Match  app-firelens*
    region us-west-2
    delivery_stream my-stream
```

> **Note**
>
> FireLens mengelola `match` konfigurasi. Anda tidak menentukan `match` konfigurasi dalam definisi tugas Anda.

Gunakan file konfigurasi khusus

Anda dapat menentukan file konfigurasi khusus. Format file konfigurasi adalah format asli untuk router log yang Anda gunakan. Untuk informasi selengkapnya, lihat Sintaks File Konfigurasi Fluentd dan Konfigurasi YAMM.

Dalam file konfigurasi kustom Anda, untuk tugas yang menggunakan mode `awsvpc` jaringan `bridge` atau, jangan setel input forward Fluentd atau Fluent Bit melalui TCP karena FireLens menambahkannya ke konfigurasi input.

FireLensKonfigurasi Anda harus berisi opsi berikut untuk menentukan file konfigurasi kustom:

`config-file-type`

Lokasi sumber file konfigurasi kustom. Opsi yang tersedia adalah `s3` atau `file`.

> **Note**
>
> Tugas yang di-host AWS Fargate hanya mendukung jenis file `file` konfigurasi.

`config-file-value`

Sumber untuk file konfigurasi kustom. Jika jenis file `s3` konfigurasi digunakan, nilai file konfigurasi adalah ARN lengkap dari bucket dan file Amazon S3. Jika jenis file `file` konfigurasi digunakan, nilai file konfigurasi adalah jalur lengkap dari file konfigurasi yang ada baik dalam gambar kontainer atau pada volume yang dipasang di wadah.

> **Important**
>
> Saat menggunakan file konfigurasi khusus, Anda harus menentukan jalur yang berbeda dari yang FireLens digunakan. Amazon ECS mencadangkan jalur `/fluent-bit/etc/fluent-bit.conf` file untuk Fluent Bit dan Fluentd. `/fluentd/etc/fluent.conf`

Contoh berikut menunjukkan sintaksis yang diperlukan saat menentukan konfigurasi kustom.

> **Important**
>
> Untuk menentukan file konfigurasi kustom yang di-host di Amazon S3, pastikan Anda telah membuat peran IAM eksekusi tugas dengan izin yang tepat.

Berikut ini menunjukkan sintaks yang diperlukan saat menentukan konfigurasi kustom.

```
{
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "config-file-type": "s3 | file",
          "config-file-value": "arn:aws:s3:::amzn-s3-demo-bucket/fluent.conf
 | filepath"
        }
      }
    }
  ]
```

```
}
```

> **Note**
>
> Tugas yang dihosting AWS Fargate hanya mendukung jenis file `file` konfigurasi.

Menggunakan gambar AWS non-kontainer di Amazon ECS

Gunakan registri pribadi untuk menyimpan kredensyal Anda AWS Secrets Manager, dan kemudian mereferensikannya dalam definisi tugas Anda. Ini menyediakan cara untuk mereferensikan gambar kontainer yang ada di pendaftar pribadi di luar AWS yang memerlukan otentikasi dalam definisi tugas Anda. Fitur ini didukung oleh tugas yang dihosting di Fargate, EC2 instans Amazon, dan instans eksternal menggunakan Amazon ECS Anywhere.

> **Important**
>
> Jika definisi tugas Anda mereferensikan gambar yang disimpan di Amazon ECR, topik ini tidak berlaku. Untuk informasi selengkapnya, lihat Menggunakan Gambar Amazon ECR dengan Amazon ECS di Panduan Pengguna Amazon Elastic Container Registry.

Untuk tugas yang dihosting di EC2 instans Amazon, fitur ini memerlukan versi `1.19.0` atau yang lebih baru dari agen penampung. Akan tetapi, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.

Untuk tugas yang dihosting di Fargate, fitur ini memerlukan versi platform `1.2.0` atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.

Dalam definisi kontainer Anda, tentukan `repositoryCredentials` objek dengan detail rahasia yang Anda buat. Rahasia yang direferensikan dapat berasal dari akun yang berbeda Wilayah AWS atau berbeda dari tugas yang menggunakannya.

> **Note**
>
> Saat menggunakan Amazon ECS API, AWS CLI, atau AWS SDK, jika rahasia ada Wilayah AWS sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN

lengkap atau nama rahasia. Jika rahasia ada di akun yang berbeda, ARN penuh rahasia harus ditentukan. Saat menggunakan AWS Management Console, ARN penuh rahasia harus ditentukan selalu.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan parameter yang diperlukan:

Gantikan parameter berikut:

- *private-repo*dengan nama host repositori pribadi
- *private-image*dengan nama gambar
- *arn:aws:secretsmanager:region:aws_account_id:secret:secret_name*dengan nama sumber daya Amazon rahasia (ARN)

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

Note

Metode lain untuk mengaktifkan otentikasi registri pribadi menggunakan variabel lingkungan agen penampung Amazon ECS untuk mengautentikasi ke pendaftar pribadi. Metode ini hanya didukung untuk tugas yang dihosting di EC2 instans Amazon. Untuk informasi selengkapnya, lihat Mengonfigurasi instans penampung Amazon ECS untuk gambar Docker pribadi .

Untuk menggunakan registri pribadi

1. Definisi tugas harus memiliki peran eksekusi tugas. Hal ini mengizinkan agen kontainer untuk menarik citra kontainer. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Autentikasi registri pribadi memungkinkan tugas Amazon ECS Anda menarik gambar kontainer dari pendaftar pribadi di luar AWS (seperti Docker Hub, Quay.io, atau registri pribadi Anda sendiri) yang memerlukan kredensyal otentikasi. Fitur ini menggunakan Secrets Manager untuk menyimpan kredensyal registri Anda dengan aman, yang kemudian direferensikan dalam definisi tugas Anda menggunakan parameter. `repositoryCredentials`

Untuk informasi selengkapnya tentang mengonfigurasi autentikasi registri pribadi, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

Untuk menyediakan akses ke rahasia yang berisi kredensyal registri pribadi Anda, tambahkan izin berikut sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `secretsmanager:GetSecretValue`—Diperlukan untuk mengambil kredensi registri pribadi dari Secrets Manager.
- `kms:Decrypt`—Wajib hanya jika rahasia Anda menggunakan kunci KMS khusus dan bukan kunci default. Nama Sumber Daya Amazon (ARN) untuk kunci kustom Anda harus ditambahkan sebagai sumber daya.

Berikut ini adalah contoh kebijakan inline yang menambahkan izin.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [

 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

2. Gunakan AWS Secrets Manager untuk membuat rahasia untuk kredensyal registri pribadi Anda. Untuk informasi tentang cara membuat rahasia, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

   Masukkan kredensyal registri pribadi Anda menggunakan format berikut:

   ```
   {
     "username" : "privateRegistryUsername",
     "password" : "privateRegistryPassword"
   }
   ```

3. Mendaftarkan ketentuan tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.

Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer

Anda dapat mengaktifkan kebijakan restart untuk setiap wadah penting dan non-esensial yang ditentukan dalam definisi tugas Anda, untuk mengatasi kegagalan sementara lebih cepat dan menjaga ketersediaan tugas. Saat Anda mengaktifkan kebijakan restart untuk penampung, Amazon ECS dapat memulai ulang penampung jika keluar, tanpa perlu mengganti tugas.

Kebijakan restart tidak diaktifkan untuk kontainer secara default. Saat mengaktifkan kebijakan restart untuk kontainer, Anda dapat menentukan kode keluar tempat penampung tidak akan dimulai ulang. Ini bisa berupa kode keluar yang menunjukkan keberhasilan, seperti kode keluar`0`, yang tidak memerlukan restart. Anda juga dapat menentukan berapa lama kontainer harus berjalan dengan sukses sebelum restart dapat dicoba. Untuk informasi selengkapnya tentang parameter ini, lihat Mulai ulang kebijakan. Untuk contoh definisi tugas yang menentukan nilai-nilai ini, lihatMenentukan kebijakan restart container dalam definisi tugas Amazon ECS.

Anda dapat menggunakan titik akhir metadata tugas Amazon ECS atau CloudWatch Wawasan Kontainer untuk memantau berapa kali penampung dimulai ulang. Untuk informasi selengkapnya tentang titik akhir metadata tugas, lihat dan. Titik akhir metadata tugas Amazon ECS versi 4 Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate Untuk informasi selengkapnya tentang metrik Wawasan Kontainer untuk Amazon ECS, lihat metrik Amazon ECS Container Insights di Panduan Pengguna Amazon. CloudWatch

Kebijakan restart kontainer didukung oleh tugas yang dihosting di Fargate, EC2 instans Amazon, dan instans eksternal menggunakan Amazon ECS Anywhere.

Pertimbangan

Pertimbangkan hal berikut sebelum mengaktifkan kebijakan restart untuk penampung Anda:

- Kebijakan restart tidak didukung untuk wadah Windows di Fargate.
- Untuk tugas yang dihosting di EC2 instans Amazon, fitur ini memerlukan versi `1.86.0` atau yang lebih baru dari agen penampung. Akan tetapi, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.
- Untuk tugas yang dihosting di Fargate, fitur ini memerlukan versi platform `1.4.0` atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.
- Jika Anda menggunakan tipe EC2 peluncuran dengan mode `bridge` jaringan, variabel `FLUENT_HOST` lingkungan dalam wadah aplikasi Anda dapat menjadi tidak akurat setelah restart wadah router FireLens log (wadah dengan `firelensConfiguration` objek dalam definisi kontainer). Ini karena `FLUENT_HOST` merupakan alamat IP dinamis dan dapat berubah setelah restart. Logging langsung dari wadah aplikasi ke alamat `FLUENT_HOST` IP dapat mulai gagal setelah alamat berubah. Untuk informasi selengkapnya tentang `FLUENT_HOST`, lihat Mengkonfigurasi log Amazon ECS untuk throughput tinggi.
- Agen Amazon ECS menangani kebijakan restart kontainer. Jika karena alasan yang tidak terduga agen Amazon ECS gagal atau tidak lagi berjalan, penampung tidak akan dimulai ulang.
- Periode percobaan restart yang ditentukan dalam kebijakan Anda menentukan periode waktu (dalam detik) penampung harus dijalankan sebelum Amazon ECS memulai ulang penampung.

Menentukan kebijakan restart container dalam definisi tugas Amazon ECS

Untuk menentukan kebijakan restart untuk kontainer dalam definisi tugas, dalam definisi container, tentukan `restartPolicy` objek. Untuk informasi lebih lanjut tentang objek `restartPolicy`, lihat Mulai ulang kebijakan.

Berikut ini adalah definisi tugas menggunakan wadah Linux pada jenis peluncuran Fargate yang mengatur server web. Definisi kontainer menyertakan `restartPolicy` objek, dengan `enabled` disetel ke true untuk mengaktifkan kebijakan restart untuk penampung. Wadah harus berjalan selama 180 detik sebelum dapat dimulai ulang dan tidak akan dimulai ulang jika keluar dengan kode `0` keluar, yang menunjukkan keberhasilan.

```
{
  "containerDefinitions": [
```

```
    {
      "command": [
        "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
      ],
      "entryPoint": ["sh", "-c"],
      "essential": true,
      "image": "public.ecr.aws/docker/library/httpd:2.4",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fargate-task-definition",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "name": "sample-fargate-app",
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      }
    }
  ],
  "cpu": "256",
  "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
  "family": "fargate-task-definition",
  "memory": "512",
  "networkMode": "awsvpc",
  "runtimePlatform": {
    "operatingSystemFamily": "LINUX"
  },
  "requiresCompatibilities": ["FARGATE"]
```

```
}
```

Setelah Anda mendaftarkan definisi tugas dengan `restartPolicy` objek dalam definisi kontainer, Anda dapat menjalankan tugas atau membuat layanan dengan definisi tugas tersebut. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS dan Membuat penyebaran pembaruan bergulir Amazon ECS.

Meneruskan data sensitif ke wadah Amazon ECS

Anda dapat meneruskan data sensitif dengan aman, seperti kredensyal ke database, ke dalam wadah Anda.

Rahasia, seperti kunci API dan kredensyal basis data, sering digunakan oleh aplikasi untuk mendapatkan akses sistem lain. Mereka sering terdiri dari nama pengguna dan kata sandi, sertifikat, atau kunci API. Akses ke rahasia ini harus dibatasi pada prinsipal IAM tertentu yang menggunakan IAM dan disuntikkan ke dalam wadah saat runtime.

Rahasia dapat disuntikkan dengan mulus ke dalam wadah dari AWS Secrets Manager dan Amazon EC2 Systems Manager Parameter Store. Rahasia-rahasia ini dapat direferensikan dalam tugas Anda sebagai salah satu dari berikut ini.

1. Mereka direferensikan sebagai variabel lingkungan yang menggunakan parameter definisi `secrets` wadah.
2. Mereka direferensikan `secretOptions` seolah-olah platform logging Anda memerlukan otentikasi. Untuk informasi selengkapnya, lihat opsi konfigurasi logging.
3. Mereka direferensikan sebagai rahasia yang ditarik oleh gambar yang menggunakan parameter definisi `repositoryCredentials` wadah jika registri tempat penampung ditarik memerlukan otentikasi. Gunakan metode ini saat menarik gambar dari Galeri Publik Amazon ECR. Untuk informasi selengkapnya, lihat Autentikasi registri pribadi untuk tugas.

Kami menyarankan Anda melakukan hal berikut saat mengatur manajemen rahasia.

Gunakan AWS Secrets Manager atau AWS Systems Manager Parameter Menyimpan untuk menyimpan materi rahasia

Anda harus menyimpan kunci API, kredensyal database, dan materi rahasia lainnya dengan aman di Secrets Manager atau sebagai parameter terenkripsi di Systems Manager Parameter Store. Layanan ini serupa karena keduanya merupakan penyimpanan nilai kunci terkelola yang digunakan AWS KMS

untuk mengenkripsi data sensitif. Secrets Manager, bagaimanapun, juga mencakup kemampuan untuk secara otomatis memutar rahasia, menghasilkan rahasia acak, dan berbagi rahasia di seluruh akun. Untuk memanfaatkan fitur-fitur ini, gunakan Secrets Manager. Jika tidak, gunakan parameter terenkripsi di Systems Manager Parameter Store.

> **Important**
>
> Jika rahasia Anda berubah, Anda harus memaksa penerapan baru atau meluncurkan tugas baru untuk mengambil nilai rahasia terbaru. Untuk informasi selengkapnya, lihat topik berikut:
>
> - Tugas - Hentikan tugas, lalu mulailah. Untuk informasi selengkapnya, lihat Menghentikan tugas Amazon ECS dan Menjalankan aplikasi sebagai tugas Amazon ECS.
> - Layanan - Perbarui layanan dan gunakan opsi force new deployment. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Mengambil data dari bucket Amazon S3 terenkripsi

Anda harus menyimpan rahasia di bucket Amazon S3 terenkripsi dan menggunakan peran tugas untuk membatasi akses ke rahasia tersebut. Ini mencegah nilai variabel lingkungan bocor secara tidak sengaja di log dan terungkap saat dijalankan. `docker inspect` Ketika Anda melakukan ini, aplikasi Anda harus ditulis untuk membaca rahasia dari ember Amazon S3. Untuk petunjuknya, lihat Menyetel perilaku enkripsi sisi server default untuk bucket Amazon S3.

Pasang rahasia ke volume menggunakan wadah sespan

Karena ada peningkatan risiko kebocoran data dengan variabel lingkungan, Anda harus menjalankan wadah sespan yang membaca rahasia Anda AWS Secrets Manager dan menuliskannya ke volume bersama. Kontainer ini dapat berjalan dan keluar sebelum penampung aplikasi dengan menggunakan pemesanan kontainer Amazon ECS. Ketika Anda melakukan ini, wadah aplikasi kemudian memasang volume tempat rahasia itu ditulis. Seperti metode bucket Amazon S3, aplikasi Anda harus ditulis untuk membaca rahasia dari volume bersama. Karena volume tercakup pada tugas, volume secara otomatis dihapus setelah tugas berhenti. Sebagai contoh, lihat proyek task-def.json.

Di Amazon EC2, volume tempat rahasia ditulis dapat dienkripsi dengan kunci yang dikelola AWS KMS pelanggan. AWS Fargate Aktif, penyimpanan volume secara otomatis dienkripsi menggunakan kunci yang dikelola layanan.

Meneruskan variabel lingkungan individual ke wadah Amazon ECS

> **Important**
>
> Sebaiknya simpan data sensitif Anda baik dalam parameter AWS Secrets Manager rahasia atau AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.
> Variabel lingkungan yang ditentukan dalam definisi tugas dapat dibaca oleh semua pengguna dan peran yang diizinkan `DescribeTaskDefinition` tindakan untuk definisi tugas.

Anda dapat meneruskan variabel lingkungan ke kontainer Anda dengan cara berikut:

- Secara individual menggunakan parameter ketentuan kontainer `environment`. Ini memetakan ke `--env` opsi untuk docker container run.
- Secara massal, menggunakan parameter definisi `environmentFiles` kontainer untuk mencantumkan satu atau lebih file yang berisi variabel lingkungan. File harus di-host di Amazon S3. Ini memetakan ke `--env-file` opsi untuk docker run.

Berikut ini adalah potongan ketentuan tugas yang menunjukkan cara menentukan variabel lingkungan individual.

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environment": [
                {
                    "name": "variable",
                    "value": "value"
                }
            ],
            ...
        }
    ],
    ...
```

```
}
```

Meneruskan variabel lingkungan ke wadah Amazon ECS

> **Important**
>
> Sebaiknya simpan data sensitif Anda baik dalam parameter AWS Secrets Manager rahasia atau AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.
> File variabel lingkungan adalah objek di Amazon S3 dan semua pertimbangan keamanan Amazon S3 berlaku.
> Anda tidak dapat menggunakan `environmentFiles` parameter pada wadah Windows dan wadah Windows di Fargate.

Anda dapat membuat file variabel lingkungan dan menyimpannya di Amazon S3 untuk meneruskan variabel lingkungan ke wadah Anda.

Dengan menentukan variabel lingkungan dalam file, Anda dapat menyuntikkan variabel lingkungan massal. Dalam definisi container Anda, tentukan `environmentFiles` objek dengan daftar bucket Amazon S3 yang berisi file variabel lingkungan Anda.

Amazon ECS tidak memberlakukan batas ukuran pada variabel lingkungan, tetapi file variabel lingkungan yang besar mungkin mengisi ruang disk. Setiap tugas yang menggunakan file variabel lingkungan menyebabkan salinan file diunduh ke disk. Amazon ECS menghapus file sebagai bagian dari pembersihan tugas.

Untuk informasi tentang variabel lingkungan yang didukung, lihat Parameter definisi kontainer lanjutan- Lingkungan.

Pertimbangkan hal berikut saat menentukan file variabel lingkungan dalam definisi wadah.

- Untuk tugas Amazon ECS di Amazon EC2, instans penampung Anda mengharuskan agen penampung adalah versi `1.39.0` atau yang lebih baru untuk menggunakan fitur ini. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS.
- Untuk tugas Amazon ECS di AWS Fargate, tugas Anda harus menggunakan `1.4.0` versi platform atau yang lebih baru (Linux) untuk menggunakan fitur ini. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.

Verifikasi bahwa variabel didukung untuk platform sistem operasi. Untuk informasi selengkapnya, lihat the section called “Definisi kontainer” dan the section called “Parameter ketentuan tugas lainnya”.

- File harus menggunakan ekstensi file `.env` dan pengodean UTF-8.
- Peran eksekusi tugas diperlukan untuk menggunakan fitur ini dengan izin tambahan untuk Amazon S3. Ini memungkinkan agen penampung untuk menarik file variabel lingkungan dari Amazon S3. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
- Ada batas 10 file per definisi tugas.
- Setiap baris dalam file lingkungan harus berisi variabel lingkungan dalam `VARIABLE=VALUE` format. Spasi atau tanda kutip disertakan sebagai bagian dari nilai untuk file Amazon ECS. Baris yang diawali dengan # diperlakukan sebagai komentar dan diabaikan. Untuk informasi selengkapnya tentang sintaks file variabel lingkungan, lihat Mengatur variabel lingkungan (-e, --env, --env-file) dalam dokumentasi. Docker

  Berikut ini adalah sintaks yang sesuai.

  ```
  #This is a comment and will be ignored
  VARIABLE=VALUE
  ENVIRONMENT=PRODUCTION
  ```

- Jika ada variabel lingkungan yang ditentukan menggunakan parameter `environment` dalam ketentuan kontainer, maka akan lebih diutamakan daripada variabel yang terkandung dalam file lingkungan.
- Jika beberapa file lingkungan ditentukan dan mereka berisi variabel yang sama, mereka diproses dalam urutan entri. Ini berarti bahwa nilai pertama dari variabel digunakan dan nilai-nilai berikutnya dari variabel duplikat diabaikan. Kami merekomendasikan agar Anda menggunakan nama variabel yang unik.
- Jika file lingkungan ditentukan sebagai penimpaan kontainer, itu digunakan. Selain itu, file lingkungan lain yang ditentukan dalam definisi wadah diabaikan.
- Aturan berikut berlaku untuk jenis peluncuran Fargate:
  - File ini ditangani mirip dengan env-file Docker asli.
  - Definisi kontainer yang mereferensikan variabel lingkungan yang kosong dan disimpan di Amazon S3 tidak muncul di wadah.
  - Tidak ada dukungan untuk penanganan pelarian shell.
  - Titik masuk kontainer menyelingi nilai-nilai. `VARIABLE`

Contoh

Berikut ini adalah potongan ketentuan tugas yang menunjukkan cara menentukan file variabel lingkungan.

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environmentFiles": [
                {
                    "value": "arn:aws:s3:::amzn-s3-demo-
bucket/envfile_object_name.env",
                    "type": "s3"
                }
            ],
            ...
        }
    ],
    ...
}
```

Lulus rahasia Secrets Manager secara terprogram di Amazon ECS

Alih-alih hardcoding informasi sensitif dalam teks biasa di aplikasi Anda, Anda dapat menggunakan Secrets Manager untuk menyimpan data sensitif.

Kami merekomendasikan metode ini untuk mengambil data sensitif karena jika rahasia Secrets Manager kemudian diperbarui, aplikasi secara otomatis mengambil versi terbaru dari rahasia.

Buat rahasia di Secrets Manager. Setelah Anda membuat rahasia Secrets Manager, perbarui kode aplikasi Anda untuk mengambil rahasia.

Tinjau pertimbangan berikut sebelum mengamankan data sensitif di Secrets Manager.

- Hanya rahasia yang menyimpan data teks, yang merupakan rahasia yang dibuat dengan `SecretString` parameter CreateSecretAPI, yang didukung. Rahasia yang menyimpan data biner, yang merupakan rahasia yang dibuat dengan `SecretBinary` parameter CreateSecretAPI tidak didukung.

- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan. Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi tentang titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna.AWS Secrets Manager
- VPC yang digunakan tugas Anda harus menggunakan resolusi DNS.
- Definisi tugas Anda harus menggunakan peran tugas dengan izin tambahan untuk Secrets Manager. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Buat rahasia Secrets Manager

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Untuk informasi tentang cara membuat rahasia, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Perbarui aplikasi Anda untuk mengambil rahasia Secrets Manager secara terprogram

Anda dapat mengambil rahasia dengan panggilan ke Secrets Manager APIs langsung dari aplikasi Anda. Untuk selengkapnya, lihat Mengambil rahasia dari AWS Secrets Manager Panduan AWS Secrets Manager Pengguna.

Untuk mengambil data sensitif yang disimpan di dalam AWS Secrets Manager, lihat Contoh kode untuk AWS Secrets Manager digunakan AWS SDKs di Pustaka Kode Contoh Kode AWS SDK.

Pass Systems Manager Parameter Simpan rahasia secara terprogram di Amazon ECS

Systems Manager Parameter Store menyediakan penyimpanan dan pengelolaan rahasia yang aman. Anda dapat menyimpan data seperti password, string database, EC2 instance IDs dan AMI IDs, dan kode lisensi sebagai nilai parameter, bukan hardcoding informasi ini dalam aplikasi Anda. Anda dapat menyimpan nilai sebagai teks biasa atau data terenkripsi.

Kami merekomendasikan metode ini untuk mengambil data sensitif karena jika parameter Systems Manager Parameter Store kemudian diperbarui, aplikasi secara otomatis mengambil versi terbaru.

Tinjau pertimbangan berikut sebelum mengamankan data sensitif di Systems Manager Parameter Store.

- Hanya rahasia yang menyimpan data teks yang didukung. Rahasia yang menyimpan data biner tidak didukung.
- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan.
- VPC yang digunakan tugas Anda harus menggunakan resolusi DNS.

- Untuk tugas yang menggunakan tipe EC2 peluncuran, Anda harus menggunakan variabel konfigurasi agen Amazon ECS `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` untuk menggunakan fitur ini. Anda dapat menambahkannya ke file `/etc/ecs/ecs.config` selama pembuatan instans kontainer atau Anda dapat menambahkannya ke instans yang ada, lalu memulai ulang agen ECS. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Definisi tugas Anda harus menggunakan peran tugas dengan izin tambahan untuk Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Buat parameter

Anda dapat menggunakan konsol Systems Manager untuk membuat parameter Systems Manager Parameter Store untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat parameter Systems Manager (konsol) atau Membuat parameter Systems Manager (AWS CLI) di Panduan AWS Systems Manager Pengguna.

Perbarui aplikasi Anda untuk mengambil rahasia Systems Manager Parameter Store secara terprogram

Untuk mengambil data sensitif yang disimpan dalam parameter Systems Manager Parameter Store, lihat Contoh kode untuk Systems Manager yang digunakan AWS SDKs di AWS SDK Code Examples Code Library.

Lulus rahasia Secrets Manager melalui variabel lingkungan Amazon ECS

Ketika Anda menyuntikkan rahasia sebagai variabel lingkungan, Anda dapat menentukan isi lengkap rahasia, kunci JSON tertentu dalam rahasia. Hal ini membantu Anda mengontrol data sensitif yang diekspos ke kontainer Anda. Untuk informasi selengkapnya tentang pembuatan versi rahasia, lihat Apa yang ada di rahasia Secrets Manager? dalam AWS Secrets Manager User Guide.

Berikut ini harus dipertimbangkan saat menggunakan variabel lingkungan untuk menyuntikkan rahasia Secrets Manager ke dalam wadah.

- Data sensitif disuntikkan ke dalam kontainer Anda ketika kontainer awalnya dimulai. Jika rahasia kemudian diperbarui atau diputar, kontainer tidak akan menerima nilai yang diperbarui secara otomatis. Anda harus menjalankan tugas baru atau jika tugas Anda adalah bagian dari layanan Anda dapat memperbarui layanan dan menggunakan Memaksa deployment baru untuk memaksa layanan untuk meluncurkan tugas baru.

- Aplikasi yang berjalan pada container dan container log dan debugging tools memiliki akses ke variabel lingkungan.
- Untuk tugas Amazon ECS aktif AWS Fargate, pertimbangkan hal berikut:
  - Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi `1.3.0` platform atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.
  - Untuk menyuntikkan kunci JSON tertentu atau versi rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` (Windows). Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.
- Untuk tugas Amazon ECS aktif EC2, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan rahasia menggunakan kunci JSON atau versi rahasia tertentu, instans kontainer Anda harus memiliki versi `1.37.0` agen kontainer atau yang lebih baru. Namun, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

    Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau untuk menyuntikkan rahasia dalam konfigurasi log, instans kontainer Anda harus memiliki versi `1.22.0` agen kontainer atau yang lebih baru.
- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan dan terhubung ke Secrets Manager melalui subnet pribadi. Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi tentang titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna.AWS Secrets Manager Untuk informasi selengkapnya tentang menggunakan Secrets Manager dan Amazon VPC, lihat Cara menyambung ke layanan Secrets Manager dalam Amazon VPC.
- Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Gunakan sintaks berikut:

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

- Definisi tugas Anda harus menggunakan peran eksekusi tugas dengan izin tambahan untuk Secrets Manager. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Buat AWS Secrets Manager rahasianya

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Tambahkan variabel lingkungan ke definisi kontainer

Dalam ketentuan kontainer Anda, Anda dapat menentukan hal berikut:

- Objek `secrets` yang berisi nama variabel lingkungan untuk diatur dalam kontainer
- Amazon Resource Name (ARN) rahasia Secrets Manager
- Parameter tambahan yang berisi data sensitif untuk hadir ke kontainer

Contoh berikut menunjukkan sintaks lengkap yang harus ditentukan untuk rahasia Secrets Manager.

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

Bagian berikut menjelaskan parameter tambahan. Parameter ini opsional, tetapi jika Anda tidak menggunakannya, Anda harus menyertakan titik dua `:` untuk menggunakan nilai default. Contoh disediakan di bawah ini untuk konteks yang lebih.

`json-key`

Menentukan nama kunci dalam pasangan kunci-nilai dengan nilai yang ingin Anda tetapkan sebagai nilai variabel lingkungan. Hanya nilai dalam format JSON yang didukung. Jika Anda tidak menentukan kunci JSON, maka konten penuh dari rahasia tersebut akan digunakan.

`version-stage`

Menentukan label pementasan versi rahasia yang ingin Anda gunakan. Jika label pementasan versi ditentukan, Anda tidak dapat menentukan ID versi. Jika tidak ada tahap versi yang ditentukan, perilaku default adalah untuk mengambil rahasia dengan label pementasan `AWSCURRENT`.

Label tahapan digunakan untuk melacak berbagai versi rahasia saat diperbarui atau dirotasi. Setiap versi rahasia memiliki satu atau beberapa label tahapan dan satu ID.

`version-id`

Menentukan pengenal unik dari versi rahasia yang ingin Anda gunakan. Jika ID versi ditentukan, Anda tidak dapat menentukan label pementasan versi. Jika tidak ada ID versi yang ditentukan, perilaku default adalah untuk mengambil rahasia dengan label pementasan `AWSCURRENT`.

Versi IDs digunakan untuk melacak versi rahasia yang berbeda ketika mereka diperbarui atau diputar. Setiap versi rahasia memiliki satu ID. Untuk informasi selengkapnya, lihat Syarat dan Konsep Utama AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna.

Contoh ketentuan kontainer

Contoh berikut menunjukkan cara Anda dapat mereferensikan rahasia Secrets Manager dalam definisi container Anda.

Example mereferensikan rahasia penuh

Berikut ini adalah cuplikan ketentuan tugas yang menunjukkan format ketika mereferensikan teks lengkap dari rahasia Secrets Manager.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

Untuk mengakses nilai rahasia ini dari dalam wadah, Anda perlu memanggil file`$environment_variable_name`.

Example mereferensikan rahasia lengkap

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan teks lengkap dari beberapa rahasia Secrets Manager.

```
{
  "containerDefinitions": [{
     "secrets": [
      {
        "name": "environment_variable_name1",
         "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      },
      {
        "name": "environment_variable_name2",
         "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
abcdef"
      },
      {
        "name": "environment_variable_name3",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
ABCDEF"
      }
    ]
  }]
}
```

Untuk mengakses nilai rahasia ini dari dalam wadah, Anda perlu memanggil`$environment_variable_name1`,`$environment_variable_name2`, dan`$environment_variable_name3`.

Example mereferensikan kunci tertentu dalam sebuah rahasia

Berikut ini menunjukkan contoh output dari get-secret-valueperintah yang menampilkan isi rahasia bersama dengan label pementasan versi dan ID versi yang terkait dengannya.

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981ddEXAMPLE",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

Referensikan kunci tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf:username1::"
    }]
  }]
}
```

Example mereferensikan versi rahasia tertentu

Berikut ini menunjukkan contoh output dari perintah describe-secret yang menampilkan konten rahasia yang tidak terenkripsi bersama dengan metadata untuk semua versi rahasia.

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981ddEXAMPLE": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE": [
            "AWSPREVIOUS"
        ]
    }
}
```

Referensikan label tahapan versi tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
  "containerDefinitions": [{
    "secrets": [{
```

```
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

Referensikan ID versi tertentu dari output sebelumnya dalam ketentuan kontainer dengan menentukan nama kunci di akhir ARN.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Example mereferensikan kunci dan label tahapan versi tertentu dari suatu rahasia

Bagian berikut menunjukkan cara mereferensikan kunci tertentu dalam rahasia dan label tahapan versi tertentu.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

Untuk menentukan kunci dan versi ID tertentu, gunakan sintaks berikut.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
```

```
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Untuk informasi tentang cara membuat definisi tugas dengan rahasia yang ditentukan dalam variabel lingkungan, lihatMembuat definisi tugas Amazon ECS menggunakan konsol.

Lulus parameter Systems Manager melalui variabel lingkungan Amazon ECS

Amazon ECS memungkinkan Anda untuk menyuntikkan data sensitif ke dalam container Anda dengan menyimpan data sensitif Anda dalam AWS Systems Manager parameter Parameter Store dan kemudian mereferensikannya dalam definisi container Anda.

Pertimbangkan hal berikut saat menggunakan variabel lingkungan untuk menyuntikkan rahasia Systems Manager ke dalam wadah.

- Data sensitif disuntikkan ke dalam kontainer Anda ketika kontainer awalnya dimulai. Jika rahasia kemudian diperbarui atau diputar, kontainer tidak akan menerima nilai yang diperbarui secara otomatis. Anda harus menjalankan tugas baru atau jika tugas Anda adalah bagian dari layanan Anda dapat memperbarui layanan dan menggunakan Memaksa deployment baru untuk memaksa layanan untuk meluncurkan tugas baru.
- Untuk tugas Amazon ECS aktif AWS Fargate, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi `1.3.0` platform atau yang lebih baru. Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.
  - Untuk menyuntikkan kunci JSON tertentu atau versi rahasia sebagai variabel lingkungan atau dalam konfigurasi log, Anda harus menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` (Windows). Untuk informasi, lihat Versi platform Fargate untuk Amazon ECS.
- Untuk tugas Amazon ECS aktif EC2, hal-hal berikut harus dipertimbangkan:
  - Untuk menyuntikkan rahasia menggunakan kunci JSON atau versi rahasia tertentu, instans kontainer Anda harus memiliki versi `1.37.0` agen kontainer atau yang lebih baru. Namun, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Untuk menyuntikkan konten penuh rahasia sebagai variabel lingkungan atau untuk menyuntikkan rahasia dalam konfigurasi log, instans kontainer Anda harus memiliki versi `1.22.0` agen kontainer atau yang lebih baru.

- Gunakan titik akhir VPC antarmuka untuk meningkatkan kontrol keamanan. Anda harus membuat titik akhir VPC antarmuka untuk Systems Manager. Untuk informasi tentang titik akhir VPC, lihat Meningkatkan keamanan EC2 instans dengan menggunakan titik akhir VPC untuk Systems Manager di Panduan Pengguna.AWS Systems Manager
- Definisi tugas Anda harus menggunakan peran eksekusi tugas dengan izin tambahan untuk Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
- Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Gunakan sintaks berikut:

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

Buat parameter Systems Manager

Anda dapat menggunakan konsol Systems Manager untuk membuat parameter Systems Manager Parameter Store untuk data sensitif Anda. Untuk informasi selengkapnya, lihat Membuat parameter Systems Manager (konsol) atau Membuat parameter Systems Manager (AWS CLI) di Panduan AWS Systems Manager Pengguna.

Tambahkan variabel lingkungan ke definisi kontainer

Dalam definisi kontainer Anda dalam definisi tugas, tentukan `secrets` dengan nama variabel lingkungan yang akan disetel dalam wadah dan ARN lengkap dari parameter Systems Manager Parameter Store yang berisi data sensitif yang akan ditampilkan ke wadah. Untuk informasi selengkapnya, lihat `secrets`.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan parameter Systems Manager Parameter Store. Jika parameter Systems Manager Parameter Store ada di

Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama parameter. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

Untuk informasi tentang cara membuat definisi tugas dengan rahasia yang ditentukan dalam variabel lingkungan, lihatMembuat definisi tugas Amazon ECS menggunakan konsol.

Perbarui aplikasi Anda untuk mengambil rahasia Systems Manager Parameter Store secara terprogram

Untuk mengambil data sensitif yang disimpan dalam parameter Systems Manager Parameter Store, lihat Contoh kode untuk Systems Manager yang digunakan AWS SDKs di AWS SDK Code Examples Code Library.

Lulus rahasia untuk konfigurasi logging Amazon ECS

Anda dapat menggunakan `secretOptions` parameter `logConfiguration` untuk meneruskan data sensitif yang digunakan untuk logging.

Anda dapat menyimpan rahasia di Secrets Manager atau Systems Manager.

Gunakan Secrets Manager

Dalam definisi kontainer Anda, saat menentukan a, `logConfiguration` Anda dapat menentukan `secretOptions` dengan nama opsi driver log untuk disetel dalam wadah dan ARN lengkap dari rahasia Secrets Manager yang berisi data sensitif untuk disajikan ke wadah. Untuk informasi selengkapnya tentang membuat rahasia, lihat Membuat AWS Secrets Manager.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan rahasia Secrets Manager.

```
{
  "containerDefinitions": [{
```

```
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://your_splunk_instance:8088"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

Tambahkan variabel lingkungan ke definisi kontainer

Dalam ketentuan kontainer Anda, tentukan `secrets` dengan nama variabel lingkungan yang akan ditetapkan dalam kontainer dan ARN lengkap parameter Systems Manager Parameter Store yang berisi data sensitif untuk diberikan ke kontainer. Untuk informasi selengkapnya, lihat `secrets`.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan parameter Systems Manager Parameter Store. Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama parameter. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

Untuk informasi tentang cara membuat definisi tugas dengan rahasia yang ditentukan dalam variabel lingkungan, lihatMembuat definisi tugas Amazon ECS menggunakan konsol.

Gunakan Systems Manager

Anda dapat menyuntikkan data sensitif dalam konfigurasi log. Dalam ketentuan kontainer Anda, ketika menentukan `logConfiguration`, Anda dapat menentukan `secretOptions` dengan nama

opsi driver log yang kan ditetapkan dalam kontainer dan ARN lengkap parameter Systems Manager Parameter Store yang berisi data sensitif untuk diberikan ke kontainer.

> **Important**
>
> Jika parameter Systems Manager Parameter Store ada di Region yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama parameter. Jika parameter ada di Wilayah yang berbeda, maka tentukan ARN lengkap.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan format saat mereferensikan parameter Systems Manager Parameter Store.

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:/parameter_name"
      }]
    }]
  }]
}
```

Menentukan data sensitif menggunakan rahasia Secrets Manager di Amazon ECS

Amazon ECS memungkinkan Anda untuk menyuntikkan data sensitif ke dalam container Anda dengan menyimpan data sensitif Anda dalam AWS Secrets Manager rahasia dan kemudian mereferensikannya dalam definisi container Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

Pelajari cara membuat rahasia Secrets Manager, mereferensikan rahasia dalam definisi tugas Amazon ECS, lalu verifikasi itu berhasil dengan menanyakan variabel lingkungan di dalam wadah yang menunjukkan konten rahasia.

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- Pengguna Anda memiliki izin IAM yang diperlukan untuk membuat Secrets Manager dan sumber daya Amazon ECS.

Langkah 1: Buat rahasia Secrets Manager

Anda dapat menggunakan konsol Secrets Manager untuk membuat rahasia untuk data sensitif Anda. Dalam tutorial ini kita akan menciptakan rahasia dasar untuk menyimpan nama pengguna dan kata sandi untuk mereferensi nanti dalam sebuah kontainer. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

key/value Pasangan yang akan disimpan dalam rahasia ini adalah nilai variabel lingkungan dalam wadah Anda di akhir tutorial.

Simpan ARN Rahasia untuk referensi dalam kebijakan IAM eksekusi tugas Anda dan definisi tugas di langkah selanjutnya.

Langkah 2: Tambahkan izin rahasia ke peran eksekusi tugas

Agar Amazon ECS dapat mengambil data sensitif dari rahasia Secrets Manager Anda, Anda harus memiliki izin rahasia untuk peran eksekusi tugas. Untuk informasi selengkapnya, lihat Izin Secrets Manager atau Systems Manager.

Langkah 3: Membuat sebuah penetapan tugas

Anda dapat menggunakan konsol Amazon ECS untuk membuat definisi tugas yang mereferensikan rahasia Secrets Manager.

Untuk membuat ketentuan tugas yang menetapkan rahasia

Gunakan konsol IAM untuk memperbarui peran eksekusi tugas Anda dengan izin yang diperlukan.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.

4. Di kotak editor JSON, masukkan teks JSON definisi tugas berikut, pastikan Anda menentukan ARN lengkap rahasia Secrets Manager yang Anda buat di langkah 1 dan peran eksekusi tugas yang Anda perbarui di langkah 2. Pilih Simpan.

5.

```
{
    "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "secrets": [
                {
                    "valueFrom":
 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value",
                    "name": "username_value"
                }
            ],
            "memory": 300,
            "image": "public.ecr.aws/docker/library/httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

6. Pilih Buat.

Langkah 4: Buat cluster

Anda dapat menggunakan konsol Amazon ECS untuk membuat klaster yang berisi instance container untuk menjalankan tugas. Jika Anda memiliki klaster yang ada dengan setidaknya satu instans kontainer yang terdaftar untuk itu dengan sumber daya yang tersedia untuk menjalankan satu instans dari ketentuan tugas yang dibuat untuk tutorial ini Anda dapat melompat ke langkah berikutnya.

Untuk tutorial ini kita akan membuat cluster dengan satu instance `t2.micro` container menggunakan Amazon ECS Amazon Linux 2 AMI yang dioptimalkan Amazon.

Untuk informasi tentang cara membuat klaster untuk jenis EC2 peluncuran, lihatthe section called “Membuat cluster untuk jenis EC2 peluncuran Amazon”.

Langkah 5: Jalankan tugas

Anda dapat menggunakan konsol Amazon ECS untuk menjalankan tugas menggunakan definisi tugas yang Anda buat. Untuk tutorial ini kita akan menjalankan tugas menggunakan tipe EC2 peluncuran, menggunakan cluster yang kita buat pada langkah sebelumnya.

Untuk informasi tentang cara menjalankan tugas, lihatthe section called “Menjalankan aplikasi sebagai tugas”.

Langkah 6: Verifikasi

Anda dapat memverifikasi bahwa semua langkah berhasil diselesaikan dan variabel lingkungan dibuat dengan benar dalam kontainer Anda menggunakan langkah-langkah berikut.

Untuk memverifikasi bahwa variabel lingkungan telah dibuat

1. Cari IP publik atau alamat DNS untuk instans kontainer Anda.
    a. Buka konsol di https://console.aws.amazon.com/ecs/v2.
    b. Di panel navigasi, pilih Cluster, lalu pilih cluster yang Anda buat.
    c. Pilih Infrastruktur, lalu pilih instance container.
    d. Rekam IP Publik atau DNS Publik untuk instans Anda.
2. Jika Anda menggunakan komputer macOS atau Linux, sambungkan ke instans Anda dengan perintah berikut, ganti jalur ke kunci pribadi dan alamat publik untuk instans Anda:

```
$ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
```

Untuk informasi selengkapnya tentang menggunakan komputer Windows, lihat Connect ke instans Linux menggunakan PuTTY di EC2 Panduan Pengguna Amazon.

> **Important**
>
> Untuk informasi selengkapnya tentang masalah apa pun saat menyambung ke instans, lihat Memecahkan Masalah Menyambung ke Instans Anda di EC2 Panduan Pengguna Amazon.

3. Cantumkan kontainer yang berjalan pada instans. Catat ID kontainer untuk kontainer `ecs-secrets-tutorial`.

```
docker ps
```

4. Terhubung ke kontainer `ecs-secrets-tutorial` menggunakan ID kontainer dari output dari langkah sebelumnya.

```
docker exec -it container_ID /bin/bash
```

5. Gunakan perintah `echo` untuk mencetak nilai variabel lingkungan.

```
echo $username_value
```

Jika tutorial berhasil, Anda akan melihat output sebagai berikut:

```
password_value
```

> **Note**
>
> Atau, Anda dapat mencantumkan semua variabel lingkungan dalam wadah Anda menggunakan perintah `env` (atau `printenv`).

Langkah 7: Bersihkan

Setelah selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari timbulnya biaya untuk sumber daya yang tidak terpakai.

Untuk membersihkan sumber daya

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pilih Hapus klaster.
5. Di kotak konfirmasi, masukkan hapus ***cluster name***, lalu pilih Hapus.
6. Buka konsol IAM di https://console.aws.amazon.com/iam/.
7. Di panel navigasi, pilih Peran.
8. Cari daftar peran untuk `ecsTaskExecutionRole` dan pilih peran itu.
9. Pilih Izin, lalu pilih X di sebelah ECSSecretsTutorial. Pilih Hapus.
10. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
11. Pilih rahasia username_value yang Anda buat dan pilih Tindakan, Hapus rahasia.

Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate

Definisi tugas dibagi menjadi beberapa bagian terpisah: keluarga tugas, peran tugas AWS Identity and Access Management (IAM), mode jaringan, definisi wadah, volume, dan jenis peluncuran. Definisi keluarga dan wadah diperlukan dalam definisi tugas. Sebaliknya, peran tugas, mode jaringan, volume, dan jenis peluncuran bersifat opsional.

Anda dapat menggunakan parameter ini dalam file JSON untuk mengonfigurasi definisi tugas Anda.

Berikut ini adalah deskripsi yang lebih rinci untuk setiap parameter definisi tugas untuk jenis peluncuran Fargate.

Rangkaian

`family`

Tipe: String

Diperlukan: Ya

Ketika Anda mendaftarkan ketentuan tugas, Anda memberikan sebuah famili, yang mirip dengan nama untuk beberapa versi ketentuan tugas, yang ditentukan dengan nomor revisi. Definisi tugas pertama yang terdaftar ke dalam keluarga tertentu diberikan revisi 1, dan definisi tugas apa pun yang terdaftar setelah itu diberi nomor revisi berurutan.

Jenis peluncuran

Saat mendaftarkan definisi tugas, Anda dapat menentukan jenis peluncuran yang harus divalidasi oleh Amazon ECS. Jika definisi tugas tidak memvalidasi terhadap kompatibilitas yang ditentukan, pengecualian klien dikembalikan. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Parameter berikut diperbolehkan dalam definisi tugas.

`requiresCompatibilities`

Tipe: Array string

Wajib: Tidak

Nilai yang Valid: `FARGATE`

Tipe peluncuran untuk memvalidasi ketentuan tugas. Ini memulai pemeriksaan untuk memastikan bahwa semua parameter yang digunakan dalam definisi tugas memenuhi persyaratan jenis peluncuran.

Peran tugas

`taskRoleArn`

Tipe: String

Wajib: Tidak

Saat mendaftarkan definisi tugas, Anda dapat memberikan peran tugas untuk peran IAM yang memungkinkan kontainer dalam izin tugas memanggil AWS APIs yang ditentukan dalam kebijakan terkait atas nama Anda. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Peran eksekusi tugas

`executionRoleArn`

Tipe: String

Wajib: Bersyarat

Nama Sumber Daya Amazon (ARN) dari peran eksekusi tugas yang memberikan izin agen penampung Amazon ECS untuk melakukan panggilan AWS API atas nama Anda.

> **Note**
>
> IAM role eksekusi tugas diperlukan sesuai dengan persyaratan tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Mode jaringan

`networkMode`

Tipe: String

Diperlukan: Ya

Mode jaringan Docker digunakan untuk kontainer dalam tugas. Untuk tugas Amazon ECS yang dihosting di Fargate, `awsvpc` mode jaringan diperlukan.

Platform runtime

`operatingSystemFamily`

Tipe: String

Wajib: Bersyarat

Standar: LINUX

Parameter ini diperlukan untuk tugas Amazon ECS yang di-host di Fargate.

Saat Anda mendaftarkan definisi tugas, Anda menentukan keluarga sistem operasi.

Nilai yang valid adalah`LINUX`,`WINDOWS_SERVER_2025_FULL`,`WINDOWS_SERVER_2025_CORE`,`WINDOWS_SERVER_202` dan`WINDOWS_SERVER_2019_CORE`.

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika definisi tugas adalah bagian dari layanan, nilai ini harus sesuai dengan `platformFamily` nilai layanan.

`cpuArchitecture`

Tipe: String

Wajib: Bersyarat

Standar: X86_64

Jika parameter dibiarkan sebagai`null`, nilai default secara otomatis ditetapkan pada inisiasi tugas yang dihosting di Fargate.

Saat Anda mendaftarkan definisi tugas, Anda menentukan arsitektur CPU. Nilai yang valid adalah `X86_64` dan `ARM64`.

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika Anda memiliki tugas Linux, Anda dapat mengatur nilainya`ARM64`. Untuk informasi selengkapnya, lihat the section called “Definisi tugas untuk beban kerja ARM 64-bit”.

Ukuran tugas

Saat Anda mendaftarkan definisi tugas, Anda dapat menentukan total CPU dan memori yang digunakan untuk tugas tersebut. Hal ini terpisah dari nilai `cpu` dan `memory` pada tingkat ketentuan kontainer. Untuk tugas yang di-host di Fargate (baik Linux dan Windows), bidang ini diperlukan dan ada nilai khusus untuk keduanya `cpu` dan `memory` yang didukung.

Parameter berikut diizinkan dalam ketentuan tugas:

`cpu`

Tipe: String

Diperlukan: Ya

> **Note**
>
> Parameter CPU dan memori tingkat tugas diperlukan dan digunakan untuk menentukan jenis dan ukuran instance tempat tugas dijalankan. Untuk tugas Windows, nilai-nilai ini tidak diberlakukan saat runtime, karena Windows tidak memiliki mekanisme asli yang dapat dengan mudah menegakkan batasan sumber daya kolektif pada sekelompok kontainer. Jika Anda ingin menerapkan batasan sumber daya, sebaiknya gunakan sumber daya tingkat kontainer untuk kontainer Windows.

Batas tegas unit CPU untuk hadir untuk tugas tersebut. Anda dapat menentukan nilai CPU dalam file JSON sebagai string dalam unit CPU atau virtual CPUs (vCPUs). Misalnya, Anda dapat menentukan nilai CPU baik seperti `1024` dalam unit CPU atau `1 vCPU` di vCPUs. Ketika definisi tugas terdaftar, nilai vCPU dikonversi ke bilangan bulat yang menunjukkan unit CPU.

Bidang ini diperlukan dan Anda harus menggunakan salah satu nilai berikut, yang menentukan rentang nilai yang didukung untuk `memory` parameter. Tabel berikut menunjukkan kombinasi yang valid antara CPU dan memori tingkat tugas.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU)	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
Note Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.		
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

`memory`

Tipe: String

Diperlukan: Ya

> **Note**
>
> Parameter CPU dan memori tingkat tugas diperlukan dan digunakan untuk menentukan jenis dan ukuran instance tempat tugas dijalankan. Untuk tugas Windows, nilai-nilai ini tidak diberlakukan saat runtime, karena Windows tidak memiliki mekanisme asli yang dapat dengan mudah menegakkan batasan sumber daya kolektif pada sekelompok kontainer. Jika Anda ingin menerapkan batasan sumber daya, sebaiknya gunakan sumber daya tingkat kontainer untuk kontainer Windows.

Batas memori yang sulit untuk disajikan pada tugas. Anda dapat menentukan nilai memori dalam definisi tugas sebagai string dalam mebibytes (MiB) atau gigabyte (GB). Misalnya, Anda

dapat menentukan nilai memori baik seperti `3072` di MiB atau `3 GB` GB. Ketika ketentuan tugas terdaftar, nilai GB dikonversi ke integer menunjukkan MiB.

Bidang ini diperlukan dan Anda harus menggunakan salah satu nilai berikut, yang menentukan rentang nilai yang didukung untuk `cpu` parameter:

Nilai memori (dalam MiB, dengan perkiraan nilai setara dalam GB)	Nilai CPU	Sistem operasi yang didukung untuk Fargate
512 (0,5 GB), 1024 (1 GB), 2048 (2 GB)	256 (.25 vCPU)	Linux
1024 (1 GB), ,2048 (2 GB), 3072 (3 GB), 4096 (4GB)	512 (.5 vCPU)	Linux
2048 (2 GB), 3072 (3 GB), 4096 (4GB), 5120 (5 GB), 6144 (6 GB), 7168 (7 GB), 8192 (8 GB)	1024 (1 vCPU)	Linux, Windows
Antara 4096 (4 GB) hingga 16384 (16 GB) sebagai tambahan dari 1024 (1 GB)	2048 (2 vCPU)	Linux, Windows
Antara 8192 (8 GB) hingga 30720 (30 GB) sebagai tambahan dari 1024 (1 GB)	4096 (4 vCPU)	Linux, Windows
Antara 16 GB dan 60 GB dalam peningkatan 4 GB **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	8192 (8 vCPU)	Linux

Nilai memori (dalam MiB, dengan perkiraan nilai setara dalam GB)	Nilai CPU	Sistem operasi yang didukung untuk Fargate
Antara 32 GB dan 120 GB dalam peningkatan 8 GB **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	16384 (16vCPU)	Linux

Definisi kontainer

Saat Anda mendaftarkan definisi tugas, Anda harus menentukan daftar definisi kontainer yang diteruskan ke Docker daemon pada instance container. Parameter berikut diizinkan dalam ketentuan kontainer.

Topik

- Parameter definisi wadah standar
- Parameter definisi wadah tingkat lanjut
- Parameter definisi wadah lainnya

Parameter definisi wadah standar

Parameter ketentuan tugas berikut diperlukan atau digunakan di sebagian besar ketentuan kontainer.

Topik

- Nama
- Citra
- Memori
- Pemetaan pelabuhan

- Kredensyal Repositori Pribadi

Nama

`name`

Tipe: String

Diperlukan: Ya

Nama kontainer. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Jika Anda menautkan beberapa kontainer dalam definisi tugas, `name` satu kontainer dapat dimasukkan ke dalam `links` wadah lain. Ini untuk menghubungkan wadah.

Citra

`image`

Tipe: String

Diperlukan: Ya

Citra yang digunakan untuk memulai kontainer. String ini diteruskan langsung ke Docker daemon. Secara default, gambar dalam Docker Hub registri tersedia. Anda juga dapat menentukan repositori lain dengan salah satu atau*repository-url*/*image*:*tag*. *repository-url*/*image*@*digest* Maksimal 255 huruf (huruf besar dan kecil), angka, tanda hubung, garis bawah, titik dua, titik, garis miring, dan tanda angka diizinkan. Parameter ini memetakan ke `Image` dalam perintah docker create-container dan parameter `IMAGE` dari perintah docker run.

- Ketika tugas baru dimulai, agen kontainer Amazon ECS menarik versi terbaru dari citra yang ditentukan dan tanda untuk kontainer yang akan digunakan. Namun, pembaruan berikutnya ke gambar repositori tidak disebarkan ke tugas yang sudah berjalan.
- Ketika Anda tidak menentukan tag atau intisari di jalur gambar dalam definisi tugas, agen penampung Amazon ECS menarik versi terbaru dari gambar yang ditentukan.
- Namun, pembaruan berikutnya ke gambar repositori tidak disebarkan ke tugas yang sudah berjalan.
- Citra di registri privat didukung. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

- Gambar di repositori Amazon ECR dapat ditentukan dengan menggunakan konvensi lengkap `registry/repository:tag` atau `registry/repository@digest` penamaan (misalnya, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com /`*my-web-app*`:`*latest* atau). *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com /`*my-web-app*`@`*sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE*
- Gambar di repositori resmi Docker Hub menggunakan satu nama (misalnya, `ubuntu` ataumongo).
- Gambar di repositori lain di Docker Hub memenuhi syarat dengan nama organisasi (misalnya,`amazon/amazon-ecs-agent`).
- Gambar di repositori online lainnya memenuhi syarat lebih lanjut dengan nama domain (misalnya, `quay.io/assemblyline/ubuntu`).

`versionConsistency`

Tipe: String

Nilai yang valid: `enabled` | `disabled`

Wajib: Tidak

Menentukan apakah Amazon ECS akan menyelesaikan tag image container yang disediakan dalam definisi container ke image digest. Secara default, perilaku ini adalah`enabled`. Jika Anda menetapkan nilai untuk container sebagai`disabled`, Amazon ECS tidak akan menyelesaikan tag image container ke intisari dan akan menggunakan URI gambar asli yang ditentukan dalam definisi container untuk penerapan. Untuk informasi selengkapnya tentang resolusi gambar kontainer, lihatResolusi gambar kontainer.

Memori

`memory`

Tipe: Integer

Wajib: Tidak

Jumlah (dalam MiB) memori yang akan ditampilkan ke kontainer. Jika kontainer Anda mencoba untuk melebihi memori yang ditentukan di sini, kontainer akan dimatikan. Jumlah total memori yang disimpan untuk semua kontainer dalam tugas harus lebih rendah dari nilai `memory` tugas, jika ada yang ditentukan. Parameter ini memetakan ke `Memory` dalam perintah docker create-container dan opsi `--memory` untuk docker run.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatMemesan memori instans kontainer Amazon ECS Linux.

`memoryReservation`

Tipe: Integer

Wajib: Tidak

Batas lunak (di MiB) memori untuk menyimpan kontainer. Ketika memori sistem sedang diperdebatkan, Docker upaya untuk menjaga memori kontainer ke batas lunak ini. Namun, wadah Anda dapat menggunakan lebih banyak memori saat dibutuhkan. Wadah dapat menggunakan hingga batas keras yang ditentukan dengan `memory` parameter (jika ada) atau semua memori yang tersedia pada instance kontainer, mana yang lebih dulu. Parameter ini memetakan ke `MemoryReservation` dalam perintah docker create-container dan opsi `--memory-reservation` untuk docker run.

Jika nilai memori tingkat tugas tidak ditentukan, Anda harus menentukan bilangan bulat bukan nol untuk salah satu atau keduanya `memory` atau `memoryReservation` dalam definisi wadah. Jika Anda menentukan keduanya, `memory` harus lebih besar dari `memoryReservation`. Jika Anda menentukan`memoryReservation`, maka nilai tersebut dikurangi dari sumber daya memori yang tersedia untuk instance wadah tempat penampung ditempatkan. Jika tidak, nilai `memory` digunakan.

Misalnya, kontainer Anda biasanya menggunakan memori 128 MiB, tetapi kadang-kadang meledak hingga 256 MiB memori untuk waktu yang singkat. Anda dapat mengatur 128 MiB, dan batas `memory` keras 300 MiB. `memoryReservation` Konfigurasi ini memungkinkan penampung untuk hanya menyimpan 128 MiB memori dari sumber daya yang tersisa pada instance container. Pada saat yang sama, konfigurasi ini juga memungkinkan wadah untuk menggunakan lebih banyak sumber daya memori bila diperlukan.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatMemesan memori instans kontainer Amazon ECS Linux.

Pemetaan pelabuhan

`portMappings`

Tipe: Array objek

Wajib: Tidak

Pemetaan port mengekspos port jaringan kontainer Anda ke dunia luar. ini memungkinkan klien untuk mengakses aplikasi Anda. Ini juga digunakan untuk komunikasi antar kontainer dalam tugas yang sama.

Untuk definisi tugas yang menggunakan mode `awsvpc` jaringan, hanya tentukan`containerPort`. `hostPort`Itu selalu diabaikan, dan port kontainer secara otomatis dipetakan ke port acak bernomor tinggi pada host.

Pemetaan port pada Windows menggunakan alamat gateway `NetNAT` daripada `localhost`. Tidak ada loopback untuk pemetaan port di Windows, jadi Anda tidak dapat mengakses port yang dipetakan kontainer dari host itu sendiri.

Sebagian besar bidang parameter ini (termasuk`containerPort`,`hostPort`,`protocol`) dipetakan ke `PortBindings` dalam perintah create-container docker dan opsi `--publish` untuk

docker run. Jika mode jaringan dari definisi tugas diatur kehost, port host harus tidak ditentukan atau cocok dengan port kontainer dalam pemetaan port.

> **Note**
>
> Setelah tugas mencapai status RUNNING , tugas port kontainer dan host manual dan otomatis akan terlihat di lokasi-lokasi berikut:
>
> - Konsol: bagian Pengikatan jaringan dari deskripsi kontainer untuk tugas yang dipilih.
> - AWS CLI: Bagian networkBindings dari output perintah describe-tasks .
> - API: Respons DescribeTasks .
> - Metadata: Titik akhir metadata tugas.

`appProtocol`

Tipe: String

Wajib: Tidak

Protokol aplikasi yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect. Kami menyarankan Anda mengatur parameter ini agar konsisten dengan protokol yang digunakan aplikasi Anda. Jika Anda menyetel parameter ini, Amazon ECS akan menambahkan penanganan koneksi khusus protokol ke proxy service connect. Jika Anda menyetel parameter ini, Amazon ECS menambahkan telemetri khusus protokol di konsol Amazon ECS dan. CloudWatch

Jika Anda tidak menetapkan nilai untuk parameter ini, maka TCP digunakan. Namun, Amazon ECS tidak menambahkan telemetri khusus protokol untuk TCP.

Untuk informasi selengkapnya, lihat the section called “ Layanan Connect”.

Nilai protokol yang valid: `"http" | "http2" | "grpc"`

`containerPort`

Jenis: Integer

Diperlukan: Ya, kapan portMappings digunakan

Nomor port pada kontainer yang terikat ke port host yang ditentukan pengguna atau secara otomatis ditetapkan.

Untuk tugas yang menggunakan mode `awsvpc` jaringan, Anda gunakan `containerPort` untuk menentukan port yang terbuka.

Untuk wadah Windows di Fargate, Anda tidak dapat menggunakan port 3150 untuk file. `containerPort` Ini karena sudah dipesan.

`containerPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada kontainer yang terikat pada rentang port host yang dipetakan secara dinamis.

Anda hanya dapat mengatur parameter ini dengan menggunakan `register-task-definition` API. Opsi ini tersedia dalam `portMappings` parameter. Untuk informasi selengkapnya, lihat register-task-definition dalam Referensi AWS Command Line Interface .

Aturan berikut berlaku saat Anda menentukan`containerPortRange`:

- Anda harus menggunakan mode `awsvpc` jaringan.
- Parameter ini tersedia untuk sistem operasi Linux dan Windows.
- Instance container harus memiliki setidaknya versi 1.67.0 dari agen kontainer dan setidaknya versi 1.67.0-1 dari paket. `ecs-init`
- Anda dapat menentukan maksimum 100 rentang port untuk setiap kontainer.
- Anda tidak menentukan`hostPortRange`. Nilai `hostPortRange` ditetapkan sebagai berikut:
  - Untuk kontainer dalam tugas dengan mode `awsvpc` jaringan, `hostPort` diatur ke nilai yang sama dengan`containerPort`. Ini adalah strategi pemetaan statis.
- Nilai yang `containerPortRange` valid adalah antara 1 dan 65535.
- Sebuah port hanya dapat dimasukkan dalam satu pemetaan port untuk setiap kontainer.
- Anda tidak dapat menentukan rentang port yang tumpang tindih.
- Port pertama dalam kisaran harus kurang dari port terakhir dalam kisaran.
- Dockermerekomendasikan agar Anda mematikan docker-proxy di file konfigurasi Docker daemon ketika Anda memiliki banyak port.

  Untuk informasi selengkapnya, lihat Masalah #11185 di GitHub.

Untuk informasi tentang cara mematikan docker-proxy di file konfigurasi daemon, lihat Docker daemon Docker di Panduan Pengembang Amazon ECS.

Anda dapat menelepon DescribeTasksuntuk melihat`hostPortRange`, yang merupakan port host yang terikat ke port kontainer.

Rentang port tidak disertakan dalam peristiwa tugas Amazon ECS, yang dikirim ke EventBridge. Untuk informasi selengkapnya, lihat the section called “Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge”.

`hostPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada host yang digunakan dengan pengikatan jaringan. Ini ditugaskan oleh Docker dan dikirimkan oleh agen Amazon ECS.

`hostPort`

Tipe: Integer

Wajib: Tidak

Nomor port pada instans kontainer untuk menyimpan kontainer Anda.

`hostPort`Dapat tetap kosong atau nilainya sama dengan`containerPort`.

Rentang port fana default Docker versi 1.6.0 dan yang lebih baru tercantum pada instance di bawah. `/proc/sys/net/ipv4/ip_local_port_range` Jika parameter kernel ini tidak tersedia, port sementara default berkisar sejak `49153–65535` digunakan. Jangan mencoba menentukan port host dalam rentang port sementara. Ini karena ini dicadangkan untuk penugasan otomatis. Secara umum, port di bawah `32768` berada di luar jangkauan port fana.

Port cadangan default adalah `22` untuk SSH, Docker port `2375` dan`2376`, dan port agen kontainer Amazon ECS. `51678-51680` Port host apa pun yang sebelumnya ditentukan pengguna untuk tugas yang sedang berjalan juga dicadangkan saat tugas sedang berjalan. Setelah tugas berhenti, port host dilepaskan. Port cadangan saat ini ditampilkan di `remainingResources` describe-container-instances output. Instance kontainer mungkin

memiliki hingga 100 port cadangan sekaligus, termasuk port cadangan default. Port yang ditetapkan secara otomatis tidak dihitung terhadap kuota 100 port cadangan.

`name`

Tipe: String

Wajib: Tidak, diperlukan agar Service Connect dan VPC Lattice dikonfigurasi dalam layanan

Nama yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect dan VPC Lattice. Parameter ini adalah nama yang Anda gunakan dalam konfigurasi Service Connect dan VPC Lattice suatu layanan.

Untuk informasi selengkapnya, lihat Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek.

Dalam contoh berikut, kedua bidang wajib untuk Service Connect dan VPC Lattice digunakan.

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
]
```

`protocol`

Tipe: String

Wajib: Tidak

Protokol yang digunakan untuk pemetaan port. Nilai yang valid adalah `tcp` dan `udp`. Default adalah `tcp`.

> **Important**
>
> Hanya `tcp` didukung untuk Service Connect. Ingat itu `tcp` tersirat jika bidang ini tidak disetel.

Jika Anda menentukan port host, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
]
```

Jika Anda menginginkan port host yang ditetapkan secara otomatis, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer
    }
    ...
]
```

Kredensyal Repositori Pribadi

`repositoryCredentials`

Tipe: Objek RepositoryCredentials

Wajib: Tidak

Kredensyal repositori untuk otentikasi registri pribadi.

Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

`credentialsParameter`

Tipe: String

Diperlukan: Ya, kapan `repositoryCredentials` digunakan

Amazon Resource Name (ARN) dari rahasia yang berisi kredensial repositori privat.

Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

> **Note**
>
> Saat Anda menggunakan Amazon ECS API, AWS CLI, atau AWS SDKs, jika rahasia ada di Wilayah yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama rahasianya. Bila Anda menggunakan AWS Management Console, Anda harus menentukan ARN penuh rahasia.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan parameter yang diperlukan:

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

Parameter definisi wadah tingkat lanjut

Parameter definisi kontainer lanjutan berikut memberikan kemampuan yang diperluas ke perintah docker run yang digunakan untuk meluncurkan container di instance container Amazon ECS Anda.

Topik

- Mulai ulang kebijakan
- Pemeriksaan kondisi
- Lingkungan
- Pengaturan jaringan
- Penyimpanan dan pencatatan
- Keamanan
- Batas sumber daya
- Label docker

Mulai ulang kebijakan

`restartPolicy`

Kebijakan restart kontainer dan parameter konfigurasi terkait. Saat Anda menyiapkan kebijakan restart untuk penampung, Amazon ECS dapat memulai ulang penampung tanpa perlu mengganti tugas. Untuk informasi selengkapnya, lihat Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer.

`enabled`

Jenis: Boolean

Wajib: Ya

Menentukan apakah kebijakan restart diaktifkan untuk wadah.

`ignoredExitCodes`

Jenis: himpunan integer

Wajib: Tidak

Daftar kode keluar yang akan diabaikan Amazon ECS dan tidak mencoba memulai ulang. Anda dapat menentukan maksimum 50 kode keluar kontainer. Secara default, Amazon ECS tidak mengabaikan kode keluar apa pun.

`restartAttemptPeriod`

Tipe: Integer

Wajib: Tidak

Periode waktu (dalam detik) wadah harus dijalankan sebelum restart dapat dicoba. Sebuah wadah dapat dimulai ulang hanya sekali setiap `restartAttemptPeriod` detik. Jika kontainer tidak dapat berjalan untuk periode waktu ini dan keluar lebih awal, itu tidak akan dimulai ulang. Anda dapat mengatur minimal `restartAttemptPeriod` 60 detik dan maksimum `restartAttemptPeriod` 1800 detik. Secara default, kontainer harus berjalan selama 300 detik sebelum dapat dimulai ulang.

Pemeriksaan kondisi

`healthCheck`

Perintah pemeriksaan kesehatan kontainer dan parameter konfigurasi terkait untuk wadah. Untuk informasi selengkapnya, lihat Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer.

`command`

Array string yang mewakili perintah yang dijalankan kontainer untuk menentukan apakah itu sehat. Array string dapat dimulai dengan `CMD` untuk menjalankan argumen perintah secara langsung, atau `CMD-SHELL` untuk menjalankan perintah dengan shell default container. Jika tidak ada yang `CMD` ditentukan, digunakan.

Saat mendaftarkan definisi tugas di AWS Management Console, gunakan daftar perintah yang dipisahkan koma. Perintah ini dikonversi ke string setelah definisi tugas dibuat. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

Saat mendaftarkan definisi tugas menggunakan panel AWS Management Console JSON,, atau AWS CLI APIs, lampirkan daftar perintah dalam tanda kurung. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

Kode keluar 0, tanpa `stderr` output, menunjukkan keberhasilan, dan kode keluar bukan nol menunjukkan kegagalan.

`interval`

Periode waktu (dalam detik) antara setiap pemeriksaan kesehatan. Anda dapat menentukan antara 5 dan 300 detik. Nilai defaultnya adalah 30 detik.

`timeout`

Periode waktu (dalam hitungan detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal. Anda dapat menentukan antara 2 dan 60 detik. Nilai defaultnya adalah 5 detik.

`retries`

Jumlah raktu untuk mencoba kembali pemeriksaan kondisi yang gagal sebelum kontainer dianggap tidak sehat. Anda dapat menentukan antara 1 dan 10 percobaan ulang. Nilai default-nya adalah tiga kali percobaan ulang.

`startPeriod`

Masa tenggang opsional untuk menyediakan waktu kontainer untuk bootstrap sebelum pemeriksaan kesehatan yang gagal dihitung terhadap jumlah maksimum percobaan ulang. Anda dapat menentukan nilai antara 0 dan 300 detik. Secara default, `startPeriod` dinonaktifkan.

Jika pemeriksaan kondisi berhasil dalam `startPeriod`, maka kontainer dianggap sehat dan kegagalan berikutnya dihitung terhadap jumlah maksimum percobaan kembali.

Lingkungan

`cpu`

Tipe: Integer

Wajib: Tidak

Jumlah `cpu` unit cadangan agen kontainer Amazon ECS untuk kontainer. Di Linux, parameter ini dipetakan `CpuShares` di bagian Create a container.

Bidang ini opsional untuk tugas yang menggunakan jenis peluncuran Fargate. Jumlah total CPU yang disediakan untuk semua kontainer yang berada dalam tugas harus lebih rendah dari nilai tingkat tugas`cpu`.

Linuxkontainer berbagi unit CPU yang tidak terisi dengan kontainer lain pada instance kontainer dengan rasio yang sama dengan jumlah yang dialokasikan. Misalnya, asumsikan bahwa Anda menjalankan tugas kontainer tunggal pada tipe instans inti tunggal dengan 512 unit CPU yang ditentukan untuk kontainer tersebut. Selain itu, tugas itu adalah satu-satunya tugas yang berjalan pada instance container. Dalam contoh ini, wadah dapat menggunakan 1.024 pangsa unit CPU penuh pada waktu tertentu. Namun, asumsikan bahwa Anda meluncurkan salinan lain dari tugas yang sama pada instance kontainer itu. Setiap tugas dijamin minimal 512 unit CPU bila diperlukan. Demikian pula, jika wadah lain tidak menggunakan CPU yang tersisa, setiap kontainer dapat mengambang ke penggunaan CPU yang lebih tinggi. Namun, jika kedua tugas tersebut 100% aktif sepanjang waktu, mereka terbatas pada 512 unit CPU.

Pada instance Linux container, Docker daemon pada instance container menggunakan nilai CPU untuk menghitung rasio pangsa CPU relatif untuk menjalankan container. Nilai pangsa CPU minimum yang valid yang diizinkan oleh kernel Linux adalah 2, dan nilai pangsa CPU valid maksimum yang diizinkan oleh kernel Linux adalah 262144. Namun, parameter CPU tidak diperlukan, dan Anda dapat menggunakan nilai CPU di bawah dua dan di atas 262144 dalam definisi container Anda. Untuk nilai CPU di bawah dua (termasuk null) dan di atas 262144, perilaku bervariasi berdasarkan versi agen penampung Amazon ECS Anda:

Pada instance Windows kontainer, kuota CPU diberlakukan sebagai kuota absolut. Windowscontainer hanya memiliki akses ke jumlah CPU tertentu yang ditentukan dalam definisi tugas. Nilai CPU nol atau nol diteruskan ke Docker as`0`. Windowskemudian menafsirkan nilai ini sebagai 1% dari satu CPU.

Untuk contoh selengkapnya, lihat Cara Amazon ECS mengelola sumber daya CPU dan memori.

`gpu`

Parameter ini tidak didukung untuk kontainer yang di-host di Fargate.

Tipe: Objek ResourceRequirement

Wajib: Tidak

Jumlah fisik `GPUs` yang dicadangkan agen kontainer Amazon ECS untuk kontainer. Jumlah GPUs cadangan untuk semua kontainer dalam tugas tidak boleh melebihi jumlah yang tersedia GPUs pada instance kontainer tempat tugas diluncurkan. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.

`Elastic Inference accelerator`

Parameter ini tidak didukung untuk kontainer yang di-host di Fargate.

Tipe: Objek ResourceRequirement

Wajib: Tidak

Untuk `InferenceAccelerator` jenisnya, `value` cocok dengan yang `deviceName` `InferenceAccelerator` ditentukan dalam definisi tugas. Untuk informasi selengkapnya, lihat the section called “Nama akselerator Elastic Inference”.

`essential`

Tipe: Boolean

Wajib: Tidak

Misalkan `essential` parameter wadah ditandai sebagai`true`, dan wadah itu gagal atau berhenti karena alasan apa pun. Kemudian, semua wadah lain yang merupakan bagian dari tugas dihentikan. Jika `essential` parameter wadah ditandai sebagai`false`, maka kegagalannya tidak mempengaruhi sisa kontainer dalam suatu tugas. Jika parameter ini dihilangkan, kontainer diasumsikan menjadi penting.

Semua tugas harus memiliki setidaknya satu kontainer penting. Misalkan Anda memiliki aplikasi yang terdiri dari beberapa kontainer. Kemudian, kelompokkan wadah yang digunakan untuk tujuan bersama menjadi komponen, dan pisahkan komponen yang berbeda menjadi beberapa definisi tugas. Untuk informasi selengkapnya, lihat Arsitek aplikasi Anda untuk Amazon ECS.

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> Versi awal agen kontainer Amazon ECS tidak menangani `entryPoint` parameter dengan benar. Jika Anda memiliki masalah menggunakan `entryPoint`, perbarui agen kontainer Anda atau masukkan perintah dan argumen Anda sebagai item array `command` sebagai gantinya.

Tipe: Array string

Wajib: Tidak

Titik masuk yang diteruskan ke wadah.

```
"entryPoint": ["string", ...]
```

`command`

Tipe: Array string

Wajib: Tidak

Perintah yang diteruskan ke kontainer. Parameter ini memetakan ke `Cmd` dalam perintah create-container dan parameter `COMMAND` ke docker run. Jika ada beberapa argumen, pastikan bahwa setiap argumen adalah string terpisah dalam array.

```
"command": ["string", ...]
```

`workingDirectory`

Tipe: String

Wajib: Tidak

Direktori kerja untuk menjalankan perintah di dalam wadah di. Parameter ini memetakan ke `WorkingDir` di bagian Membuat kontainer dari API Jarak Jauh Docker dan pilihan `--workdir` untuk docker run.

```
"workingDirectory": "string"
```

`environmentFiles`

Ini tidak tersedia untuk wadah Windows di Fargate.

Tipe: Array objek

Wajib: Tidak

Daftar file yang berisi variabel lingkungan untuk diteruskan ke kontainer. Parameter ini memetakan ke `--env-file` opsi ke perintah docker run.

Anda dapat menentukan hingga 10 file lingkungan. File harus memiliki ekstensi file `.env`. Setiap baris dalam file lingkungan berisi variabel lingkungan dalam `VARIABLE=VALUE` format. Baris yang dimulai dengan # diperlakukan sebagai komentar dan diabaikan.

Jika ada variabel lingkungan individu yang ditentukan dalam ketentuan kontainer, maka akan lebih diutamakan daripada variabel yang ada di dalam file lingkungan. Jika beberapa file lingkungan ditentukan yang berisi variabel yang sama, mereka diproses dari atas ke bawah. Kami merekomendasikan agar Anda menggunakan nama variabel yang unik. Untuk informasi selengkapnya, lihat Meneruskan variabel lingkungan individual ke wadah Amazon ECS.

`value`

Tipe: String

Wajib: Ya

Amazon Resource Name (ARN) dari objek Amazon S3 yang berisi variabel lingkungan file.

`type`

Tipe: String

Wajib: Ya

Tipe file yang akan digunakan. Satu-satunya nilai yang didukung adalah `s3`.

`environment`

Tipe: Array objek

Wajib: Tidak

Variabel lingkungan untuk disampaikan ke kontainer. Parameter ini memetakan ke `Env` dalam perintah docker create-container dan opsi `--env` ke perintah docker run.

> **Important**
>
> Kami tidak menyarankan menggunakan variabel lingkungan teks biasa untuk informasi sensitif, seperti data kredensial.

`name`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nama variabel lingkungan.

`value`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nilai dari variabel lingkungan.

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
```

```
]
```

`secrets`

Tipe: Object array

Wajib: Tidak

Objek yang mewakili rahasia untuk mengekspos ke wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk mengekspos ke kontainer. Nilai yang didukung adalah Nama Sumber Daya Amazon (ARN) lengkap dari AWS Secrets Manager rahasia atau ARN lengkap parameter di Parameter Store. AWS Systems Manager

> **Note**
>
> Jika parameter Systems Manager Parameter Store atau parameter Secrets Manager Wilayah AWS sama dengan tugas yang Anda luncurkan, Anda dapat menggunakan ARN lengkap atau nama rahasia. Jika parameter ada di Wilayah yang berbeda, ARN lengkap harus disebutkan.

```
"secrets": [
    {
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

Pengaturan jaringan

`disableNetworking`

Parameter ini tidak didukung untuk tugas yang berjalan di Fargate.

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini benar, jaringan mati di dalam wadah.

Nilai default-nya `false`.

```
"disableNetworking": true|false
```

`links`

Parameter ini tidak didukung untuk tugas yang menggunakan mode `awsvpc` jaringan.

Tipe: Array string

Wajib: Tidak

Parameter `link` mengizinkan kontainer untuk berkomunikasi satu sama lain tanpa perlu port pemetaan. Parameter ini hanya didukung jika mode jaringan dari definisi tugas diatur ke`bridge`. `name:internalName`Konstruk ini analog dengan dalam tautan. `name:alias` Docker Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung, dan garis bawah diperbolehkan..

> **Important**
>
> Kontainer yang ditempatkan pada instance kontainer yang sama dapat berkomunikasi satu sama lain tanpa memerlukan tautan atau pemetaan port host. Isolasi jaringan pada instans kontainer dikendalikan oleh grup keamanan dan pengaturan VPC.

```
"links": ["name:internalName", ...]
```

`hostname`

Tipe: String

Wajib: Tidak

Nama host yang digunakan untuk kontainer Anda. Parameter ini memetakan ke `Hostname` dalam docker create-container dan opsi `--hostname` untuk docker run.

> **Note**
>
> Jika Anda menggunakan mode `awsvpc` jaringan, `hostname` parameter tidak didukung.

```
"hostname": "string"
```

`dnsServers`

Ini tidak didukung untuk tugas yang berjalan di Fargate.

Tipe: Array string

Wajib: Tidak

Daftar server DNS yang disajikan untuk kontainer.

```
"dnsServers": ["string", ...]
```

`extraHosts`

Parameter ini tidak didukung untuk tugas yang menggunakan mode `awsvpc` jaringan.

Tipe: Array objek

Wajib: Tidak

Daftar nama host dan pemetaan alamat IP untuk ditambahkan ke file `/etc/hosts` di kontainer.

Parameter ini memetakan ke `ExtraHosts` dalam perintah docker create-container dan opsi `--add-host` untuk docker run.

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
      }
```

```
    ...
]
```

`hostname`

Tipe: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Nama host yang akan digunakan dalam entri `/etc/hosts`.

`ipAddress`

Tipe: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Alamat IP yang akan digunakan dalam `/etc/hosts` entri.

Penyimpanan dan pencatatan

`readonlyRootFilesystem`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini betul, kontainer diberikan akses hanya-baca ke sistem file asalnya. Parameter ini memetakan ke `ReadonlyRootfs` dalam perintah docker create-container opsi `--read-only` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Nilai default-nya `false`.

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

Tipe: Array objek

Wajib: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` dalam create-container Docker API dan opsi `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive. Anda harus menentukan titik pemasangan untuk melampirkan volume Amazon EBS langsung ke tugas Amazon ECS.

`sourceVolume`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Untuk tugas yang berjalan pada EC2 instance yang menjalankan sistem operasi Windows, biarkan nilainya sebagai default. `false`

`volumesFrom`

Tipe: Array objek

Wajib: Tidak

Volume data untuk memasang dari kontainer lain. Parameter ini memetakan ke `VolumesFrom` dalam perintah docker create-container dan opsi `--volumes-from` untuk docker run.

`sourceContainer`

Tipe: String

Diperlukan: Ya, saat `volumesFrom` digunakan

Nama kontainer untuk memasang volume.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

```
"volumesFrom": [
                {
                  "sourceContainer": "string",
                  "readOnly": true|false
                }
              ]
```

`logConfiguration`

Jenis: LogConfigurationObjek

Wajib: Tidak

Spesifikasi konfigurasi log untuk kontainer.

Misalnya definisi tugas yang menggunakan konfigurasi log, lihatContoh definisi tugas Amazon ECS.

Parameter ini memetakan ke `LogConfig` dalam perintah docker create-container dan opsi `--log-driver` untuk docker run. Secara default, kontainer menggunakan driver logging yang sama dengan yang digunakan Docker daemon. Namun, penampung mungkin menggunakan driver logging yang berbeda dari Docker daemon dengan menentukan driver log dengan parameter ini dalam definisi container. Untuk menggunakan driver penlog yang berbeda untuk kontainer, sistem log harus dikonfigurasi dengan benar pada instans kontainer (atau pada server log yang berbeda untuk opsi penlog jarak jauh).

Pertimbangkan hal berikut saat menentukan konfigurasi log untuk kontainer Anda:

- Amazon ECS mendukung subset driver logging yang tersedia untuk daemon. Docker Driver log tambahan mungkin tersedia di rilis mendatang agen kontainer Amazon ECS.
- Parameter ini memerlukan Docker Remote API versi 1.18 atau yang lebih baru pada instance container Anda.
- Anda harus menginstal perangkat lunak tambahan apa pun di luar tugas. Misalnya, agregator Fluentd keluaran atau host jarak jauh yang menjalankan Logstash untuk mengirim log ke. Gelf

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","splunk","syslog","awsfirelens",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

`logDriver`

Tipe: String

Nilai yang valid: `"awslogs","fluentd","gelf","json-file","journald","splunk","syslog","awsfirelens"`

Diperlukan: Ya, saat `logConfiguration` digunakan

Driver log yang digunakan untuk kontainer. Secara default, nilai valid yang tercantum sebelumnya adalah driver log yang dapat berkomunikasi dengan agen penampung Amazon ECS.

Driver log yang didukung adalah`awslogs`,`splunk`, dan`awsfirelens`.

Untuk informasi selengkapnya tentang cara menggunakan driver `awslogs` log dalam definisi tugas untuk mengirim log kontainer Anda ke CloudWatch Log, lihatKirim log Amazon ECS ke CloudWatch .

Untuk informasi selengkapnya tentang penggunaan driver log `awsfirelens`, lihat Perutean Log Kustom.

> **Note**
>
> Jika Anda memiliki driver khusus yang tidak terdaftar, Anda dapat melakukan fork proyek agen penampung Amazon ECS yang tersedia GitHub dan menyesuaikannya agar berfungsi dengan driver itu. Kami mendorong Anda untuk mengirim permintaan tarik untuk perubahan yang ingin Anda sertakan. Namun, saat ini kami tidak mendukung menjalankan salinan modifikasi dari perangkat lunak ini.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

`options`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta opsi konfigurasi untuk dikirim ke driver log.

Opsi yang dapat Anda tentukan tergantung pada driver log. Beberapa opsi yang dapat Anda tentukan saat Anda menggunakan `awslogs` router untuk merutekan log ke Amazon CloudWatch meliputi yang berikut:

`awslogs-create-group`

Wajib: Tidak

Tentukan apakah Anda ingin grup log dibuat secara otomatis. Jika opsi ini tidak ditentukan, default-nya adalah `false`.

> **Note**
>
> Kebijakan IAM Anda harus menyertakan `logs:CreateLogGroup` izin sebelum Anda mencoba menggunakannya`awslogs-create-group`.

`awslogs-region`

Wajib: Ya

Tentukan Wilayah AWS driver `awslogs` log untuk mengirim log Docker Anda. Anda dapat memilih untuk mengirim semua log Anda dari kluster di Wilayah yang berbeda ke satu wilayah di CloudWatch Log. Ini agar mereka semua terlihat di satu lokasi. Jika tidak, Anda dapat memisahkannya berdasarkan Wilayah untuk perincian lebih lanjut. Pastikan bahwa grup log yang ditentukan ada di Wilayah yang Anda tentukan dengan opsi ini.

`awslogs-group`

Wajib: Ya

Pastikan untuk menentukan grup log tempat driver `awslogs` log mengirimkan aliran lognya.

`awslogs-stream-prefix`

Wajib: Ya

Gunakan `awslogs-stream-prefix` opsi untuk mengaitkan aliran log dengan awalan yang ditentukan, nama penampung, dan ID tugas Amazon ECS tempat penampung tersebut berada. Jika Anda menentukan awalan dengan opsi ini, maka aliran log mengambil format berikut.

```
prefix-name/container-name/ecs-task-id
```

Jika Anda tidak menentukan awalan dengan opsi ini, maka aliran log dinamai sesuai dengan ID penampung yang ditetapkan oleh daemon Docker pada instance container. Karena sulit untuk melacak log kembali ke wadah yang mengirimnya hanya dengan ID wadah Docker (yang hanya tersedia pada instance penampung), kami sarankan Anda menentukan awalan dengan opsi ini.

Untuk layanan Amazon ECS, Anda dapat menggunakan nama layanan sebagai awalan. Dengan melakukannya, Anda dapat melacak aliran log ke layanan tempat penampung milik, nama penampung yang mengirimnya, dan ID tugas yang menjadi milik penampung tersebut.

Anda harus menentukan awalan aliran agar log Anda muncul di panel Log saat menggunakan konsol Amazon ECS.

`awslogs-datetime-format`

Wajib: Tidak

Opsi ini mendefinisikan pola mulai multiline di Python format `strftime`. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Salah satu contoh kasus penggunaan format ini adalah untuk penguraian output seperti buangan tumpukan, yang mungkin bisa tercatat dalam beberapa entri. Pola yang benar akan memungkinkannya ditangkap dalam satu entri.

Untuk informasi selengkapnya, lihat awslogs-datetime-format.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

`awslogs-multiline-pattern`

Wajib: Tidak

Opsi ini mendefinisikan pola mulai multiline yang menggunakan ekspresi reguler. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Untuk informasi selengkapnya, lihat awslogs-multiline-pattern.

Opsi ini diabaikan jika `awslogs-datetime-format` juga dikonfigurasi.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

Opsi berikut berlaku untuk semua driver log yang didukung.

`mode`

Wajib: Tidak

Nilai yang valid: `non-blocking` | `blocking`

Opsi ini mendefinisikan mode pengiriman pesan log dari wadah ke driver log yang ditentukan menggunakan`logDriver`. Mode pengiriman yang Anda pilih memengaruhi ketersediaan aplikasi saat aliran log dari kontainer terganggu.

Jika Anda menggunakan `blocking` mode dan aliran log terputus, panggilan dari kode kontainer untuk menulis ke `stdout` dan `stderr` aliran akan diblokir. Thread logging aplikasi akan diblokir sebagai hasilnya. Hal ini dapat menyebabkan aplikasi menjadi tidak responsif dan menyebabkan kegagalan pemeriksaan kesehatan kontainer.

Jika Anda menggunakan `non-blocking` mode, log kontainer malah disimpan dalam buffer perantara dalam memori yang dikonfigurasi dengan opsi. `max-buffer-size` Ini mencegah aplikasi menjadi tidak responsif ketika log tidak dapat dikirim. Kami merekomendasikan menggunakan mode ini jika Anda ingin memastikan ketersediaan layanan dan baik-baik saja dengan beberapa kehilangan log. Untuk informasi selengkapnya, lihat Mencegah kehilangan log dengan mode non-pemblokiran di driver log `awslogs` kontainer.

Anda dapat mengatur default `mode` untuk semua kontainer tertentu Wilayah AWS dengan menggunakan setelan `defaultLogDriverMode` akun. Jika Anda tidak menentukan `mode` opsi di `logConfiguration` atau mengonfigurasi pengaturan akun, Amazon ECS akan default ke `non-blocking` mode. Untuk informasi selengkapnya tentang pengaturan akun, lihatMode driver log default.

Saat `non-blocking` mode digunakan, opsi `max-buffer-size` log mengontrol ukuran buffer yang digunakan untuk penyimpanan pesan perantara. Pastikan untuk menentukan ukuran buffer yang memadai berdasarkan aplikasi Anda. Jumlah total memori yang dialokasikan pada tingkat tugas harus lebih besar dari jumlah memori yang dialokasikan untuk semua wadah selain buffer memori driver log.

> **Note**
>
> Pada 25 Juni 2025, Amazon ECS mengubah mode driver log default dari menjadi `blocking` `non-blocking` memprioritaskan ketersediaan tugas daripada

pencatatan. Untuk terus menggunakan `blocking` mode setelah perubahan ini, lakukan salah satu hal berikut:

- Tetapkan `mode` opsi dalam definisi kontainer Anda `logConfiguration` sebagai`blocking`.
- Atur pengaturan `defaultLogDriverMode` akun ke`blocking`.

`max-buffer-size`

Wajib: Tidak

Nilai default: `10m`

Saat `non-blocking` mode digunakan, opsi `max-buffer-size` log mengontrol ukuran buffer yang digunakan untuk penyimpanan pesan perantara. Pastikan untuk menentukan ukuran buffer yang memadai berdasarkan aplikasi Anda. Ketika buffer terisi, log lebih lanjut tidak dapat disimpan. Log yang tidak dapat disimpan hilang.

Untuk merutekan `splunk` log menggunakan router log, Anda perlu menentukan a `splunk-token` dan a`splunk-url`.

Saat Anda menggunakan router `awsfirelens` log untuk merutekan log ke AWS Partner Network tujuan Layanan AWS atau penyimpanan log dan analitik, Anda dapat mengatur `log-driver-buffer-limit` opsi untuk membatasi jumlah peristiwa yang di-buffer dalam memori, sebelum dikirim ke wadah router log. Ini dapat membantu menyelesaikan potensi masalah kehilangan log karena throughput yang tinggi dapat mengakibatkan memori habis untuk buffer di dalamnya. Docker Untuk informasi selengkapnya, lihat the section called “Mengkonfigurasi log untuk throughput tinggi”.

Opsi lain yang dapat Anda tentukan saat menggunakan `awsfirelens` untuk merutekan log tergantung pada tujuan. Saat mengekspor log ke Amazon Data Firehose, Anda dapat menentukan Wilayah AWS dengan `region` dan nama untuk aliran log. `delivery_stream`

Saat Anda mengekspor log ke Amazon Kinesis Data Streams, Anda dapat Wilayah AWS menentukan dengan dan `region` nama aliran data dengan. `stream`

Saat mengekspor log ke Amazon OpenSearch Service, Anda dapat menentukan opsi sepertiName, `Host` (Titik akhir OpenSearch layanan tanpa protokol),,,`Port`,`Index`,`Type`,`Aws_auth`, `Aws_regionSuppress_Type_Name`, dan`tls`.

Saat mengekspor log ke Amazon S3, Anda dapat menentukan bucket menggunakan opsi. `bucket` Anda juga dapat menentukan`region`,`total_file_size`,`upload_timeout`, dan `use_put_object` sebagai opsi.

Parameter ini memerlukan API Docker Remote versi 1.19 atau lebih besar pada instans kontainer Anda.

`secretOptions`

Tipe: Array objek

Wajib: Tidak

Objek yang mewakili rahasia untuk diteruskan ke konfigurasi log. Rahasia yang digunakan dalam konfigurasi log dapat mencakup token otentikasi, sertifikat, atau kunci enkripsi. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk diekspos ke konfigurasi log kontainer.

```
"logConfiguration": {
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
```

```
}
```

`firelensConfiguration`

Jenis: FirelensConfigurationObjek

Wajib: Tidak

FireLens Konfigurasi untuk wadah. Digunakan untuk menentukan dan mengonfigurasi router log untuk log kontainer. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
            "KeyName": ""
        }
    }
}
```

`options`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta opsi untuk digunakan saat mengkonfigurasi router log. Bidang ini opsional dan dapat digunakan untuk menentukan file konfigurasi kustom atau untuk menambahkan metadata tambahan, seperti tugas, ketentuan tugas, klaster, dan detail instans kontainer ke log acara. Jika ditentukan, sintaksis yang digunakan adalah `"options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-value":"arn:aws:s3:::amzn-s3-demo-bucket/fluent.conf|filepath"}`. Untuk informasi selengkapnya, lihat Contoh definisi tugas Amazon ECS: Rute log ke FireLens.

`type`

Tipe: String

Wajib: Ya

Router log yang akan digunakan. Nilai yang valid adalah `fluentd` atau `fluentbit`.

Keamanan

Untuk informasi selengkapnya tentang keamanan kontainer, lihat praktik terbaik tugas Amazon ECS dan keamanan kontainer.

`credentialSpecs`

Tipe: Array string

Wajib: Tidak

Daftar ARNs di SSM atau Amazon S3 ke file credential spec `CredSpec` () yang mengonfigurasi wadah untuk otentikasi Active Directory. Kami menyarankan Anda menggunakan parameter ini alih-alih`dockerSecurityOptions`. Jumlah maksimum ARNs adalah 1.

Ada dua format untuk setiap ARN.

CredentialSpecDomainless:mYarn

Anda gunakan `credentialspecdomainless:MyARN` untuk `CredSpec` menyediakan bagian tambahan untuk rahasia di Secrets Manager. Anda memberikan kredensyal login ke domain secara rahasia.

Setiap tugas yang berjalan pada instance kontainer apa pun dapat bergabung dengan domain yang berbeda.

Anda dapat menggunakan format ini tanpa menggabungkan instance container ke domain.

Spesifikasi kredensial:mYarn

Anda gunakan `credentialspec:MyARN` untuk menyediakan `CredSpec` untuk satu domain.

Anda harus menggabungkan instance penampung ke domain sebelum memulai tugas apa pun yang menggunakan definisi tugas ini.

Dalam kedua format, ganti `MyARN` dengan ARN di SSM atau Amazon S3.

`credspec`Harus menyediakan ARN di Secrets Manager untuk rahasia yang berisi nama pengguna, kata sandi, dan domain untuk terhubung. Untuk keamanan yang lebih baik, instance tidak digabungkan ke domain untuk otentikasi tanpa domain. Aplikasi lain pada instance tidak dapat menggunakan kredensyal tanpa domain. Anda dapat menggunakan parameter ini untuk menjalankan tugas pada instance yang sama, bahkan tugas harus bergabung dengan domain yang berbeda. Untuk informasi selengkapnya, lihat Menggunakan g MSAs untuk Kontainer Windows dan Menggunakan g MSAs untuk Kontainer Linux.

`user`

Tipe: String

Wajib: Tidak

Pengguna yang akan digunakan di dalam kontainer. Parameter ini memetakan ke `User` dalam perintah docker create-container dan opsi `--user` untuk docker run.

> **Important**
>
> Saat menjalankan tugas yang menggunakan mode `host` jaringan, jangan jalankan kontainer menggunakan pengguna root (UID 0). Sebagai praktik terbaik keamanan, selalu gunakan pengguna non-root.

Anda dapat menentukan `user` dengan menggunakan format berikut. Jika menentukan UID atau GID, Anda harus menentukannya sebagai bilangan bulat positif.

- `user`
- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"user": "string"
```

Batas sumber daya

`ulimits`

Tipe: Array objek

Wajib: Tidak

Daftar nilai `ulimit` yang harus ditentukan untuk kontainer. Nilai ini menimpa pengaturan kuota sumber daya default untuk sistem operasi. Parameter ini memetakan ke `Ulimits` dalam perintah docker create-container dan opsi `--ulimit` untuk docker run.

Tugas Amazon ECS yang dihosting di Fargate menggunakan nilai batas sumber daya default yang ditetapkan oleh sistem operasi dengan pengecualian parameter batas sumber daya. `nofile` Batas `nofile` sumber daya menetapkan batasan pada jumlah file terbuka yang dapat digunakan wadah. Di Fargate, batas `nofile` lunak default adalah `65535` dan batas keras adalah. `65535` Anda dapat mengatur nilai dari kedua batas hingga`1048576`. Untuk informasi selengkapnya, lihat Batas sumber daya tugas.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
    ]
```

`name`

Tipe: String

Nilai yang valid: `"core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"`

Diperlukan: Ya, kapan `ulimits` digunakan

`type` dari `ulimit`.

`hardLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas keras untuk tipe `ulimit`. Nilai dapat ditentukan dalam byte, detik, atau sebagai hitungan, tergantung `type` pada`ulimit`.

`softLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas lunak untuk tipe `ulimit`. Nilai dapat ditentukan dalam byte, detik, atau sebagai hitungan, tergantung `type` pada`ulimit`.

Label docker

`dockerLabels`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta label untuk ditambahkan ke wadah. Parameter ini memetakan ke `Labels` dalam perintah docker create-container dan opsi `--label` untuk docker run.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

```
"dockerLabels": {"string": "string"
      ...}
```

Parameter definisi wadah lainnya

Parameter definisi kontainer berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Topik

- Parameter Linux
- Dependensi kontainer
- Waktu habis kontainer
- Kontrol sistem
- Interaktif
- Terminal semu

Parameter Linux

`linuxParameters`

Tipe: Objek LinuxParameters

Wajib: Tidak

Linux-opsi spesifik yang diterapkan ke wadah, seperti KernelCapabilities.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer.

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

Tipe: Objek KernelCapabilities

Wajib: Tidak

LinuxKemampuan untuk wadah yang dijatuhkan dari konfigurasi default yang disediakan olehDocker. Untuk informasi lebih lanjut tentang Linux kemampuan ini, lihat halaman manual kemampuan (7) Linux.

`add`

Tipe: Array string

Nilai yang valid: `"SYS_PTRACE"`

Wajib: Tidak

LinuxKemampuan kontainer untuk ditambahkan ke konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapAdd` dalam perintah docker create-container dan opsi `--cap-add` untuk docker run.

`drop`

Tipe: Array string

Nilai yang valid: `"ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

Wajib: Tidak

LinuxKemampuan untuk wadah untuk menghapus dari konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapDrop` dalam perintah docker create-container dan opsi `--cap-drop` untuk docker run.

`devices`

Perangkat host apa pun yang akan diekspos ke kontainer. Parameter ini memetakan ke `Devices` dalam perintah docker create-container dan opsi `--device` untuk docker run.

> **Note**
>
> `devices`Parameter tidak didukung saat Anda menggunakan tipe peluncuran Fargate.

Tipe: Array objek Perangkat

Wajib: Tidak

`hostPath`

Jalur untuk perangkat pada instans kontainer host.

Tipe: String

Diperlukan: Ya

`containerPath`

Jalur di dalam wadah untuk mengekspos perangkat host di.

Tipe: String

Wajib: Tidak

`permissions`

Izin eksplisit untuk diberikan ke kontainer untuk perangkat. Secara default, kontainer memiliki izin untuk`read`, `write`, dan `mknod` pada perangkat.

Tipe: Array string

Nilai yang Valid: `read` | `write` | `mknod`

`initProcessEnabled`

Jalankan proses `init` di dalam kontainer yang meneruskan sinyal dan melakukan proses kembali. Parameter ini memetakan ke `--init` opsi untuk docker run.

Parameter ini memerlukan API Docker Remote versi 1.25 atau lebih besar pada instans kontainer Anda.

`maxSwap`

Ini tidak didukung untuk tugas yang berjalan di Fargate.

Jumlah total memori tukar (dalam MiB) yang dapat digunakan oleh kontainer. Parameter ini diterjemahkan ke `--memory-swap` opsi untuk docker run di mana nilainya adalah jumlah memori kontainer ditambah nilainya. `maxSwap`

Jika nilai `maxSwap` sebesar `0` ditentukan, kontainer tidak menggunakan swap. Nilai yang diterima adalah `0` atau bilangan bulat positif. Jika `maxSwap` parameter diabaikan, kontainer

menggunakan konfigurasi swap untuk instans kontainer tempatnya berjalan. Nilai `maxSwap` harus ditetapkan untuk parameter `swappiness` yang akan digunakan.

`sharedMemorySize`

Nilai untuk ukuran (dalam MiB) volume `/dev/shm`. Parameter ini memetakan ke `--shm-size` opsi untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `sharedMemorySize` parameter tidak didukung.

Jenis: Integer

`tmpfs`

Jalur kontainer, opsi pemasangan, dan ukuran maksimum (dalam MiB) dudukan. tmpfs Parameter ini memetakan ke `--tmpfs` opsi untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas yang menggunakan tipe peluncuran Fargate, `tmpfs` parameter tidak didukung.

Tipe: Array objek Tmpfs

Wajib: Tidak

`containerPath`

Jalur file absolut tempat tmpfs volume akan dipasang.

Tipe: String

Diperlukan: Ya

`mountOptions`

Daftar opsi pemasangan volume tmpfs.

Tipe: Array string

Wajib: Tidak

Nilai yang Valid: `"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"`

`size`

Ukuran maksimum (dalam MiB) volume. tmpfs

Jenis: Integer

Wajib: Ya

Dependensi kontainer

`dependsOn`

Tipe: Array objek ContainerDependency

Wajib: Tidak

Dependensi yang ditentukan untuk pemulaian dan penonaktifan kontainer. Kontainer dapat berisi beberapa dependensi. Ketika dependensi ditentukan untuk memulai kontainer, maka akan dibalik jika ingin menonaktifkan kontainer. Misalnya, lihat Dependensi kontainer.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Parameter ini mengharuskan tugas atau layanan menggunakan versi platform `1.3.0` atau yang lebih baru (Linux) atau `1.0.0` (Windows).

```
"dependsOn": [
```

```
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

Tipe: String

Wajib: Ya

Nama kontainer yang harus memenuhi syarat yang ditentukan.

`condition`

Tipe: String

Wajib: Ya

Syarat dependensi kontainer. Berikut ini adalah syarat yang tersedia dan perilakunya:

- `START`Kondisi ini meniru perilaku link dan volume hari ini. Kondisi memvalidasi bahwa kontainer dependen dimulai sebelum mengizinkan kontainer lain untuk memulai.
- `COMPLETE`— Kondisi ini memvalidasi bahwa kontainer dependen berjalan hingga selesai (keluar) sebelum mengizinkan kontainer lain untuk memulai. Ini dapat berguna untuk wadah non-esensial yang menjalankan skrip dan kemudian keluar. Kondisi ini tidak dapat diatur pada wadah penting.
- `SUCCESS`— Kondisi ini sama dengan`COMPLETE`, tetapi juga mengharuskan wadah keluar dengan `zero` status. Kondisi ini tidak dapat diatur pada wadah penting.
- `HEALTHY`- Kondisi ini memvalidasi bahwa wadah dependen melewati pemeriksaan kesehatan kontainer sebelum mengizinkan wadah lain untuk memulai. Ini mengharuskan wadah dependen memiliki pemeriksaan kesehatan yang dikonfigurasi dalam definisi tugas. Syarat ini dikonfirmasi hanya pada memulai tugas.

Waktu habis kontainer

`startTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum menyerah pada penyelesaian dependensi untuk kontainer.

Misalnya, Anda menentukan dua kontainer dalam ketentuan tugas dengan `containerA` yang memiliki dependensi untuk `containerB` yang mencapai status `COMPLETE`, `SUCCESS`, atau `HEALTHY`. Jika `startTimeout` nilai ditentukan untuk `containerB` dan tidak mencapai status yang diinginkan dalam waktu itu, maka `containerA` tidak dimulai.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Parameter ini mengharuskan tugas atau layanan menggunakan versi platform `1.3.0` atau yang lebih baru (Linux). Nilai maksimumnya adalah 120 detik.

`stopTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum kontainer dimatikan secara paksa jika tidak keluar dengan sendirinya secara normal.

Parameter ini mengharuskan tugas atau layanan menggunakan versi platform `1.3.0` atau yang lebih baru (Linux). Jika parameter tidak ditentukan, maka nilai default 30 detik digunakan. Nilai maksimumnya adalah 120 detik.

Kontrol sistem

`systemControls`

Tipe: Objek SystemControl

Wajib: Tidak

Daftar parameter kernel namespace untuk diatur dalam wadah. Parameter ini memetakan ke `Sysctls` dalam perintah docker create-container dan opsi untuk docker run. `--sysctl` Misalnya, Anda dapat mengonfigurasi `net.ipv4.tcp_keepalive_time` pengaturan untuk mempertahankan koneksi yang lebih lama.

Kami tidak menyarankan Anda menentukan `systemControls` parameter terkait jaringan untuk beberapa kontainer dalam satu tugas yang juga menggunakan mode `awsvpc` atau `host` jaringan. Melakukan hal ini memiliki kelemahan sebagai berikut:

- Jika Anda mengatur `systemControls` untuk wadah apa pun, itu berlaku untuk semua kontainer dalam tugas. Jika Anda menyetel berbeda `systemControls` untuk beberapa kontainer dalam satu tugas, penampung yang dimulai terakhir menentukan mana yang `systemControls` berlaku.

Jika Anda menyetel namespace sumber daya IPC untuk digunakan untuk kontainer dalam tugas, kondisi berikut berlaku untuk kontrol sistem Anda. Untuk informasi selengkapnya, lihat Mode IPC.

- Untuk tugas yang menggunakan mode `host` IPC, namespace `systemControls` IPC tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` nilai namespace IPC berlaku untuk semua kontainer dalam tugas.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

> **Note**
>
> Parameter ini hanya didukung untuk tugas yang di-host AWS Fargate jika tugas menggunakan versi platform `1.4.0` atau yang lebih baru (Linux). Ini tidak didukung untuk wadah Windows di Fargate.

```
"systemControls": [
    {
         "namespace":"string",
```

```
        "value":"string"
    }
]
```

`namespace`

Tipe: String

Wajib: Tidak

Parameter kernel namespace untuk menetapkan untuk. `value`

Nilai namespace IPC yang valid:`"kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"`, dan `Sysctls` itu dimulai dengan `"fs.mqueue.*"`

Nilai namespace jaringan yang valid: `Sysctls` yang dimulai dengan. `"net.*"` Di Fargate, hanya namespace yang ada di dalam `Sysctls` container yang diterima.

Semua nilai ini didukung oleh Fargate.

`value`

Tipe: String

Wajib: Tidak

Nilai untuk parameter kernel namespace yang ditentukan dalam. `namespace`

Interaktif

`interactive`

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini`true`, Anda dapat menerapkan aplikasi kontainer yang memerlukan `stdin` atau `tty` dialokasikan. Parameter ini memetakan ke `OpenStdin` dalam perintah docker create-container dan opsi `--interactive` untuk docker run.

Nilai default-nya `false`.

Terminal semu

`pseudoTerminal`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini `true`, TTY dialokasikan. Parameter ini memetakan ke `Tty` dalam perintah docker create-container dan opsi `--tty` untuk docker run.

Nilai default-nya `false`.

Nama akselerator Elastic Inference

Persyaratan sumber daya akselerator Elastic Inference untuk definisi tugas Anda.

> **Note**
>
> Amazon Elastic Inference (EI) tidak lagi tersedia untuk pelanggan.

Parameter berikut diizinkan dalam definisi tugas:

`deviceName`

Tipe: String

Diperlukan: Ya

Nama perangkat akselerator Elastic Inference. Itu juga `deviceName` harus direferensikan dalam definisi kontainer lihat`Elastic Inference accelerator`.

`deviceType`

Tipe: String

Diperlukan: Ya

Akselerator Elastic Inference untuk digunakan.

Konfigurasi proxy

`proxyConfiguration`

Tipe: Objek ProxyConfiguration

Wajib: Tidak

Detail konfigurasi untuk proxy App Mesh.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
           "value": "string"
        }
    ]
}
```

`type`

Tipe: String

Nilai nilai: `APPMESH`

Wajib: Tidak

Tipe proxy. Satu-satunya nilai yang didukung adalah `APPMESH`.

`containerName`

Tipe: String

Diperlukan: Ya

Nama wadah yang berfungsi sebagai proxy App Mesh.

`properties`

Tipe: Array objek KeyValuePair

Wajib: Tidak

Set parameter konfigurasi jaringan berfungsi untuk menyediakan plugin Container Network Interface (CNI), yang ditetapkan sebagai pasangan nilai kunci.

- `IgnoredUID`— (Wajib) ID pengguna (UID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredGID` ditentukan, bidang ini tidak boleh kosong.
- `IgnoredGID`— (Wajib) ID grup (GID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredUID` ditentukan, bidang ini tidak boleh kosong.
- `AppPorts`— (Wajib) Daftar port yang digunakan aplikasi. Lalu lintas jaringan ke port ini diteruskan ke `ProxyIngressPort` dan `ProxyEgressPort`.
- `ProxyIngressPort`— (Wajib) Menentukan port yang lalu lintas masuk ke `AppPorts` diarahkan.
- `ProxyEgressPort`— (Wajib) Menentukan port yang lalu lintas keluar dari `AppPorts` diarahkan ke.
- `EgressIgnoredPorts`— (Wajib) Lalu lintas keluar yang menuju ke port yang ditentukan ini diabaikan dan tidak dialihkan ke. `ProxyEgressPort` Ia bisa berupa daftar kosong.
- `EgressIgnoredIPs`— (Wajib) Lalu lintas keluar yang menuju ke alamat IP yang ditentukan ini diabaikan dan tidak dialihkan ke. `ProxyEgressPort` Ia bisa berupa daftar kosong.

`name`

Tipe: String

Wajib: Tidak

Nama pasangan nilai kunci.

`value`

Tipe: String

Wajib: Tidak

Nilai pasangan nilai kunci.

Volume

Saat Anda mendaftarkan definisi tugas, Anda dapat secara opsional menentukan daftar volume yang akan diteruskan ke Docker daemon pada instance container, yang kemudian tersedia untuk diakses oleh container lain pada instance container yang sama.

Berikut ini adalah jenis volume data yang dapat digunakan:

- Volume Amazon EBS - Menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Anda dapat melampirkan 1 volume Amazon EBS per tugas Amazon ECS saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Volume Amazon EBS didukung untuk tugas-tugas Linux yang dihosting di Fargate. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EBS dengan Amazon ECS.
- Volume Amazon EFS — Menyediakan penyimpanan file yang sederhana, terukur, dan persisten untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan dan ketika mereka membutuhkannya. Volume Amazon EFS didukung untuk tugas yang di-host di Fargate. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.
- FSx untuk volume Windows File Server - Menyediakan server file Microsoft Windows yang dikelola sepenuhnya. Server file ini didukung oleh sistem file Windows. Saat menggunakan FSx untuk Windows File Server bersama dengan Amazon ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Gunakan FSx untuk volume Server File Windows dengan Amazon ECS.

  Wadah Windows di Fargate tidak mendukung opsi ini.
- Bind mount — File atau direktori pada mesin host yang dipasang ke dalam wadah. Volume host Bind mount didukung saat menjalankan tugas. Untuk menggunakan volume host bind mount, tentukan `sourcePath` nilai `host` dan opsional dalam definisi tugas Anda.

Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.

Parameter berikut diizinkan dalam ketentuan kontainer.

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`host`

Wajib: Tidak

`host`Parameter ini digunakan untuk mengikat siklus hidup bind mount ke EC2 instance Amazon host, bukan tugas, dan tempat penyimpanannya. Jika `host` parameternya kosong, maka daemon Docker menetapkan jalur host untuk volume data Anda, tetapi data tidak dijamin akan bertahan setelah wadah yang terkait dengannya berhenti berjalan.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`.

> **Note**
>
> `sourcePath`Parameter hanya didukung saat menggunakan tugas yang di-host di EC2 instans Amazon.

`sourcePath`

Tipe: String

Wajib: Tidak

Saat `host` parameter digunakan, tentukan a `sourcePath` untuk mendeklarasikan jalur pada EC2 instance Amazon host yang disajikan ke wadah. Jika parameter ini kosong, daemon Docker akan menetapkan jalur host untuk Anda. Jika `host` parameter berisi lokasi `sourcePath` file, maka volume data tetap ada di lokasi yang ditentukan pada EC2 instance

Amazon host hingga Anda menghapusnya secara manual. Jika `sourcePath` nilai tidak ada pada EC2 instance Amazon host, daemon Docker membuatnya. Jika lokasinya memang ada, konten dari folder jalur sumber diekspor.

`configuredAtLaunch`

Tipe: Boolean

Wajib: Tidak

Menentukan apakah volume dikonfigurasi saat peluncuran. Saat disetel ke`true`, Anda dapat mengonfigurasi volume saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Saat disetel ke`true`, Anda tidak akan dapat memberikan konfigurasi volume lain dalam definisi tugas. Parameter ini harus diatur `true` untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas. Menyetel `configuredAtLaunch` ke `true` dan menunda konfigurasi volume ke fase peluncuran memungkinkan Anda membuat definisi tugas yang tidak dibatasi untuk jenis volume atau pengaturan volume tertentu. Melakukan hal ini membuat definisi tugas Anda dapat digunakan kembali di seluruh lingkungan eksekusi yang berbeda. Untuk informasi selengkapnya, lihat volume Amazon EBS.

`dockerVolumeConfiguration`

Jenis: DockerVolumeConfigurationObjek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Docker. Volume Docker hanya didukung saat menjalankan tugas pada EC2 instance. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan pemasangan mengikat, tentukan `host` saja.

`scope`

Tipe: String

Nilai Valid: `task` | `shared`

Wajib: Tidak

Cakupan untuk volume Docker, yang menentukan siklus hidupnya. Volume Docker yang tercakup ke `task` secara otomatis disediakan saat tugas dimulai dan dihancurkan saat tugas berhenti. Volume Docker yang tercakup sebagai `shared` dipertahankan setelah tugas berhenti.

`autoprovision`

Jenis: Boolean

Nilai default: `false`

Wajib: Tidak

Jika nilai ini`true`, volume Docker dibuat jika belum ada. Bidang ini hanya digunakan jika `scope` ada`shared`. Jika `scope` ya`task`, maka parameter ini harus dihilangkan.

`driver`

Tipe: String

Wajib: Tidak

Driver volume Docker yang digunakan. Nilai driver harus sesuai dengan nama driver yang disediakan oleh Docker karena nama ini digunakan untuk penempatan tugas. Jika driver diinstal dengan menggunakan CLI plugin Docker, `docker plugin ls` gunakan untuk mengambil nama driver dari instance container Anda. Jika driver diinstal dengan menggunakan metode lain, gunakan penemuan plugin Docker untuk mengambil nama driver.

`driverOpts`

Tipe: String

Wajib: Tidak

Peta opsi khusus driver Docker untuk dilewati. Parameter ini dipetakan ke `DriverOpts` bagian Create a volume dari Docker.

`labels`

Tipe: String

Wajib: Tidak

Metadata kustom untuk ditambahkan ke volume Docker.

`efsVolumeConfiguration`

Jenis: EFSVolumeKonfigurasi Objek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Amazon EFS.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Amazon EFS yang akan digunakan.

`rootDirectory`

Tipe: String

Wajib: Tidak

Direktori dalam sistem file Amazon EFS untuk dipasang sebagai direktori root di dalam host. Jika parameter ini dihilangkan, root volume Amazon EFS akan digunakan. Menentukan `/` memiliki efek yang sama seperti mengabaikan parameter ini.

> **Important**
>
> Jika titik akses EFS ditentukan dalam`authorizationConfig`, parameter direktori root harus dihilangkan atau disetel ke`/`, yang akan menerapkan jalur yang disetel pada titik akses EFS.

`transitEncryption`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan mengaktifkan enkripsi untuk data Amazon EFS dalam perjalanan antara host Amazon ECS dan server Amazon EFS. Jika otorisasi Amazon EFS IAM digunakan, enkripsi transit harus diaktifkan. Jika parameter ini diabaikan, nilai default `DISABLED` akan digunakan. Untuk informasi lebih lanjut, lihat Pengenkripsian Data Saat Transit di Panduan Pengguna Amazon Elastic File System.

`transitEncryptionPort`

Tipe: Integer

Wajib: Tidak

Port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, tugas akan menggunakan strategi pemilihan port yang digunakan oleh helper mount Amazon EFS. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS di Panduan Pengguna Amazon Elastic File System.

`authorizationConfig`

Jenis: EFSAuthorizationConfig Object

Wajib: Tidak

Detail konfigurasi otorisasi untuk sistem file Amazon EFS.

`accessPointId`

Tipe: String

Wajib: Tidak

ID titik akses yang akan digunakan. Jika titik akses ditentukan, nilai direktori root di dalam `efsVolumeConfiguration` harus dihilangkan atau disetel ke`/`, yang akan memberlakukan jalur yang ditetapkan pada titik akses EFS. Jika titik akses digunakan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Untuk informasi lebih lanjut, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

`iam`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat memasang sistem file Amazon EFS. Jika diaktifkan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Jika Anda menghilangkan

parameter ini, nilai default `DISABLED` akan digunakan. Untuk informasi selengkapnya, lihat IAM Role pada Tugas.

`FSxWindowsFileServerVolumeConfiguration`

Jenis: FSxWindowsFileServerVolumeConfigurationObjek

Wajib: Ya

Parameter ini ditentukan saat Anda menggunakan sistem file Amazon FSx untuk Windows File Server untuk penyimpanan tugas.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Windows File Server untuk digunakan. FSx

`rootDirectory`

Tipe: String

Diperlukan: Ya

Direktori dalam sistem file FSx untuk Windows File Server untuk dipasang sebagai direktori root di dalam host.

`authorizationConfig`

`credentialsParameter`

Tipe: String

Diperlukan: Ya

Opsi kredensi otorisasi.

pilihan:

- Nama Sumber Daya Amazon (ARN) dari sebuah AWS Secrets Managerrahasia.
- ARN dari suatu AWS Systems Managerparameter.

`domain`

Tipe: String

Diperlukan: Ya

Nama domain yang sepenuhnya memenuhi syarat yang dihosting oleh direktori AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) atau Direktori EC2 Aktif yang dihosting sendiri.

Tanda

Ketika mendaftarkan ketentuan tugas, Anda secara opsional dapat menentukan tanda metadata yang diterapkan ke ketentuan tugas. Tanda membantu mengategorikan dan mengatur ketentuan tugas Anda. Setiap tanda terdiri dari kunci dan nilai opsional. Anda mendefinisikan keduanya. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.

> **Important**
>
> Jangan menambahkan informasi identitas pribadi atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak AWS layanan, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan untuk data pribadi atau sensitif.

Parameter berikut diperbolehkan dalam objek tanda.

`key`

Tipe: String

Wajib: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Wajib: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

Parameter ketentuan tugas lainnya

Parameter definisi tugas berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Topik

- Penyimpanan sementara
- Mode IPC
- Mode PID
- Injeksi kesalahan

Penyimpanan sementara

`ephemeralStorage`

Tipe: Objek EphemeralStorage

Wajib: Tidak

Jumlah penyimpanan sementara (dalam GB) untuk dialokasikan untuk tugas tersebut. Parameter ini digunakan untuk memperluas jumlah total penyimpanan sementara yang tersedia, di luar jumlah default, untuk tugas yang di-host. AWS Fargate Untuk informasi selengkapnya, lihat the section called “Pemasangan terikat”.

> **Note**
>
> Parameter ini hanya didukung pada versi platform `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` atau yang lebih baru (Windows).

Mode IPC

`ipcMode`

Ini tidak didukung untuk tugas yang berjalan di Fargate.

Tipe: String

Wajib: Tidak

Namespace sumber daya IPC yang digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host`, `task`, atau `none`. Jika `host` ditentukan, maka semua kontainer yang berada dalam tugas yang menentukan mode `host` IPC pada instance kontainer yang sama berbagi sumber daya IPC yang sama dengan instans Amazon EC2 host. Jika `task` ditentukan, semua kontainer yang berada dalam tugas yang ditentukan berbagi sumber daya IPC yang sama. Jika `none` ditentukan, maka sumber daya IPC dalam kontainer tugas bersifat privat dan tidak dibagi dengan kontainer lain dalam tugas atau pada instans kontainer. Jika tidak ada nilai yang ditentukan, maka berbagi namespace sumber daya IPC tergantung pada pengaturan daemon Docker pada instans kontainer.

Jika mode `host` IPC digunakan, ada risiko tinggi eksposur namespace IPC yang tidak diinginkan.

Jika Anda menyetel parameter kernel namespace yang digunakan `systemControls` untuk kontainer dalam tugas, berikut ini berlaku untuk namespace sumber daya IPC Anda.

- Untuk tugas yang menggunakan mode `host` IPC, namespace IPC yang terkait `systemControls` tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` yang berhubungan dengan namespace IPC berlaku untuk semua kontainer dalam tugas.

> **Note**
>
> Parameter ini tidak didukung untuk wadah atau tugas Windows menggunakan tipe peluncuran Fargate.

Mode PID

`pidMode`

Tipe: String

Nilai Valid: `host` | `task`

Wajib: Tidak

Namspace proses yang akan digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host` atau `task`. Pada wadah Linux, satu-satunya nilai yang valid adalah`task`. Misalnya,

pemantauan sidecar mungkin `pidMode` perlu mengakses informasi tentang kontainer lain yang berjalan dalam tugas yang sama.

Jika `task` ditentukan, semua kontainer dalam tugas tertentu berbagi namespace proses yang sama.

Jika tidak ada nilai yang ditentukan, defaultnya adalah namespace pribadi untuk setiap kontainer.

> **Note**
>
> Parameter ini hanya didukung untuk tugas yang di-host AWS Fargate jika tugas menggunakan versi platform `1.4.0` atau yang lebih baru (Linux). Ini tidak didukung untuk wadah Windows di Fargate.

Injeksi kesalahan

`enableFaultInjection`

Jenis: Boolean

Nilai Valid: `true` | `false`

Wajib: Tidak

Jika parameter ini disetel ke`true`, dalam payload tugas, Amazon ECS dan Fargate menerima permintaan injeksi kesalahan dari penampung tugas. Secara default, parameter ini diatur ke`false`.

Parameter definisi tugas Amazon ECS untuk jenis EC2 peluncuran

Definisi tugas dibagi menjadi beberapa bagian terpisah: keluarga tugas, peran tugas AWS Identity and Access Management (IAM), mode jaringan, definisi wadah, volume, batasan penempatan tugas, dan jenis peluncuran. Definisi keluarga dan wadah diperlukan dalam definisi tugas. Sebaliknya, peran tugas, mode jaringan, volume, kendala penempatan tugas, dan jenis peluncuran bersifat opsional.

Anda dapat menggunakan parameter ini dalam file JSON untuk mengonfigurasi definisi tugas Anda.

Berikut ini adalah deskripsi yang lebih rinci untuk setiap parameter definisi tugas untuk jenis EC2 peluncuran.

Rangkaian

`family`

Tipe: String

Diperlukan: Ya

Ketika Anda mendaftarkan ketentuan tugas, Anda memberikan sebuah famili, yang mirip dengan nama untuk beberapa versi ketentuan tugas, yang ditentukan dengan nomor revisi. Definisi tugas pertama yang terdaftar ke dalam keluarga tertentu diberikan revisi 1, dan definisi tugas apa pun yang terdaftar setelah itu diberi nomor revisi berurutan.

Jenis peluncuran

Saat mendaftarkan definisi tugas, Anda dapat menentukan jenis peluncuran yang harus divalidasi oleh Amazon ECS. Jika definisi tugas tidak memvalidasi terhadap kompatibilitas yang ditentukan, pengecualian klien dikembalikan. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Parameter berikut diperbolehkan dalam definisi tugas.

`requiresCompatibilities`

Tipe: Array string

Wajib: Tidak

Nilai yang Valid: `EC2`

Tipe peluncuran untuk memvalidasi ketentuan tugas. Ini memulai pemeriksaan untuk memastikan bahwa semua parameter yang digunakan dalam definisi tugas memenuhi persyaratan jenis peluncuran.

Peran tugas

`taskRoleArn`

Tipe: String

Wajib: Tidak

Saat mendaftarkan definisi tugas, Anda dapat memberikan peran tugas untuk peran IAM yang memungkinkan kontainer dalam izin tugas memanggil AWS APIs yang ditentukan dalam kebijakan terkait atas nama Anda. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Saat Anda meluncurkan AMI Windows Server Amazon ECS yang dioptimalkan, peran IAM untuk tugas di Windows mengharuskan `-EnableTaskIAMRole` opsi disetel. Container Anda juga harus menjalankan beberapa kode konfigurasi untuk menggunakan fitur tersebut. Untuk informasi selengkapnya, lihat Konfigurasi tambahan instans Amazon EC2 Windows.

Peran eksekusi tugas

`executionRoleArn`

Tipe: String

Wajib: Bersyarat

Nama Sumber Daya Amazon (ARN) dari peran eksekusi tugas yang memberikan izin agen penampung Amazon ECS untuk melakukan panggilan AWS API atas nama Anda.

> **Note**
>
> IAM role eksekusi tugas diperlukan sesuai dengan persyaratan tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Mode jaringan

`networkMode`

Tipe: String

Wajib: Tidak

Mode jaringan Docker digunakan untuk kontainer dalam tugas. Untuk tugas Amazon ECS yang di-host di instans Amazon EC2 Linux, nilai yang valid adalahnone,, `bridge`awsvpc, dan. `host` Jika tidak ada mode jaringan yang ditentukan, mode jaringan default adalah`bridge`. Untuk tugas

Amazon ECS yang dihosting di instans Amazon EC2 Windows, nilai yang valid adalah`default`, dan. `awsvpc` Jika tidak ada mode jaringan yang ditentukan, mode `default` jaringan digunakan.

Jika mode jaringan disetel ke`none`, kontainer tugas tidak memiliki konektivitas eksternal dan pemetaan port tidak dapat ditentukan dalam definisi kontainer.

Jika mode jaringan`bridge`, tugas menggunakan jaringan virtual bawaan Docker di Linux, yang berjalan di dalam setiap EC2 instans Amazon yang menghosting tugas. Jaringan virtual bawaan di Linux menggunakan driver jaringan `bridge` Docker.

Jika mode jaringan`host`, tugas menggunakan jaringan host yang melewati jaringan virtual bawaan Docker dengan memetakan port kontainer langsung ke ENI EC2 instance Amazon yang menghosting tugas tersebut. Pemetaan port dinamis tidak dapat digunakan dalam mode jaringan ini. Wadah dalam definisi tugas yang menggunakan mode ini harus menentukan `hostPort` nomor tertentu. Nomor port pada host tidak dapat digunakan oleh banyak tugas. Akibatnya, Anda tidak dapat menjalankan beberapa tugas dengan definisi tugas yang sama pada satu EC2 instans Amazon.

> **Important**
>
> Saat menjalankan tugas yang menggunakan mode `host` jaringan, jangan jalankan kontainer menggunakan pengguna root (UID 0) untuk keamanan yang lebih baik. Sebagai praktik terbaik keamanan, selalu gunakan pengguna non-root.

Jika mode jaringan adalah `awsvpc`, tugas dialokasikan sebagai antarmuka jaringan elastis, dan Anda harus menentukan `NetworkConfiguration` ketika Anda membuat layanan atau menjalankan tugas dengan ketentuan tugas. Untuk informasi selengkapnya, lihat Opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran.

Jika mode jaringan`default`, tugas menggunakan jaringan virtual bawaan Docker di Windows, yang berjalan di dalam setiap EC2 instans Amazon yang menghosting tugas. Jaringan virtual bawaan pada Windows menggunakan driver jaringan `nat` Docker.

Mode `host` dan `awsvpc` jaringan menawarkan kinerja jaringan tertinggi untuk kontainer karena mereka menggunakan tumpukan EC2 jaringan Amazon. Dengan mode `host` dan `awsvpc` jaringan, port kontainer yang terbuka dipetakan langsung ke port host yang sesuai (untuk mode `host` jaringan) atau port antarmuka elastis network yang terpasang (untuk mode `awsvpc` jaringan). Karena itu, Anda tidak dapat menggunakan pemetaan port host dinamis.

Mode jaringan yang diijinkan tergantung pada sistem operasi EC2 instans yang mendasarinya. Jika Linux, mode jaringan apa pun bisa digunakan. Jika Windows, hanya dapat menggunakan mode jaringan `default`, dan `awsvpc`.

Platform runtime

`operatingSystemFamily`

Tipe: String

Wajib: Bersyarat

Standar: LINUX

Saat Anda mendaftarkan definisi tugas, Anda menentukan keluarga sistem operasi.

Nilai yang valid adalah`LINUX`,`WINDOWS_SERVER_2025_FULL`,`WINDOWS_SERVER_2025_CORE`,`WINDOWS_SERVER_202` dan`WINDOWS_SERVER_2019_CORE`,`WINDOWS_SERVER_2016_FULL`,`WINDOWS_SERVER_2004_CORE`, dan`WINDOWS_SERVER_20H2_CORE`.

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika definisi tugas adalah bagian dari layanan, nilai ini harus sesuai dengan `platformFamily` nilai layanan.

`cpuArchitecture`

Tipe: String

Wajib: Bersyarat

Standar: X86_64

Saat Anda mendaftarkan definisi tugas, Anda menentukan arsitektur CPU. Nilai yang valid adalah `X86_64` dan `ARM64`.

Semua definisi tugas yang digunakan dalam layanan harus memiliki nilai yang sama untuk parameter ini.

Ketika Anda memiliki tugas Linux, Anda dapat mengatur nilainya`ARM64`. Untuk informasi selengkapnya, lihat the section called “Definisi tugas untuk beban kerja ARM 64-bit”.

Ukuran tugas

Saat Anda mendaftarkan definisi tugas, Anda dapat menentukan total CPU dan memori yang digunakan untuk tugas tersebut. Hal ini terpisah dari nilai `cpu` dan `memory` pada tingkat ketentuan kontainer. Untuk tugas yang di-host di EC2 instans Amazon, bidang ini bersifat opsional.

> **Note**
>
> Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer. Kami merekomendasikan penentuan sumber daya tingkat kontainer untuk Windows kontainer.

`cpu`

Tipe: String

Wajib: Bersyarat

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Batas tegas unit CPU untuk hadir untuk tugas tersebut. Anda dapat menentukan nilai CPU dalam file JSON sebagai string dalam unit CPU atau virtual CPUs (vCPUs). Misalnya, Anda dapat menentukan nilai CPU baik seperti `1024` dalam unit CPU atau `1 vCPU` di vCPUs. Ketika definisi tugas terdaftar, nilai vCPU dikonversi ke bilangan bulat yang menunjukkan unit CPU.

Bidang ini bersifat opsional. Jika klaster Anda tidak memiliki instance kontainer terdaftar dengan unit CPU yang diminta tersedia, tugas gagal. Nilai yang didukung adalah antara `0.125` v CPUs dan 192 vCPUs.

`memory`

Tipe: String

Wajib: Bersyarat

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Batas memori yang sulit untuk disajikan pada tugas. Anda dapat menentukan nilai memori dalam definisi tugas sebagai string dalam mebibytes (MiB) atau gigabyte (GB). Misalnya, Anda dapat menentukan nilai memori baik seperti `3072` di MiB atau `3 GB` GB. Ketika ketentuan tugas terdaftar, nilai GB dikonversi ke integer menunjukkan MiB.

Bidang ini bersifat opsional dan nilai apapun dapat digunakan. Jika nilai memori tingkat tugas ditentukan, maka nilai memori tingkat kontainer adalah opsional. Jika klaster Anda tidak memiliki instance kontainer terdaftar dengan memori yang diminta tersedia, tugas gagal. Anda dapat memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan tugas Anda sebanyak mungkin memori untuk jenis instans tertentu. Untuk informasi selengkapnya, lihat Memesan memori instans kontainer Amazon ECS Linux.

Definisi kontainer

Saat Anda mendaftarkan definisi tugas, Anda harus menentukan daftar definisi kontainer yang diteruskan ke Docker daemon pada instance container. Parameter berikut diizinkan dalam ketentuan kontainer.

Topik

- Parameter definisi wadah standar
- Parameter definisi wadah tingkat lanjut
- Parameter definisi wadah lainnya

Parameter definisi wadah standar

Parameter ketentuan tugas berikut diperlukan atau digunakan di sebagian besar ketentuan kontainer.

Topik

- Nama
- Citra

- Memori
- Pemetaan pelabuhan
- Kredensyal Repositori Pribadi

Nama

`name`

Tipe: String

Diperlukan: Ya

Nama kontainer. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Jika Anda menautkan beberapa kontainer dalam definisi tugas, `name` satu kontainer dapat dimasukkan ke dalam `links` wadah lain. Ini untuk menghubungkan wadah.

Citra

`image`

Tipe: String

Diperlukan: Ya

Citra yang digunakan untuk memulai kontainer. String ini diteruskan langsung ke Docker daemon. Secara default, gambar dalam Docker Hub registri tersedia. Anda juga dapat menentukan repositori lain dengan salah satu atau*repository-url*/*image*:*tag*. *repository-url*/*image*@*digest* Maksimal 255 huruf (huruf besar dan kecil), angka, tanda hubung, garis bawah, titik dua, titik, garis miring, dan tanda angka diizinkan. Parameter ini memetakan ke `Image` dalam perintah docker create-container dan parameter `IMAGE` dari perintah docker run.

- Ketika tugas baru dimulai, agen kontainer Amazon ECS menarik versi terbaru dari citra yang ditentukan dan tanda untuk kontainer yang akan digunakan. Namun, pembaruan berikutnya ke gambar repositori tidak disebarkan ke tugas yang sudah berjalan.
- Ketika Anda tidak menentukan tag atau intisari di jalur gambar dalam definisi tugas, agen penampung Amazon ECS menarik versi terbaru dari gambar yang ditentukan.
- Namun, pembaruan berikutnya ke gambar repositori tidak disebarkan ke tugas yang sudah berjalan.

- Citra di registri privat didukung. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.
- Gambar di repositori Amazon ECR dapat ditentukan dengan menggunakan konvensi lengkap `registry/repository:tag` atau `registry/repository@digest` penamaan (misalnya, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com /`*`my-web-app`*`:`*`latest`* atau). *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com /`*`my-web-app`*`@`*`sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE`*
- Gambar di repositori resmi Docker Hub menggunakan satu nama (misalnya, `ubuntu` ataumongo).
- Gambar di repositori lain di Docker Hub memenuhi syarat dengan nama organisasi (misalnya,`amazon/amazon-ecs-agent`).
- Gambar di repositori online lainnya memenuhi syarat lebih lanjut dengan nama domain (misalnya, `quay.io/assemblyline/ubuntu`).

`versionConsistency`

Tipe: String

Nilai yang valid: `enabled` | `disabled`

Wajib: Tidak

Menentukan apakah Amazon ECS akan menyelesaikan tag image container yang disediakan dalam definisi container ke image digest. Secara default, perilaku ini adalah`enabled`. Jika Anda menetapkan nilai untuk container sebagai`disabled`, Amazon ECS tidak akan menyelesaikan tag image container ke intisari dan akan menggunakan URI gambar asli yang ditentukan dalam definisi container untuk penerapan. Untuk informasi selengkapnya tentang resolusi gambar kontainer, lihatResolusi gambar kontainer.

Memori

`memory`

Tipe: Integer

Wajib: Tidak

Jumlah (dalam MiB) memori yang akan ditampilkan ke kontainer. Jika kontainer Anda mencoba untuk melebihi memori yang ditentukan di sini, kontainer akan dimatikan. Jumlah total memori

yang disimpan untuk semua kontainer dalam tugas harus lebih rendah dari nilai `memory` tugas, jika ada yang ditentukan. Parameter ini memetakan ke `Memory` dalam perintah docker create-container dan opsi `--memory` untuk docker run.

Anda harus menentukan nilai memori tingkat tugas atau nilai memori tingkat kontainer. Jika Anda menentukan tingkat kontainer `memory` dan `memoryReservation` nilai, nilainya harus lebih besar dari `memory` nilainya. `memoryReservation` Jika Anda menentukan`memoryReservation`, maka nilai tersebut dikurangi dari sumber daya memori yang tersedia untuk instance wadah tempat penampung ditempatkan. Jika tidak, nilai `memory` digunakan.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatMemesan memori instans kontainer Amazon ECS Linux.

`memoryReservation`

Tipe: Integer

Wajib: Tidak

Batas lunak (di MiB) memori untuk menyimpan kontainer. Ketika memori sistem sedang diperdebatkan, Docker upaya untuk menjaga memori kontainer ke batas lunak ini. Namun, wadah Anda dapat menggunakan lebih banyak memori saat dibutuhkan. Wadah dapat menggunakan hingga batas keras yang ditentukan dengan `memory` parameter (jika ada) atau semua memori yang tersedia pada instance kontainer, mana yang lebih dulu. Parameter ini memetakan ke `MemoryReservation` dalam perintah docker create-container dan opsi `--memory-reservation` untuk docker run.

Jika nilai memori tingkat tugas tidak ditentukan, Anda harus menentukan bilangan bulat bukan nol untuk salah satu atau keduanya `memory` atau `memoryReservation` dalam definisi wadah. Jika Anda menentukan keduanya, `memory` harus lebih besar dari `memoryReservation`. Jika

Anda menentukan`memoryReservation`, maka nilai tersebut dikurangi dari sumber daya memori yang tersedia untuk instance wadah tempat penampung ditempatkan. Jika tidak, nilai `memory` digunakan.

Misalnya, kontainer Anda biasanya menggunakan memori 128 MiB, tetapi kadang-kadang meledak hingga 256 MiB memori untuk waktu yang singkat. Anda dapat mengatur 128 MiB, dan batas `memory` keras 300 MiB. `memoryReservation` Konfigurasi ini memungkinkan penampung untuk hanya menyimpan 128 MiB memori dari sumber daya yang tersisa pada instance container. Pada saat yang sama, konfigurasi ini juga memungkinkan wadah untuk menggunakan lebih banyak sumber daya memori bila diperlukan.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Daemon Docker 20.10.0 atau yang lebih baru menyimpan minimal 6 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 6 MiB memori untuk wadah Anda.

Docker 19.03.13-ce atau daemon sebelumnya menyimpan minimal 4 MiB memori untuk sebuah wadah. Jadi, jangan tentukan kurang dari 4 MiB memori untuk wadah Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatMemesan memori instans kontainer Amazon ECS Linux.

Pemetaan pelabuhan

`portMappings`

Tipe: Array objek

Wajib: Tidak

Pemetaan port mengekspos port jaringan kontainer Anda ke dunia luar. ini memungkinkan klien untuk mengakses aplikasi Anda. Ini juga digunakan untuk komunikasi antar kontainer dalam tugas yang sama.

Untuk definisi tugas yang menggunakan mode `awsvpc` jaringan, hanya tentukan`containerPort`. `hostPort`Itu selalu diabaikan, dan port kontainer secara otomatis dipetakan ke port acak bernomor tinggi pada host.

Pemetaan port pada Windows menggunakan alamat gateway `NetNAT` daripada `localhost`. Tidak ada loopback untuk pemetaan port di Windows, jadi Anda tidak dapat mengakses port yang dipetakan kontainer dari host itu sendiri.

Sebagian besar bidang parameter ini (termasuk`containerPort`,`hostPort`,`protocol`) dipetakan ke `PortBindings` dalam perintah create-container docker dan opsi `--publish` untuk docker run. Jika mode jaringan dari definisi tugas diatur ke`host`, port host harus tidak ditentukan atau cocok dengan port kontainer dalam pemetaan port.

> **Note**
>
> Setelah tugas mencapai status `RUNNING` , tugas port kontainer dan host manual dan otomatis akan terlihat di lokasi-lokasi berikut:
>
> - Konsol: bagian Pengikatan jaringan dari deskripsi kontainer untuk tugas yang dipilih.
> - AWS CLI: Bagian `networkBindings` dari output perintah describe-tasks .
> - API: Respons `DescribeTasks` .
> - Metadata: Titik akhir metadata tugas.

`appProtocol`

Tipe: String

Wajib: Tidak

Protokol aplikasi yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect. Kami menyarankan Anda mengatur parameter ini agar konsisten dengan protokol yang digunakan aplikasi Anda. Jika Anda menyetel parameter ini, Amazon ECS akan menambahkan penanganan koneksi khusus protokol ke proxy service connect. Jika Anda menyetel parameter ini, Amazon ECS menambahkan telemetri khusus protokol di konsol Amazon ECS dan. CloudWatch

Jika Anda tidak menetapkan nilai untuk parameter ini, maka TCP digunakan. Namun, Amazon ECS tidak menambahkan telemetri khusus protokol untuk TCP.

Untuk informasi selengkapnya, lihat the section called “ Layanan Connect”.

Nilai protokol yang valid: `"http" | "http2" | "grpc"`

`containerPort`

Jenis: Integer

Diperlukan: Ya, kapan `portMappings` digunakan

Nomor port pada kontainer yang terikat ke port host yang ditentukan pengguna atau secara otomatis ditetapkan.

Untuk tugas yang menggunakan mode `awsvpc` jaringan, Anda gunakan `containerPort` untuk menentukan port yang terbuka.

Misalkan Anda menggunakan kontainer dalam tugas dengan tipe EC2 peluncuran dan Anda menentukan port kontainer dan bukan port host. Kemudian, kontainer Anda secara otomatis menerima port host dalam rentang port sementara. Untuk informasi selengkapnya, lihat `hostPort`. Pemetaan port yang secara otomatis ditetapkan dengan cara ini tidak dihitung terhadap kuota 100 port cadangan dari instance kontainer.

`containerPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada kontainer yang terikat pada rentang port host yang dipetakan secara dinamis.

Anda hanya dapat mengatur parameter ini dengan menggunakan `register-task-definition` API. Opsi ini tersedia dalam `portMappings` parameter. Untuk informasi selengkapnya, lihat register-task-definition dalam Referensi AWS Command Line Interface .

Aturan berikut berlaku saat Anda menentukan`containerPortRange`:

- Anda harus menggunakan mode `bridge` jaringan atau mode `awsvpc` jaringan.
- Parameter ini tersedia untuk sistem operasi Linux dan Windows.
- Instance container harus memiliki setidaknya versi 1.67.0 dari agen kontainer dan setidaknya versi 1.67.0-1 dari paket. `ecs-init`

- Anda dapat menentukan maksimum 100 rentang port untuk setiap kontainer.
- Anda tidak menentukan`hostPortRange`. Nilai `hostPortRange` ditetapkan sebagai berikut:
  - Untuk kontainer dalam tugas dengan mode `awsvpc` jaringan, `hostPort` diatur ke nilai yang sama dengan`containerPort`. Ini adalah strategi pemetaan statis.
  - Untuk kontainer dalam tugas dengan mode `bridge` jaringan, agen Amazon ECS menemukan port host terbuka dari rentang singkat default dan meneruskannya ke docker untuk mengikatnya ke port kontainer.
- Nilai yang `containerPortRange` valid adalah antara 1 dan 65535.
- Sebuah port hanya dapat dimasukkan dalam satu pemetaan port untuk setiap kontainer.
- Anda tidak dapat menentukan rentang port yang tumpang tindih.
- Port pertama dalam kisaran harus kurang dari port terakhir dalam kisaran.
- Dockermerekomendasikan agar Anda mematikan docker-proxy di file konfigurasi Docker daemon ketika Anda memiliki banyak port.

  Untuk informasi selengkapnya, lihat Masalah #11185 di GitHub.

  Untuk informasi tentang cara mematikan docker-proxy di file konfigurasi daemon, lihat Docker daemon Docker di Panduan Pengembang Amazon ECS.

Anda dapat menelepon DescribeTasksuntuk melihat`hostPortRange`, yang merupakan port host yang terikat ke port kontainer.

Rentang port tidak disertakan dalam peristiwa tugas Amazon ECS, yang dikirim ke EventBridge. Untuk informasi selengkapnya, lihat the section called “Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge”.

`hostPortRange`

Tipe: String

Wajib: Tidak

Rentang nomor port pada host yang digunakan dengan pengikatan jaringan. Ini ditugaskan oleh Docker dan dikirimkan oleh agen Amazon ECS.

`hostPort`

Tipe: Integer

Wajib: Tidak

Nomor port pada instans kontainer untuk menyimpan kontainer Anda.

Anda dapat menentukan port host yang tidak dicadangkan untuk pemetaan port kontainer Anda. Ini disebut sebagai pemetaan port host statis. Atau, Anda dapat menghilangkan `hostPort` (atau mengaturnya ke`0`) sambil menentukan a. `containerPort` Container Anda secara otomatis menerima port dalam rentang port sementara untuk sistem operasi dan Docker versi instans kontainer Anda. Ini disebut sebagai pemetaan port host dinamis.

Rentang port fana default Docker versi 1.6.0 dan yang lebih baru tercantum pada instance di bawah. `/proc/sys/net/ipv4/ip_local_port_range` Jika parameter kernel ini tidak tersedia, port sementara default berkisar sejak `49153–65535` digunakan. Jangan mencoba menentukan port host dalam rentang port sementara. Ini karena ini dicadangkan untuk penugasan otomatis. Secara umum, port di bawah `32768` berada di luar jangkauan port fana.

Port cadangan default adalah `22` untuk SSH, Docker port `2375` dan`2376`, dan port agen kontainer Amazon ECS. `51678-51680` Port host apa pun yang sebelumnya ditentukan pengguna untuk tugas yang sedang berjalan juga dicadangkan saat tugas sedang berjalan. Setelah tugas berhenti, port host dilepaskan. Port cadangan saat ini ditampilkan `remainingResources` di describe-container-instances output. Instance kontainer mungkin memiliki hingga 100 port cadangan sekaligus, termasuk port cadangan default. Port yang ditetapkan secara otomatis tidak dihitung terhadap kuota 100 port cadangan.

`name`

Tipe: String

Wajib: Tidak, diperlukan agar Service Connect dan VPC Lattice dikonfigurasi dalam layanan

Nama yang digunakan untuk pemetaan port. Parameter ini hanya berlaku untuk Service Connect dan VPC Lattice. Parameter ini adalah nama yang Anda gunakan dalam konfigurasi Service Connect dan VPC Lattice suatu layanan.

Untuk informasi selengkapnya, lihat Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek.

Dalam contoh berikut, kedua bidang wajib untuk Service Connect dan VPC Lattice digunakan.

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
]
```

`protocol`

Tipe: String

Wajib: Tidak

Protokol yang digunakan untuk pemetaan port. Nilai yang valid adalah `tcp` dan `udp`. Default adalah `tcp`.

> **Important**
>
> Hanya `tcp` didukung untuk Service Connect. Ingat itu `tcp` tersirat jika bidang ini tidak disetel.

> **Important**
>
> Dukungan UDP hanya tersedia pada instance kontainer yang diluncurkan dengan agen kontainer Amazon ECS versi 1.2.0 (seperti `amzn-ami-2015.03.c-amazon-ecs-optimized` AMI) atau yang lebih baru, atau dengan agen kontainer yang telah diperbarui ke versi 1.3.0 atau yang lebih baru. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Jika Anda menentukan port host, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
```

```
]
```

Jika Anda menginginkan port host yang ditetapkan secara otomatis, gunakan sintaks berikut.

```
"portMappings": [
    {
        "containerPort": integer
    }
    ...
]
```

Kredensyal Repositori Pribadi

`repositoryCredentials`

Tipe: Objek RepositoryCredentials

Wajib: Tidak

Kredensyal repositori untuk otentikasi registri pribadi.

Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

`credentialsParameter`

Tipe: String

Diperlukan: Ya, kapan `repositoryCredentials` digunakan

Amazon Resource Name (ARN) dari rahasia yang berisi kredensial repositori privat.

Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

> **Note**
>
> Saat Anda menggunakan Amazon ECS API, AWS CLI, atau AWS SDKs, jika rahasia ada di Wilayah yang sama dengan tugas yang Anda luncurkan, maka Anda dapat menggunakan ARN lengkap atau nama rahasianya. Bila Anda menggunakan AWS Management Console, Anda harus menentukan ARN penuh rahasia.

Berikut ini adalah cuplikan definisi tugas yang menunjukkan parameter yang diperlukan:

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

Parameter definisi wadah tingkat lanjut

Parameter definisi kontainer lanjutan berikut memberikan kemampuan yang diperluas ke perintah docker run yang digunakan untuk meluncurkan container di instance container Amazon ECS Anda.

Topik

- Mulai ulang kebijakan
- Pemeriksaan kondisi
- Lingkungan
- Pengaturan jaringan
- Penyimpanan dan pencatatan
- Keamanan
- Batas sumber daya
- Label docker

Mulai ulang kebijakan

`restartPolicy`

Kebijakan restart kontainer dan parameter konfigurasi terkait. Saat Anda menyiapkan kebijakan restart untuk penampung, Amazon ECS dapat memulai ulang penampung tanpa perlu mengganti tugas. Untuk informasi selengkapnya, lihat Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer.

`enabled`

Jenis: Boolean

Wajib: Ya

Menentukan apakah kebijakan restart diaktifkan untuk wadah.

`ignoredExitCodes`

Jenis: himpunan integer

Wajib: Tidak

Daftar kode keluar yang akan diabaikan Amazon ECS dan tidak mencoba memulai ulang. Anda dapat menentukan maksimum 50 kode keluar kontainer. Secara default, Amazon ECS tidak mengabaikan kode keluar apa pun.

`restartAttemptPeriod`

Tipe: Integer

Wajib: Tidak

Periode waktu (dalam detik) wadah harus dijalankan sebelum restart dapat dicoba. Sebuah wadah dapat dimulai ulang hanya sekali setiap `restartAttemptPeriod` detik. Jika kontainer tidak dapat berjalan untuk periode waktu ini dan keluar lebih awal, itu tidak akan dimulai ulang. Anda dapat mengatur minimal `restartAttemptPeriod` 60 detik dan maksimum `restartAttemptPeriod` 1800 detik. Secara default, kontainer harus berjalan selama 300 detik sebelum dapat dimulai ulang.

Pemeriksaan kondisi

`healthCheck`

Perintah pemeriksaan kesehatan kontainer dan parameter konfigurasi terkait untuk wadah. Untuk informasi selengkapnya, lihat Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer.

`command`

Sebuah array string yang mewakili perintah bahwa wadah berjalan untuk menentukan apakah itu sehat. Array string dapat dimulai dengan `CMD` untuk menjalankan argumen perintah secara

langsung, atau `CMD-SHELL` untuk menjalankan perintah dengan shell default container. Jika tidak ada yang `CMD` ditentukan, digunakan.

Saat mendaftarkan definisi tugas di AWS Management Console, gunakan daftar perintah yang dipisahkan koma. Perintah ini dikonversi ke string setelah definisi tugas dibuat. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

Saat mendaftarkan definisi tugas menggunakan panel AWS Management Console JSON,, atau AWS CLI APIs, lampirkan daftar perintah dalam tanda kurung. Contoh masukan untuk pemeriksaan kesehatan adalah sebagai berikut.

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

Kode keluar 0, tanpa `stderr` output, menunjukkan keberhasilan, dan kode keluar bukan nol menunjukkan kegagalan.

`interval`

Periode waktu (dalam detik) antara setiap pemeriksaan kesehatan. Anda dapat menentukan antara 5 dan 300 detik. Nilai defaultnya adalah 30 detik.

`timeout`

Periode waktu (dalam hitungan detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal. Anda dapat menentukan antara 2 dan 60 detik. Nilai defaultnya adalah 5 detik.

`retries`

Jumlah raktu untuk mencoba kembali pemeriksaan kondisi yang gagal sebelum kontainer dianggap tidak sehat. Anda dapat menentukan antara 1 dan 10 percobaan ulang. Nilai default-nya adalah tiga kali percobaan ulang.

`startPeriod`

Masa tenggang opsional untuk menyediakan waktu kontainer untuk bootstrap sebelum pemeriksaan kesehatan yang gagal dihitung terhadap jumlah maksimum percobaan ulang. Anda dapat menentukan antara 0 dan 300 detik. Secara default, `startPeriod` dinonaktifkan.

Jika pemeriksaan kondisi berhasil dalam `startPeriod`, maka kontainer dianggap sehat dan kegagalan berikutnya dihitung terhadap jumlah maksimum percobaan kembali.

Lingkungan

`cpu`

Tipe: Integer

Wajib: Tidak

Jumlah `cpu` unit cadangan agen kontainer Amazon ECS untuk kontainer. Di Linux, parameter ini dipetakan `CpuShares` di bagian Create a container.

> **Note**
>
> Anda dapat menentukan jumlah unit CPU yang tersedia untuk setiap jenis EC2 instans Amazon. Untuk melakukan ini, kalikan jumlah v yang CPUs terdaftar untuk jenis instance tersebut di halaman detail EC2Instans Amazon dengan 1.024.

Linuxkontainer berbagi unit CPU yang tidak terisi dengan kontainer lain pada instance kontainer dengan rasio yang sama dengan jumlah yang dialokasikan. Misalnya, asumsikan bahwa Anda menjalankan tugas kontainer tunggal pada tipe instans inti tunggal dengan 512 unit CPU yang ditentukan untuk kontainer tersebut. Selain itu, tugas itu adalah satu-satunya tugas yang berjalan pada instance container. Dalam contoh ini, wadah dapat menggunakan 1.024 pangsa unit CPU penuh pada waktu tertentu. Namun, asumsikan bahwa Anda meluncurkan salinan lain dari tugas yang sama pada instance kontainer itu. Setiap tugas dijamin minimal 512 unit CPU bila diperlukan. Demikian pula, jika wadah lain tidak menggunakan CPU yang tersisa, setiap kontainer dapat mengambang ke penggunaan CPU yang lebih tinggi. Namun, jika kedua tugas tersebut 100% aktif sepanjang waktu, mereka terbatas pada 512 unit CPU.

Pada instance Linux container, Docker daemon pada instance container menggunakan nilai CPU untuk menghitung rasio pangsa CPU relatif untuk menjalankan container. Nilai pangsa CPU minimum yang valid yang diizinkan oleh kernel Linux adalah 2, dan nilai pangsa CPU valid maksimum yang diizinkan oleh kernel Linux adalah 262144. Namun, parameter CPU tidak diperlukan, dan Anda dapat menggunakan nilai CPU di bawah dua dan di atas 262144 dalam definisi container Anda. Untuk nilai CPU di bawah dua (termasuk null) dan di atas 262144, perilaku bervariasi berdasarkan versi agen penampung Amazon ECS Anda:

- Versi agen <= 1.1.0: Nilai CPU nol dan nol diteruskan ke Docker 0. Dockerkemudian mengubah nilai ini menjadi 1.024 saham CPU. Nilai CPU dari satu diteruskan ke Docker sebagai satu, yang Linux kernel mengkonversi menjadi dua saham CPU.

- Versi agen >= 1.2.0: Nilai nol, nol, dan CPU dari satu diteruskan Docker sebagai dua pembagian CPU.
- Versi agen >= 1.84.0: Nilai CPU yang lebih besar dari 256 vCPU diteruskan ke Docker 256, yang setara dengan 262144 pangsa CPU.

Pada instance Windows kontainer, kuota CPU diberlakukan sebagai kuota absolut. Windowscontainer hanya memiliki akses ke jumlah CPU tertentu yang ditentukan dalam definisi tugas. Nilai CPU nol atau nol diteruskan ke Docker as`0`. Windowskemudian menafsirkan nilai ini sebagai 1% dari satu CPU.

Untuk contoh selengkapnya, lihat Cara Amazon ECS mengelola sumber daya CPU dan memori.

`gpu`

Tipe: Objek ResourceRequirement

Wajib: Tidak

Jumlah fisik `GPUs` yang dicadangkan agen kontainer Amazon ECS untuk kontainer. Jumlah GPUs cadangan untuk semua kontainer dalam tugas tidak boleh melebihi jumlah yang tersedia GPUs pada instance kontainer tempat tugas diluncurkan. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer.

`Elastic Inference accelerator`

Tipe: Objek ResourceRequirement

Wajib: Tidak

Untuk `InferenceAccelerator` jenisnya, `value` cocok dengan yang `deviceName` `InferenceAccelerator` ditentukan dalam definisi tugas. Untuk informasi selengkapnya, lihat the section called “Nama akselerator Elastic Inference”.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer.

`essential`

Tipe: Boolean

Wajib: Tidak

Misalkan `essential` parameter wadah ditandai sebagai`true`, dan wadah itu gagal atau berhenti karena alasan apa pun. Kemudian, semua wadah lain yang merupakan bagian dari tugas dihentikan. Jika `essential` parameter wadah ditandai sebagai`false`, maka kegagalannya tidak mempengaruhi sisa kontainer dalam suatu tugas. Jika parameter ini dihilangkan, kontainer diasumsikan menjadi penting.

Semua tugas harus memiliki setidaknya satu kontainer penting. Misalkan Anda memiliki aplikasi yang terdiri dari beberapa kontainer. Kemudian, kelompokkan wadah yang digunakan untuk tujuan bersama menjadi komponen, dan pisahkan komponen yang berbeda menjadi beberapa definisi tugas. Untuk informasi selengkapnya, lihat Arsitek aplikasi Anda untuk Amazon ECS.

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> Versi awal agen kontainer Amazon ECS tidak menangani `entryPoint` parameter dengan benar. Jika Anda memiliki masalah menggunakan `entryPoint`, perbarui agen kontainer Anda atau masukkan perintah dan argumen Anda sebagai item array `command` sebagai gantinya.

Tipe: Array string

Wajib: Tidak

Titik masuk yang diteruskan ke wadah.

```
"entryPoint": ["string", ...]
```

`command`

Tipe: Array string

Wajib: Tidak

Perintah yang diteruskan ke kontainer. Parameter ini memetakan ke `Cmd` dalam perintah create-container dan parameter `COMMAND` ke docker run. Jika ada beberapa argumen, pastikan bahwa setiap argumen adalah string terpisah dalam array.

```
"command": ["string", ...]
```

`workingDirectory`

Tipe: String

Wajib: Tidak

Direktori kerja untuk menjalankan perintah di dalam wadah di. Parameter ini sesuai dengan `WorkingDir` di bagian Buat kontainer di Docker Remote API dan opsi `--workdir` untuk docker run.

```
"workingDirectory": "string"
```

`environmentFiles`

Tipe: Array objek

Wajib: Tidak

Daftar file yang berisi variabel lingkungan untuk diteruskan ke kontainer. Parameter ini memetakan ke `--env-file` opsi ke perintah docker run.

Saat FIPS diaktifkan, nama bucket yang memiliki periode (.) (misalnya, amzn-s3-demo-bucket1.name.example) tidak didukung. Memiliki period (.) dalam nama bucket mencegah tugas dimulai karena agen tidak dapat menarik file variabel lingkungan dari Amazon S3.

Ini tidak tersedia untuk Windows kontainer.

Anda dapat menentukan hingga 10 file lingkungan. File harus memiliki ekstensi file `.env`. Setiap baris dalam file lingkungan berisi variabel lingkungan dalam `VARIABLE=VALUE` format. Baris yang dimulai dengan # diperlakukan sebagai komentar dan diabaikan.

Jika ada variabel lingkungan individu yang ditentukan dalam ketentuan kontainer, maka akan lebih diutamakan daripada variabel yang ada di dalam file lingkungan. Jika beberapa file

lingkungan ditentukan yang berisi variabel yang sama, mereka diproses dari atas ke bawah. Kami merekomendasikan agar Anda menggunakan nama variabel yang unik. Untuk informasi selengkapnya, lihat Meneruskan variabel lingkungan individual ke wadah Amazon ECS.

`value`

Tipe: String

Wajib: Ya

Amazon Resource Name (ARN) dari objek Amazon S3 yang berisi variabel lingkungan file.

`type`

Tipe: String

Wajib: Ya

Tipe file yang akan digunakan. Satu-satunya nilai yang didukung adalah `s3`.

`environment`

Tipe: Array objek

Wajib: Tidak

Variabel lingkungan untuk disampaikan ke kontainer. Parameter ini memetakan ke `Env` dalam perintah docker create-container dan opsi `--env` ke perintah docker run.

> **Important**
>
> Kami tidak menyarankan menggunakan variabel lingkungan teks biasa untuk informasi sensitif, seperti data kredensial.

`name`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nama variabel lingkungan.

`value`

Tipe: String

Diperlukan: Ya, saat `environment` digunakan

Nilai dari variabel lingkungan.

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

`secrets`

Tipe: Object array

Wajib: Tidak

Objek yang mewakili rahasia untuk mengekspos ke wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk mengekspos ke kontainer. Nilai yang didukung adalah Nama Sumber Daya Amazon (ARN) lengkap dari AWS Secrets Manager rahasia atau ARN lengkap parameter di Parameter Store. AWS Systems Manager

> **Note**
>
> Jika parameter Systems Manager Parameter Store atau parameter Secrets Manager Wilayah AWS sama dengan tugas yang Anda luncurkan, Anda dapat menggunakan

ARN lengkap atau nama rahasia. Jika parameter ada di Wilayah yang berbeda, ARN lengkap harus disebutkan.

```
"secrets": [
    {
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

Pengaturan jaringan

`disableNetworking`

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini benar, jaringan mati di dalam wadah.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

Nilai default-nya `false`.

```
"disableNetworking": true|false
```

`links`

Tipe: Array string

Wajib: Tidak

Parameter `link` mengizinkan kontainer untuk berkomunikasi satu sama lain tanpa perlu port pemetaan. Parameter ini hanya didukung jika mode jaringan dari definisi tugas diatur ke`bridge`.

`name:internalName`Konstruk ini analog dengan dalam tautan. `name:alias` Docker Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung, dan garis bawah diperbolehkan..

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

> **Important**
>
> Kontainer yang ditempatkan pada instance kontainer yang sama dapat berkomunikasi satu sama lain tanpa memerlukan tautan atau pemetaan port host. Isolasi jaringan pada instans kontainer dikendalikan oleh grup keamanan dan pengaturan VPC.

```
"links": ["name:internalName", ...]
```

`hostname`

Tipe: String

Wajib: Tidak

Nama host yang digunakan untuk kontainer Anda. Parameter ini memetakan ke `Hostname` dalam docker create-container dan opsi `--hostname` untuk docker run.

> **Note**
>
> Jika Anda menggunakan mode `awsvpc` jaringan, `hostname` parameter tidak didukung.

```
"hostname": "string"
```

`dnsServers`

Tipe: Array string

Wajib: Tidak

Daftar server DNS yang disajikan untuk kontainer.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"dnsServers": ["string", ...]
```

`dnsSearchDomains`

Tipe: Array string

Wajib: Tidak

Pola: ^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$

Daftar domain pencarian DNS yang disajikan untuk kontainer. Parameter ini memetakan ke `DnsSearch` dalam perintah docker create-container opsi `--dns-search` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah Windows atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"dnsSearchDomains": ["string", ...]
```

`extraHosts`

Tipe: Array objek

Wajib: Tidak

Daftar nama host dan pemetaan alamat IP untuk ditambahkan ke file `/etc/hosts` di kontainer.

Parameter ini memetakan ke `ExtraHosts` dalam perintah docker create-container dan opsi `--add-host` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah Windows atau tugas yang menggunakan mode `awsvpc` jaringan.

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
      }
      ...
    ]
```

`hostname`

Tipe: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Nama host yang akan digunakan dalam entri `/etc/hosts`.

`ipAddress`

Tipe: String

Diperlukan: Ya, kapan `extraHosts` digunakan

Alamat IP yang akan digunakan dalam `/etc/hosts` entri.

Penyimpanan dan pencatatan

`readonlyRootFilesystem`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini betul, kontainer diberikan akses hanya-baca ke sistem file asalnya. Parameter ini memetakan ke `ReadonlyRootfs` dalam perintah docker create-container opsi `--read-only` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Nilai default-nya `false`.

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

Tipe: Array objek

Wajib: Tidak

Titik pemasangan untuk volume data dalam penampung Anda. Parameter ini memetakan ke `Volumes` dalam create-container Docker API dan opsi `--volume` untuk docker run.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`. Kontainer Windows tidak dapat memasang direktori pada drive yang berbeda, dan titik pemasangan tidak dapat digunakan di seluruh drive. Anda harus menentukan titik pemasangan untuk melampirkan volume Amazon EBS langsung ke tugas Amazon ECS.

`sourceVolume`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Nama volume yang akan dipasang.

`containerPath`

Tipe: String

Diperlukan: Ya, kapan `mountPoints` digunakan

Jalur dalam wadah tempat volume akan dipasang.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

Untuk tugas yang menjalankan sistem operasi Windows, biarkan nilainya sebagai default`false`.

`volumesFrom`

Tipe: Array objek

Wajib: Tidak

Volume data untuk memasang dari kontainer lain. Parameter ini memetakan ke `VolumesFrom` dalam perintah docker create-container dan opsi `--volumes-from` untuk docker run.

`sourceContainer`

Tipe: String

Diperlukan: Ya, saat `volumesFrom` digunakan

Nama kontainer untuk memasang volume.

`readOnly`

Tipe: Boolean

Wajib: Tidak

Jika nilai ini adalah `true`, kontainer memiliki akses hanya-baca ke volume. Jika nilai ini adalah `false`, maka kontainer dapat menulis ke volume. Nilai default-nya adalah `false`.

```
"volumesFrom": [
                {
                  "sourceContainer": "string",
                  "readOnly": true|false
                }
              ]
```

`logConfiguration`

Jenis: LogConfigurationObjek

Wajib: Tidak

Spesifikasi konfigurasi log untuk kontainer.

Misalnya definisi tugas yang menggunakan konfigurasi log, lihatContoh definisi tugas Amazon ECS.

Parameter ini memetakan ke `LogConfig` dalam perintah docker create-container dan opsi `--log-driver` untuk docker run. Secara default, kontainer menggunakan driver logging yang sama dengan yang digunakan Docker daemon. Namun, penampung mungkin menggunakan driver logging yang berbeda dari Docker daemon dengan menentukan driver log dengan parameter ini dalam definisi container. Untuk menggunakan driver penlog yang berbeda untuk kontainer, sistem log harus dikonfigurasi dengan benar pada instans kontainer (atau pada server log yang berbeda untuk opsi penlog jarak jauh).

Pertimbangkan hal berikut saat menentukan konfigurasi log untuk kontainer Anda:

- Amazon ECS mendukung subset driver logging yang tersedia untuk daemon. Docker Driver log tambahan mungkin tersedia di rilis mendatang agen kontainer Amazon ECS.
- Parameter ini memerlukan Docker Remote API versi 1.18 atau yang lebih baru pada instance container Anda.
- Agen penampung Amazon ECS yang berjalan pada instance container harus mendaftarkan driver logging yang tersedia pada instance tersebut dengan variabel `ECS_AVAILABLE_LOGGING_DRIVERS` lingkungan sebelum container yang ditempatkan pada instance tersebut dapat menggunakan opsi konfigurasi log ini. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","splunk","syslog","awsfirelens",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

`logDriver`

Tipe: String

Nilai yang valid: `"awslogs","fluentd","gelf","json-file","journald","splunk","syslog","awsfirelens"`

Diperlukan: Ya, saat `logConfiguration` digunakan

Driver log yang digunakan untuk kontainer. Secara default, nilai valid yang tercantum sebelumnya adalah driver log yang dapat berkomunikasi dengan agen penampung Amazon ECS.

Driver log yang didukung adalah `awslogs`, `fluentd`, `gelf`, `json-file`, `journald`, `syslog`, `splunk`, dan `awsfirelens`.

Untuk informasi selengkapnya tentang cara menggunakan driver `awslogs` log dalam definisi tugas untuk mengirim log kontainer Anda ke CloudWatch Log, lihatKirim log Amazon ECS ke CloudWatch .

Untuk informasi selengkapnya tentang penggunaan driver log `awsfirelens`, lihat Perutean Log Kustom.

> **Note**
>
> Jika Anda memiliki driver khusus yang tidak terdaftar, Anda dapat melakukan fork proyek agen penampung Amazon ECS yang tersedia GitHub dan menyesuaikannya agar berfungsi dengan driver itu. Kami mendorong Anda untuk mengirim permintaan tarik untuk perubahan yang ingin Anda sertakan. Namun, saat ini kami tidak mendukung menjalankan salinan modifikasi dari perangkat lunak ini.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

`options`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta opsi konfigurasi untuk dikirim ke driver log.

Opsi yang dapat Anda tentukan tergantung pada driver log. Beberapa opsi yang dapat Anda tentukan saat Anda menggunakan `awslogs` router untuk merutekan log ke Amazon CloudWatch meliputi yang berikut:

`awslogs-create-group`

Wajib: Tidak

Tentukan apakah Anda ingin grup log dibuat secara otomatis. Jika opsi ini tidak ditentukan, default-nya adalah `false`.

> **Note**
>
> Kebijakan IAM Anda harus menyertakan `logs:CreateLogGroup` izin sebelum Anda mencoba menggunakannya`awslogs-create-group`.

`awslogs-region`

Wajib: Ya

Tentukan Wilayah AWS driver `awslogs` log untuk mengirim log Docker Anda. Anda dapat memilih untuk mengirim semua log Anda dari kluster di Wilayah yang berbeda ke satu wilayah di CloudWatch Log. Ini agar mereka semua terlihat di satu lokasi. Jika tidak, Anda dapat memisahkannya berdasarkan Wilayah untuk perincian lebih lanjut. Pastikan bahwa grup log yang ditentukan ada di Wilayah yang Anda tentukan dengan opsi ini.

`awslogs-group`

Wajib: Ya

Pastikan untuk menentukan grup log tempat driver `awslogs` log mengirimkan aliran lognya.

`awslogs-stream-prefix`

Diperlukan: Opsional

Gunakan `awslogs-stream-prefix` opsi untuk mengaitkan aliran log dengan awalan yang ditentukan, nama penampung, dan ID tugas Amazon ECS tempat penampung tersebut berada. Jika Anda menentukan awalan dengan opsi ini, maka aliran log mengambil format berikut.

```
prefix-name/container-name/ecs-task-id
```

Jika Anda tidak menentukan awalan dengan opsi ini, maka aliran log dinamai sesuai dengan ID penampung yang ditetapkan oleh daemon Docker pada instance container. Karena sulit untuk melacak log kembali ke wadah yang mengirimnya hanya dengan ID wadah Docker (yang hanya tersedia pada instance penampung), kami sarankan Anda menentukan awalan dengan opsi ini.

Untuk layanan Amazon ECS, Anda dapat menggunakan nama layanan sebagai awalan. Dengan melakukannya, Anda dapat melacak aliran log ke layanan tempat penampung milik, nama penampung yang mengirimnya, dan ID tugas yang menjadi milik penampung tersebut.

Anda harus menentukan awalan aliran agar log Anda muncul di panel Log saat menggunakan konsol Amazon ECS.

`awslogs-datetime-format`

Wajib: Tidak

Opsi ini mendefinisikan pola mulai multiline di Python format `strftime`. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Salah satu contoh kasus penggunaan format ini adalah untuk penguraian output seperti buangan tumpukan, yang mungkin bisa tercatat dalam beberapa entri. Pola yang benar akan memungkinkannya ditangkap dalam satu entri.

Untuk informasi selengkapnya, lihat awslogs-datetime-format.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

`awslogs-multiline-pattern`

Wajib: Tidak

Opsi ini mendefinisikan pola mulai multiline yang menggunakan ekspresi reguler. Pesan log terdiri dari baris yang cocok dengan pola dan baris berikut yang tidak cocok dengan pola. Baris yang cocok adalah pembatas antara pesan log.

Untuk informasi selengkapnya, lihat awslogs-multiline-pattern.

Opsi ini diabaikan jika `awslogs-datetime-format` juga dikonfigurasi.

Anda tidak dapat mengonfigurasi opsi `awslogs-datetime-format` dan `awslogs-multiline-pattern` opsi.

> **Note**
>
> Pencatatan multibaris melakukan penguraian dan pencocokan ekspresi reguler dari semua pesan log. Ini mungkin berdampak negatif pada performa pencatatan.

Opsi berikut berlaku untuk semua driver log yang didukung.

`mode`

Wajib: Tidak

Nilai yang valid: `non-blocking` | `blocking`

Opsi ini mendefinisikan mode pengiriman pesan log dari wadah ke driver log yang ditentukan menggunakan`logDriver`. Mode pengiriman yang Anda pilih memengaruhi ketersediaan aplikasi saat aliran log dari kontainer terganggu.

Jika Anda menggunakan `blocking` mode dan aliran log terputus, panggilan dari kode kontainer untuk menulis ke `stdout` dan `stderr` aliran akan diblokir. Thread logging aplikasi akan diblokir sebagai hasilnya. Hal ini dapat menyebabkan aplikasi menjadi tidak responsif dan menyebabkan kegagalan pemeriksaan kesehatan kontainer.

Jika Anda menggunakan `non-blocking` mode, log kontainer malah disimpan dalam buffer perantara dalam memori yang dikonfigurasi dengan opsi. `max-buffer-size` Ini mencegah aplikasi menjadi tidak responsif ketika log tidak dapat dikirim. Kami merekomendasikan menggunakan mode ini jika Anda ingin memastikan ketersediaan layanan dan baik-baik saja dengan beberapa kehilangan log. Untuk informasi selengkapnya, lihat Mencegah kehilangan log dengan mode non-pemblokiran di driver log `awslogs` kontainer.

Anda dapat mengatur default `mode` untuk semua kontainer tertentu Wilayah AWS dengan menggunakan pengaturan `defaultLogDriverMode` akun. Jika Anda tidak menentukan `mode` opsi di `logConfiguration` atau mengonfigurasi pengaturan akun, Amazon ECS akan default ke `non-blocking` mode. Untuk informasi selengkapnya tentang pengaturan akun, lihatMode driver log default.

Saat `non-blocking` mode digunakan, opsi `max-buffer-size` log mengontrol ukuran buffer yang digunakan untuk penyimpanan pesan perantara. Pastikan untuk menentukan ukuran buffer yang memadai berdasarkan aplikasi Anda. Jumlah total memori yang dialokasikan pada tingkat tugas harus lebih besar dari jumlah memori yang dialokasikan untuk semua wadah selain buffer memori driver log.

> **Note**
>
> Pada 25 Juni 2025, Amazon ECS mengubah mode driver log default dari menjadi `blocking` `non-blocking` memprioritaskan ketersediaan tugas daripada pencatatan. Untuk terus menggunakan `blocking` mode setelah perubahan ini, lakukan salah satu hal berikut:
>
> - Tetapkan `mode` opsi dalam definisi kontainer Anda `logConfiguration` sebagai`blocking`.
> - Atur pengaturan `defaultLogDriverMode` akun ke`blocking`.

`max-buffer-size`

Wajib: Tidak

Nilai default: `10 m`

Saat `non-blocking` mode digunakan, opsi `max-buffer-size` log mengontrol ukuran buffer yang digunakan untuk penyimpanan pesan perantara. Pastikan untuk menentukan ukuran buffer yang memadai berdasarkan aplikasi Anda. Ketika buffer terisi, log lebih lanjut tidak dapat disimpan. Log yang tidak dapat disimpan hilang.

Untuk merutekan `splunk` log menggunakan router log, Anda perlu menentukan a `splunk-token` dan `asplunk-url`.

Saat Anda menggunakan router `awsfirelens` log untuk merutekan log ke AWS Partner Network tujuan Layanan AWS atau penyimpanan log dan analitik, Anda dapat mengatur `log-driver-buffer-limit` opsi untuk membatasi jumlah peristiwa yang di-buffer dalam memori, sebelum dikirim ke wadah router log. Ini dapat membantu menyelesaikan potensi masalah kehilangan log karena throughput yang tinggi dapat mengakibatkan memori habis untuk buffer di dalamnya. Docker Untuk informasi selengkapnya, lihat the section called “Mengkonfigurasi log untuk throughput tinggi”.

Opsi lain yang dapat Anda tentukan saat menggunakan `awsfirelens` untuk merutekan log tergantung pada tujuan. Saat mengekspor log ke Amazon Data Firehose, Anda dapat menentukan Wilayah AWS dengan `region` dan nama untuk aliran log. `delivery_stream`

Saat Anda mengekspor log ke Amazon Kinesis Data Streams, Anda dapat Wilayah AWS menentukan dengan dan `region` nama aliran data dengan. `stream`

Saat mengekspor log ke Amazon OpenSearch Service, Anda dapat menentukan opsi sepertiName, `Host` (Titik akhir OpenSearch layanan tanpa protokol),,,`Port`,`Index`,`Type`,`Aws_auth`, `Aws_regionSuppress_Type_Name`, dan`tls`.

Saat mengekspor log ke Amazon S3, Anda dapat menentukan bucket menggunakan opsi. `bucket` Anda juga dapat menentukan`region`,`total_file_size`,`upload_timeout`, dan `use_put_object` sebagai opsi.

Parameter ini memerlukan API Docker Remote versi 1.19 atau lebih besar pada instans kontainer Anda.

`secretOptions`

Tipe: Array objek

Wajib: Tidak

Objek yang mewakili rahasia untuk diteruskan ke konfigurasi log. Rahasia yang digunakan dalam konfigurasi log dapat mencakup token otentikasi, sertifikat, atau kunci enkripsi. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

`name`

Tipe: String

Wajib: Ya

Nilai untuk ditetapkan sebagai variabel lingkungan pada kontainer.

`valueFrom`

Tipe: String

Wajib: Ya

Secret untuk diekspos ke konfigurasi log kontainer.

```
"logConfiguration": {
```

```
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
}
```

`firelensConfiguration`

Jenis: FirelensConfigurationObjek

Wajib: Tidak

FireLens Konfigurasi untuk wadah. Digunakan untuk menentukan dan mengonfigurasi router log untuk log kontainer. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
            "KeyName": ""
        }
    }
}
```

`options`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta opsi untuk digunakan saat mengkonfigurasi router log. Bidang ini opsional dan dapat digunakan untuk menentukan file konfigurasi kustom atau untuk menambahkan metadata tambahan, seperti tugas, ketentuan tugas, klaster, dan detail instans kontainer ke log acara. Jika ditentukan, sintaksis yang digunakan adalah `"options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-`

`value":"arn:aws:s3:::amzn-s3-demo-bucket/fluent.conf|filepath"}`. Untuk informasi selengkapnya, lihat Contoh definisi tugas Amazon ECS: Rute log ke FireLens.

`type`

Tipe: String

Wajib: Ya

Router log yang akan digunakan. Nilai yang valid adalah `fluentd` atau `fluentbit`.

Keamanan

Untuk informasi selengkapnya tentang keamanan kontainer, lihat praktik terbaik tugas Amazon ECS dan keamanan kontainer.

`credentialSpecs`

Tipe: Array string

Wajib: Tidak

Daftar ARNs di SSM atau Amazon S3 ke file credential spec `CredSpec` () yang mengonfigurasi wadah untuk otentikasi Active Directory. Kami menyarankan Anda menggunakan parameter ini alih-alih`dockerSecurityOptions`. Jumlah maksimum ARNs adalah 1.

Ada dua format untuk setiap ARN.

CredentialSpecDomainless:mYarn

Anda gunakan `credentialspecdomainless:MyARN` untuk `CredSpec` menyediakan bagian tambahan untuk rahasia di Secrets Manager. Anda memberikan kredensyal login ke domain secara rahasia.

Setiap tugas yang berjalan pada instance kontainer apa pun dapat bergabung dengan domain yang berbeda.

Anda dapat menggunakan format ini tanpa menggabungkan instance container ke domain.

Spesifikasi kredensial:mYarn

Anda gunakan `credentialspec:MyARN` untuk menyediakan `CredSpec` untuk satu domain.

Anda harus menggabungkan instance penampung ke domain sebelum memulai tugas apa pun yang menggunakan definisi tugas ini.

Dalam kedua format, ganti `MyARN` dengan ARN di SSM atau Amazon S3.

`credspec`Harus menyediakan ARN di Secrets Manager untuk rahasia yang berisi nama pengguna, kata sandi, dan domain untuk terhubung. Untuk keamanan yang lebih baik, instance tidak digabungkan ke domain untuk otentikasi tanpa domain. Aplikasi lain pada instance tidak dapat menggunakan kredensyal tanpa domain. Anda dapat menggunakan parameter ini untuk menjalankan tugas pada instance yang sama, bahkan tugas harus bergabung dengan domain yang berbeda. Untuk informasi selengkapnya, lihat Menggunakan g MSAs untuk Kontainer Windows dan Menggunakan g MSAs untuk Kontainer Linux.

`privileged`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini betul, kontainer diberikan hak istimewa yang lebih tinggi pada instans kontainer host misalnya (mirip dengan pengguna `root`). Kami merekomendasikan agar tidak menjalankan kontainer dengan`privileged`. Dalam kebanyakan kasus, Anda dapat menentukan hak istimewa yang tepat yang Anda butuhkan dengan menggunakan parameter tertentu alih-alih menggunakan`privileged`.

Parameter ini memetakan ke `Privileged` dalam perintah docker create-container dan opsi `--privileged` untuk docker run.

> **Note**
>
> Parameter ini tidak didukung untuk wadah atau tugas Windows menggunakan tipe peluncuran Fargate.

Default-nya adalah `false`.

```
"privileged": true|false
```

`user`

Tipe: String

Wajib: Tidak

Pengguna yang akan digunakan di dalam kontainer. Parameter ini memetakan ke `User` dalam perintah docker create-container dan opsi `--user` untuk docker run.

> **Important**
>
> Saat menjalankan tugas yang menggunakan mode `host` jaringan, jangan jalankan kontainer menggunakan pengguna root (UID 0). Sebagai praktik terbaik keamanan, selalu gunakan pengguna non-root.

Anda dapat menentukan `user` dengan menggunakan format berikut. Jika menentukan UID atau GID, Anda harus menentukannya sebagai bilangan bulat positif.

- `user`
- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"user": "string"
```

`dockerSecurityOptions`

Tipe: Array string

Nilai yang valid: "no-new-privileges" | “apparmor:profile” | “label:*value*" | “credentialspec:” *CredentialSpecFilePath*

Wajib: Tidak

Daftar string untuk menyediakan konfigurasi khusus untuk beberapa sistem keamanan.

Untuk Linux tugas, parameter ini dapat digunakan untuk mereferensikan label khusus untuk SELinux dan sistem keamanan AppArmor multi-level.

Parameter ini dapat digunakan untuk mereferensikan file spesifikasi kredensyal yang mengonfigurasi wadah untuk otentikasi Active Directory. Untuk informasi selengkapnya, lihat Pelajari cara menggunakan g MSAs untuk wadah EC2 Windows untuk Amazon ECS dan Menggunakan gMSA untuk EC2 Linux wadah di Amazon ECS.

Parameter ini memetakan ke `SecurityOpt` dalam perintah docker create-container dan opsi `--security-opt` untuk docker run.

```
"dockerSecurityOptions": ["string", ...]
```

> **Note**
>
> Agen penampung Amazon ECS yang berjalan pada instance container harus mendaftar dengan variabel `ECS_APPARMOR_CAPABLE=true` lingkungan `ECS_SELINUX_CAPABLE=true` atau sebelum container yang ditempatkan pada instance tersebut dapat menggunakan opsi keamanan ini. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Batas sumber daya

`ulimits`

Tipe: Array objek

Wajib: Tidak

Daftar nilai `ulimit` yang harus ditentukan untuk kontainer. Nilai ini menimpa pengaturan kuota sumber daya default untuk sistem operasi. Parameter ini memetakan ke `Ulimits` dalam perintah docker create-container dan opsi `--ulimit` untuk docker run.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"ulimits": [
```

```
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
    ]
```

`name`

Tipe: String

Nilai yang valid: `"core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"`

Diperlukan: Ya, kapan `ulimits` digunakan

`type` dari `ulimit`.

`hardLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas keras untuk tipe `ulimit`. Nilai dapat ditentukan dalam byte, detik, atau sebagai hitungan, tergantung `type` pada`ulimit`.

`softLimit`

Jenis: Integer

Diperlukan: Ya, kapan `ulimits` digunakan

Batas lunak untuk tipe `ulimit`. Nilai dapat ditentukan dalam byte, detik, atau sebagai hitungan, tergantung `type` pada`ulimit`.

Label docker

`dockerLabels`

Tipe: Peta antar string

Wajib: Tidak

key/value Peta label untuk ditambahkan ke wadah. Parameter ini memetakan ke `Labels` dalam perintah docker create-container dan opsi `--label` untuk docker run.

Parameter ini memerlukan Docker Remote API versi 1.18 atau lebih besar pada instans kontainer Anda.

```
"dockerLabels": {"string": "string"
      ...}
```

Parameter definisi wadah lainnya

Parameter definisi kontainer berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Topik

- Parameter Linux
- Dependensi kontainer
- Waktu habis kontainer
- Kontrol sistem
- Interaktif
- Terminal semu

Parameter Linux

`linuxParameters`

Tipe: Objek LinuxParameters

Wajib: Tidak

Linux-opsi spesifik yang diterapkan ke wadah, seperti KernelCapabilities.

> **Note**
>
> Parameter ini tidak didukung untuk Windows kontainer.

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

Tipe: Objek KernelCapabilities

Wajib: Tidak

LinuxKemampuan untuk wadah yang ditambahkan ke atau dijatuhkan dari konfigurasi default yang disediakan olehDocker. Untuk informasi lebih lanjut tentang Linux kemampuan ini, lihat halaman manual kemampuan (7) Linux.

`add`

Tipe: Array string

Nilai yang valid: `"ALL" | "AUDIT_CONTROL" | "AUDIT_READ" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

Wajib: Tidak

LinuxKemampuan wadah untuk ditambahkan ke konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapAdd` dalam perintah docker create-container dan opsi `--cap-add` untuk docker run.

`add`

Tipe: Array string

Nilai yang valid: `"SYS_PTRACE"`

Wajib: Tidak

LinuxKemampuan kontainer untuk ditambahkan ke konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapAdd` dalam perintah docker create-container dan opsi `--cap-add` untuk docker run.

`drop`

Tipe: Array string

Nilai yang valid: `"ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

Wajib: Tidak

LinuxKemampuan untuk wadah untuk menghapus dari konfigurasi default yang disediakan olehDocker. Parameter ini memetakan ke `CapDrop` dalam perintah docker create-container dan opsi `--cap-drop` untuk docker run.

`devices`

Perangkat host apa pun yang akan diekspos ke kontainer. Parameter ini memetakan ke `Devices` dalam perintah docker create-container dan opsi `--device` untuk docker run.

Tipe: Array objek Perangkat

Wajib: Tidak

`hostPath`

Jalur untuk perangkat pada instans kontainer host.

Tipe: String

Diperlukan: Ya

`containerPath`

Jalur di dalam wadah untuk mengekspos perangkat host di.

Tipe: String

Wajib: Tidak

`permissions`

Izin eksplisit untuk diberikan ke kontainer untuk perangkat. Secara default, kontainer memiliki izin untuk`read`, `write`, dan `mknod` pada perangkat.

Tipe: Array string

Nilai yang Valid: `read` | `write` | `mknod`

`initProcessEnabled`

Jalankan proses `init` di dalam kontainer yang meneruskan sinyal dan melakukan proses kembali. Parameter ini memetakan ke `--init` opsi untuk docker run.

Parameter ini memerlukan API Docker Remote versi 1.25 atau lebih besar pada instans kontainer Anda.

`maxSwap`

Jumlah total memori tukar (dalam MiB) yang dapat digunakan oleh kontainer. Parameter ini diterjemahkan ke `--memory-swap` opsi untuk docker run di mana nilainya adalah jumlah memori kontainer ditambah nilainya. `maxSwap`

Jika nilai `maxSwap` sebesar `0` ditentukan, kontainer tidak menggunakan swap. Nilai yang diterima adalah `0` atau bilangan bulat positif. Jika `maxSwap` parameter diabaikan, kontainer menggunakan konfigurasi swap untuk instans kontainer tempatnya berjalan. Nilai `maxSwap` harus ditetapkan untuk parameter `swappiness` yang akan digunakan.

`sharedMemorySize`

Nilai untuk ukuran (dalam MiB) volume `/dev/shm`. Parameter ini memetakan ke `--shm-size` opsi untuk docker run.

Jenis: Integer

`swappiness`

Anda dapat menggunakan parameter ini untuk menyetel perilaku pertukaran memori kontainer. `swappiness`Nilai `0` mencegah pertukaran terjadi kecuali diperlukan. `swappiness`Nilai `100` penyebab halaman sering ditukar. Nilai yang diterima adalah bilangan bulat antara `0` dan `100`. Jika Anda tidak menentukan nilai, nilai default `60` digunakan. Selain itu, jika Anda tidak menentukan nilai untuk`maxSwap`, maka parameter ini diabaikan. Parameter ini memetakan ke `--memory-swappiness` opsi untuk docker run.

> **Note**
>
> Jika Anda menggunakan tugas di Amazon Linux 2023, `swappiness` parameter tidak didukung.

`tmpfs`

Jalur kontainer, opsi pemasangan, dan ukuran maksimum (dalam MiB) dudukan. tmpfs Parameter ini memetakan ke `--tmpfs` opsi untuk docker run.

Tipe: Array objek Tmpfs

Wajib: Tidak

`containerPath`

Jalur file absolut tempat tmpfs volume akan dipasang.

Tipe: String

Diperlukan: Ya

`mountOptions`

Daftar opsi pemasangan volume tmpfs.

Tipe: Array string

Wajib: Tidak

Nilai yang Valid: `"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime"`

```
    | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" |
    "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave"
    | "relatime" | "norelatime" | "strictatime" | "nostrictatime" |
    "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"
```

`size`

Ukuran maksimum (dalam MiB) volume. tmpfs

Jenis: Integer

Wajib: Ya

Dependensi kontainer

`dependsOn`

Tipe: Array objek ContainerDependency

Wajib: Tidak

Dependensi yang ditentukan untuk pemulaian dan penonaktifan kontainer. Kontainer dapat berisi beberapa dependensi. Ketika dependensi ditentukan untuk memulai kontainer, maka akan dibalik jika ingin menonaktifkan kontainer. Misalnya, lihat Dependensi kontainer.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Instance memerlukan setidaknya versi `1.26.0` agen kontainer untuk mengaktifkan dependensi kontainer. Namun, kami menyarankan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS. Jika Anda menggunakan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memerlukan setidaknya `1.26.0-1` versi paket. `ecs-init` Jika instance penampung Anda diluncurkan dari versi `20190301` atau yang lebih baru, instance tersebut berisi versi yang diperlukan dari agen penampung dan`ecs-init`. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

```
"dependsOn": [
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

Tipe: String

Wajib: Ya

Nama kontainer yang harus memenuhi syarat yang ditentukan.

`condition`

Tipe: String

Wajib: Ya

Syarat dependensi kontainer. Berikut ini adalah syarat yang tersedia dan perilakunya:

- `START`Kondisi ini meniru perilaku link dan volume hari ini. Kondisi memvalidasi bahwa kontainer dependen dimulai sebelum mengizinkan kontainer lain untuk memulai.
- `COMPLETE`— Kondisi ini memvalidasi bahwa kontainer dependen berjalan hingga selesai (keluar) sebelum mengizinkan kontainer lain untuk memulai. Ini dapat berguna untuk wadah non-esensial yang menjalankan skrip dan kemudian keluar. Kondisi ini tidak dapat diatur pada wadah penting.
- `SUCCESS`— Kondisi ini sama dengan`COMPLETE`, tetapi juga mengharuskan wadah keluar dengan `zero` status. Kondisi ini tidak dapat diatur pada wadah penting.
- `HEALTHY`- Kondisi ini memvalidasi bahwa wadah dependen melewati pemeriksaan kesehatan kontainer sebelum mengizinkan wadah lain untuk memulai. Ini mengharuskan wadah dependen memiliki pemeriksaan kesehatan yang dikonfigurasi dalam definisi tugas. Syarat ini dikonfirmasi hanya pada memulai tugas.

Waktu habis kontainer

`startTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum menyerah pada penyelesaian dependensi untuk kontainer.

Misalnya, Anda menentukan dua kontainer dalam ketentuan tugas dengan `containerA` yang memiliki dependensi untuk `containerB` yang mencapai status `COMPLETE`, `SUCCESS`, atau `HEALTHY`. Jika `startTimeout` nilai ditentukan untuk `containerB` dan tidak mencapai status yang diinginkan dalam waktu itu, maka `containerA` tidak dimulai.

> **Note**
>
> Jika kontainer tidak memenuhi batasan ketergantungan atau waktu habis sebelum memenuhi batasan, Amazon ECS tidak memajukan kontainer dependen ke status berikutnya.

Nilai maksimumnya adalah 120 detik.

`stopTimeout`

Tipe: Integer

Wajib: Tidak

Nilai contoh: `120`

Durasi waktu (dalam detik) untuk menunggu sebelum kontainer dimatikan secara paksa jika tidak keluar dengan sendirinya secara normal.

Jika `stopTimeout` parameter tidak ditentukan, nilai yang ditetapkan untuk variabel konfigurasi agen penampung Amazon ECS akan `ECS_CONTAINER_STOP_TIMEOUT` digunakan. Jika `stopTimeout` parameter atau variabel konfigurasi `ECS_CONTAINER_STOP_TIMEOUT` agen tidak disetel, nilai default 30 detik untuk Linux kontainer dan 30 detik pada Windows kontainer digunakan. instans kontainer memerlukan setidaknya agen kontainer versi 1.26.0 untuk mengaktifkan nilai batas waktu penghentian kontainer. Akan tetapi, kami merekomendasikan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang cara memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihatMemperbarui agen kontainer Amazon ECS. Jika Anda menggunakan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon,

instans Anda memerlukan setidaknya versi 1.26.0-1 paket. `ecs-init` Jika instance penampung Anda diluncurkan dari versi `20190301` atau yang lebih baru, instance tersebut berisi versi yang diperlukan dari agen penampung dan`ecs-init`. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

Kontrol sistem

`systemControls`

Tipe: Objek SystemControl

Wajib: Tidak

Daftar parameter kernel namespace untuk diatur dalam wadah. Parameter ini memetakan ke `Sysctls` dalam perintah docker create-container dan opsi untuk docker run. `--sysctl` Misalnya, Anda dapat mengonfigurasi `net.ipv4.tcp_keepalive_time` pengaturan untuk mempertahankan koneksi yang lebih lama.

Kami tidak menyarankan Anda menentukan `systemControls` parameter terkait jaringan untuk beberapa kontainer dalam satu tugas yang juga menggunakan mode `awsvpc` atau `host` jaringan. Melakukan hal ini memiliki kelemahan sebagai berikut:

- Untuk tugas yang menggunakan mode jaringan `awsvpc`, jika Anda mengatur `systemControls` untuk kontainer apa pun, maka akan berlaku untuk semua kontainer dalam tugas. Jika Anda menyetel berbeda `systemControls` untuk beberapa kontainer dalam satu tugas, penampung yang dimulai terakhir menentukan mana yang `systemControls` berlaku.
- Untuk tugas yang menggunakan mode `host` jaringan, namespace jaringan `systemControls` tidak didukung.

Jika Anda menyetel namespace sumber daya IPC untuk digunakan untuk kontainer dalam tugas, kondisi berikut berlaku untuk kontrol sistem Anda. Untuk informasi selengkapnya, lihat Mode IPC.

- Untuk tugas yang menggunakan mode `host` IPC, namespace `systemControls` IPC tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` nilai namespace IPC berlaku untuk semua kontainer dalam tugas.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"systemControls": [
    {
         "namespace":"string",
         "value":"string"
    }
]
```

`namespace`

Tipe: String

Wajib: Tidak

Parameter kernel namespace untuk menetapkan untuk. `value`

Nilai namespace IPC yang valid:`"kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"`, dan `Sysctls` itu dimulai dengan `"fs.mqueue.*"`

Nilai namespace jaringan yang valid: `Sysctls` yang dimulai dengan `"net.*"`

`value`

Tipe: String

Wajib: Tidak

Nilai untuk parameter kernel namespace yang ditentukan dalam. `namespace`

Interaktif

`interactive`

Tipe: Boolean

Wajib: Tidak

Ketika parameter ini`true`, Anda dapat menerapkan aplikasi kontainer yang memerlukan `stdin` atau `tty` dialokasikan. Parameter ini memetakan ke `OpenStdin` dalam perintah docker create-container dan opsi `--interactive` untuk docker run.

Nilai default-nya `false`.

Terminal semu

`pseudoTerminal`

Tipe: Boolean

Wajib: Tidak

Jika parameter ini `true`, TTY dialokasikan. Parameter ini memetakan ke `Tty` dalam perintah docker create-container dan opsi `--tty` untuk docker run.

Nilai default-nya `false`.

Nama akselerator Elastic Inference

Persyaratan sumber daya akselerator Elastic Inference untuk definisi tugas Anda.

> **Note**
>
> Amazon Elastic Inference (EI) tidak lagi tersedia untuk pelanggan.

Parameter berikut diizinkan dalam definisi tugas:

`deviceName`

Tipe: String

Diperlukan: Ya

Nama perangkat akselerator Elastic Inference. Itu juga `deviceName` harus direferensikan dalam definisi kontainer lihat`Elastic Inference accelerator`.

`deviceType`

Tipe: String

Diperlukan: Ya

Akselerator Elastic Inference untuk digunakan.

Kendala penempatan tugas

Saat mendaftarkan definisi tugas, Anda dapat memberikan batasan penempatan tugas yang menyesuaikan cara Amazon ECS menempatkan tugas.

Anda dapat menggunakan batasan untuk menempatkan tugas berdasarkan Availability Zone, tipe instans, atau atribut kustom. Untuk informasi selengkapnya, lihat Tentukan instance kontainer mana yang digunakan Amazon ECS untuk tugas.

Parameter berikut diizinkan dalam ketentuan kontainer:

`expression`

Tipe: String

Wajib: Tidak

Ekspresi bahasa kueri klaster yang akan diterapkan pada batasan. Untuk informasi selengkapnya, lihat Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS.

`type`

Tipe: String

Diperlukan: Ya

Tipe batasan. Gunakan `memberOf` untuk membatasi pemilihan ke grup kandidat yang valid.

Konfigurasi proxy

`proxyConfiguration`

Tipe: Objek ProxyConfiguration

Wajib: Tidak

Detail konfigurasi untuk proxy App Mesh.

Untuk tugas yang menggunakan tipe EC2 peluncuran, instance container memerlukan setidaknya versi 1.26.0 dari agen kontainer dan setidaknya versi 1.26.0-1 `ecs-init`

paket untuk mengaktifkan konfigurasi proxy. Jika instans kontainer Anda diluncurkan dari AMI yang dioptimalkan Amazon ECS versi `20190301` atau yang lebih baru, maka instans kontainer mengandung versi agen kontainer dan `ecs-init` yang diperlukan. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
           "value": "string"
        }
    ]
}
```

`type`

Tipe: String

Nilai nilai: `APPMESH`

Wajib: Tidak

Tipe proxy. Satu-satunya nilai yang didukung adalah `APPMESH`.

`containerName`

Tipe: String

Diperlukan: Ya

Nama wadah yang berfungsi sebagai proxy App Mesh.

`properties`

Tipe: Array objek KeyValuePair

Wajib: Tidak

Set parameter konfigurasi jaringan berfungsi untuk menyediakan plugin Container Network Interface (CNI), yang ditetapkan sebagai pasangan nilai kunci.

- `IgnoredUID`— (Wajib) ID pengguna (UID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredGID` ditentukan, bidang ini tidak boleh kosong.
- `IgnoredGID`— (Wajib) ID grup (GID) dari wadah proxy sebagaimana didefinisikan oleh `user` parameter dalam definisi kontainer. Ini digunakan untuk memastikan bahwa proksi mengabaikan lalu lintasnya sendiri. Jika `IgnoredUID` ditentukan, bidang ini tidak boleh kosong.
- `AppPorts`— (Wajib) Daftar port yang digunakan aplikasi. Lalu lintas jaringan ke port ini diteruskan ke `ProxyIngressPort` dan `ProxyEgressPort`.
- `ProxyIngressPort`— (Wajib) Menentukan port yang lalu lintas masuk ke `AppPorts` diarahkan.
- `ProxyEgressPort`— (Wajib) Menentukan port yang lalu lintas keluar dari `AppPorts` diarahkan ke.
- `EgressIgnoredPorts`— (Wajib) Lalu lintas keluar yang menuju ke port yang ditentukan ini diabaikan dan tidak dialihkan ke. `ProxyEgressPort` Ia bisa berupa daftar kosong.
- `EgressIgnoredIPs`— (Wajib) Lalu lintas keluar yang menuju ke alamat IP yang ditentukan ini diabaikan dan tidak dialihkan ke. `ProxyEgressPort` Ia bisa berupa daftar kosong.

`name`

Tipe: String

Wajib: Tidak

Nama pasangan nilai kunci.

`value`

Tipe: String

Wajib: Tidak

Nilai pasangan nilai kunci.

Volume

Saat Anda mendaftarkan definisi tugas, Anda dapat secara opsional menentukan daftar volume yang akan diteruskan ke Docker daemon pada instance container, yang kemudian tersedia untuk diakses oleh container lain pada instance container yang sama.

Berikut ini adalah jenis volume data yang dapat digunakan:

- Volume Amazon EBS - Menyediakan penyimpanan blok yang hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data. Anda dapat melampirkan 1 volume Amazon EBS per tugas Amazon ECS saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Volume Amazon EBS didukung untuk tugas-tugas Linux. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EBS dengan Amazon ECS.
- Volume Amazon EFS — Menyediakan penyimpanan file yang sederhana, terukur, dan persisten untuk digunakan dengan tugas Amazon ECS Anda. Kapasitas penyimpanan Amazon EFS bersifat elastis. Itu tumbuh dan menyusut secara otomatis saat Anda menambah dan menghapus file. Aplikasi Anda dapat memiliki penyimpanan yang mereka butuhkan dan ketika mereka membutuhkannya. Volume Amazon EFS didukung. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.
- FSx untuk volume Windows File Server - Menyediakan server file Microsoft Windows yang dikelola sepenuhnya. Server file ini didukung oleh sistem file Windows. Saat menggunakan FSx untuk Windows File Server bersama dengan Amazon ECS, Anda dapat menyediakan tugas Windows Anda dengan penyimpanan file persisten, terdistribusi, bersama, dan statis. Untuk informasi selengkapnya, lihat Gunakan FSx untuk volume Server File Windows dengan Amazon ECS.

  Wadah Windows di Fargate tidak mendukung opsi ini.
- Volume Docker — Volume yang dikelola Docker yang dibuat di bawah instans `/var/lib/docker/volumes` Amazon host. EC2 Driver volume Docker (juga disebut sebagai plugin) digunakan untuk mengintegrasikan volume dengan sistem penyimpanan eksternal, seperti Amazon EBS. Built-in driver volume `local` atau driver volume pihak ke tiga dapat digunakan. Volume Docker hanya didukung saat menjalankan tugas di EC2 instans Amazon. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan volume Docker, tentukan `dockerVolumeConfiguration` dalam ketentuan tugas Anda.
- Bind mount — File atau direktori pada mesin host yang dipasang ke dalam wadah. Volume host bind mount didukung. Untuk menggunakan volume host bind mount, tentukan `sourcePath` nilai `host` dan opsional dalam definisi tugas Anda.

Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.

Parameter berikut diizinkan dalam ketentuan kontainer.

`name`

Tipe: String

Wajib: Tidak

Nama volume. Hingga 255 huruf (huruf besar dan kecil), angka, tanda hubung (), dan garis bawah (-) diperbolehkan. _ Nama ini direferensikan dalam `sourceVolume` parameter `mountPoints` objek definisi kontainer.

`host`

Wajib: Tidak

`host`Parameter ini digunakan untuk mengikat siklus hidup bind mount ke EC2 instance Amazon host, bukan tugas, dan tempat penyimpanannya. Jika `host` parameternya kosong, maka daemon Docker menetapkan jalur host untuk volume data Anda, tetapi data tidak dijamin akan bertahan setelah wadah yang terkait dengannya berhenti berjalan.

Kontainer Windows dapat memasang seluruh direktori pada drive yang sama dengan `$env:ProgramData`.

> **Note**
>
> `sourcePath`Parameter hanya didukung saat menggunakan tugas yang di-host di EC2 instans Amazon.

`sourcePath`

Tipe: String

Wajib: Tidak

Saat `host` parameter digunakan, tentukan a `sourcePath` untuk mendeklarasikan jalur pada EC2 instance Amazon host yang disajikan ke wadah. Jika parameter ini kosong, daemon Docker akan menetapkan jalur host untuk Anda. Jika `host` parameter berisi lokasi `sourcePath` file, maka volume data tetap ada di lokasi yang ditentukan pada EC2 instance Amazon host hingga Anda menghapusnya secara manual. Jika `sourcePath` nilai tidak ada

pada EC2 instance Amazon host, daemon Docker membuatnya. Jika lokasinya memang ada, konten dari folder jalur sumber diekspor.

`configuredAtLaunch`

Tipe: Boolean

Wajib: Tidak

Menentukan apakah volume dikonfigurasi saat peluncuran. Saat disetel ke`true`, Anda dapat mengonfigurasi volume saat menjalankan tugas mandiri, atau saat membuat atau memperbarui layanan. Saat disetel ke`true`, Anda tidak akan dapat memberikan konfigurasi volume lain dalam definisi tugas. Parameter ini harus diatur `true` untuk mengonfigurasi volume Amazon EBS untuk lampiran ke tugas. Menyetel `configuredAtLaunch` ke `true` dan menunda konfigurasi volume ke fase peluncuran memungkinkan Anda membuat definisi tugas yang tidak dibatasi untuk jenis volume atau pengaturan volume tertentu. Melakukan hal ini membuat definisi tugas Anda dapat digunakan kembali di seluruh lingkungan eksekusi yang berbeda. Untuk informasi selengkapnya, lihat volume Amazon EBS.

`dockerVolumeConfiguration`

Jenis: DockerVolumeConfigurationObjek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Docker. Volume Docker hanya didukung saat menjalankan tugas pada EC2 instance. Wadah Windows hanya mendukung penggunaan `local` driver. Untuk menggunakan pemasangan mengikat, tentukan `host` saja.

`scope`

Tipe: String

Nilai Valid: `task` | `shared`

Wajib: Tidak

Cakupan untuk volume Docker, yang menentukan siklus hidupnya. Volume Docker yang tercakup ke `task` secara otomatis disediakan saat tugas dimulai dan dihancurkan saat tugas berhenti. Volume Docker yang tercakup sebagai `shared` dipertahankan setelah tugas berhenti.

`autoprovision`

Jenis: Boolean

Nilai default: `false`

Wajib: Tidak

Jika nilai ini`true`, volume Docker dibuat jika belum ada. Bidang ini hanya digunakan jika `scope` ada`shared`. Jika `scope` ya`task`, maka parameter ini harus dihilangkan.

`driver`

Tipe: String

Wajib: Tidak

Driver volume Docker yang digunakan. Nilai driver harus sesuai dengan nama driver yang disediakan oleh Docker karena nama ini digunakan untuk penempatan tugas. Jika driver diinstal dengan menggunakan CLI plugin Docker, `docker plugin ls` gunakan untuk mengambil nama driver dari instance container Anda. Jika driver diinstal dengan menggunakan metode lain, gunakan penemuan plugin Docker untuk mengambil nama driver.

`driverOpts`

Tipe: String

Wajib: Tidak

Peta opsi khusus driver Docker untuk dilewati. Parameter ini dipetakan ke `DriverOpts` bagian Create a volume dari Docker.

`labels`

Tipe: String

Wajib: Tidak

Metadata kustom untuk ditambahkan ke volume Docker.

`efsVolumeConfiguration`

Jenis: EFSVolumeKonfigurasi Objek

Wajib: Tidak

Parameter ini ditentukan saat menggunakan volume Amazon EFS.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Amazon EFS yang akan digunakan.

`rootDirectory`

Tipe: String

Wajib: Tidak

Direktori dalam sistem file Amazon EFS untuk dipasang sebagai direktori root di dalam host. Jika parameter ini dihilangkan, root volume Amazon EFS akan digunakan. Menentukan `/` memiliki efek yang sama seperti mengabaikan parameter ini.

> **Important**
>
> Jika titik akses EFS ditentukan dalam`authorizationConfig`, parameter direktori root harus dihilangkan atau disetel ke`/`, yang akan menerapkan jalur yang disetel pada titik akses EFS.

`transitEncryption`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan mengaktifkan enkripsi untuk data Amazon EFS dalam perjalanan antara host Amazon ECS dan server Amazon EFS. Jika otorisasi Amazon EFS IAM digunakan, enkripsi transit harus diaktifkan. Jika parameter ini diabaikan, nilai default `DISABLED` akan digunakan. Untuk informasi lebih lanjut, lihat Pengenkripsian Data Saat Transit di Panduan Pengguna Amazon Elastic File System.

`transitEncryptionPort`

Tipe: Integer

Wajib: Tidak

Port yang akan digunakan saat mengirim data terenkripsi antara host Amazon ECS dan server Amazon EFS. Jika Anda tidak menentukan port enkripsi transit, tugas akan menggunakan

strategi pemilihan port yang digunakan oleh helper mount Amazon EFS. Untuk informasi lebih lanjut, lihat Pembantu Pemasangan EFS di Panduan Pengguna Amazon Elastic File System.

`authorizationConfig`

Jenis: EFSAuthorizationConfig Object

Wajib: Tidak

Detail konfigurasi otorisasi untuk sistem file Amazon EFS.

`accessPointId`

Tipe: String

Wajib: Tidak

ID titik akses yang akan digunakan. Jika titik akses ditentukan, nilai direktori root di dalam `efsVolumeConfiguration` harus dihilangkan atau disetel ke`/`, yang akan memberlakukan jalur yang ditetapkan pada titik akses EFS. Jika titik akses digunakan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Untuk informasi lebih lanjut, lihat Bekerja dengan Titik Akses Amazon EFS dalam Panduan Pengguna Amazon Elastic File System.

`iam`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Menentukan apakah akan menggunakan peran IAM tugas Amazon ECS yang ditentukan dalam definisi tugas saat memasang sistem file Amazon EFS. Jika diaktifkan, enkripsi transit harus diaktifkan di `EFSVolumeConfiguration`. Jika Anda menghilangkan parameter ini, nilai default `DISABLED` akan digunakan. Untuk informasi selengkapnya, lihat IAM Role pada Tugas.

`FSxWindowsFileServerVolumeConfiguration`

Jenis: FSxWindowsFileServerVolumeConfigurationObjek

Wajib: Ya

Parameter ini ditentukan saat Anda menggunakan sistem file Amazon FSx untuk Windows File Server untuk penyimpanan tugas.

`fileSystemId`

Tipe: String

Diperlukan: Ya

ID sistem file Windows File Server untuk digunakan. FSx

`rootDirectory`

Tipe: String

Diperlukan: Ya

Direktori dalam sistem file FSx untuk Windows File Server untuk dipasang sebagai direktori root di dalam host.

`authorizationConfig`

`credentialsParameter`

Tipe: String

Diperlukan: Ya

Opsi kredensi otorisasi.

pilihan:

- Nama Sumber Daya Amazon (ARN) dari sebuah AWS Secrets Managerrahasia.
- ARN dari suatu AWS Systems Managerparameter.

`domain`

Tipe: String

Diperlukan: Ya

Nama domain yang sepenuhnya memenuhi syarat yang dihosting oleh direktori AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) atau Direktori EC2 Aktif yang dihosting sendiri.

Tanda

Ketika mendaftarkan ketentuan tugas, Anda secara opsional dapat menentukan tanda metadata yang diterapkan ke ketentuan tugas. Tanda membantu mengategorikan dan mengatur ketentuan

tugas Anda. Setiap tanda terdiri dari kunci dan nilai opsional. Anda mendefinisikan keduanya. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.

> **Important**
>
> Jangan menambahkan informasi identitas pribadi atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak AWS layanan, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan untuk data pribadi atau sensitif.

Parameter berikut diperbolehkan dalam objek tanda.

`key`

Tipe: String

Wajib: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Wajib: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

Parameter ketentuan tugas lainnya

Parameter definisi tugas berikut dapat digunakan saat mendaftarkan definisi tugas di konsol Amazon ECS dengan menggunakan opsi Configure via JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Topik

- Mode IPC
- Mode PID

- Injeksi kesalahan

Mode IPC

`ipcMode`

Tipe: String

Wajib: Tidak

Namespace sumber daya IPC yang digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host`, `task`, atau `none`. Jika `host` ditentukan, maka semua kontainer yang berada dalam tugas yang menentukan mode `host` IPC pada instance kontainer yang sama berbagi sumber daya IPC yang sama dengan instans Amazon EC2 host. Jika `task` ditentukan, semua kontainer yang berada dalam tugas yang ditentukan berbagi sumber daya IPC yang sama. Jika `none` ditentukan, maka sumber daya IPC dalam kontainer tugas bersifat privat dan tidak dibagi dengan kontainer lain dalam tugas atau pada instans kontainer. Jika tidak ada nilai yang ditentukan, maka berbagi namespace sumber daya IPC tergantung pada pengaturan daemon Docker pada instans kontainer.

Jika mode `host` IPC digunakan, ada risiko tinggi eksposur namespace IPC yang tidak diinginkan.

Jika Anda menyetel parameter kernel namespace yang digunakan `systemControls` untuk kontainer dalam tugas, berikut ini berlaku untuk namespace sumber daya IPC Anda.

- Untuk tugas yang menggunakan mode `host` IPC, namespace IPC yang terkait `systemControls` tidak didukung.
- Untuk tugas yang menggunakan mode `task` IPC, `systemControls` yang berhubungan dengan namespace IPC berlaku untuk semua kontainer dalam tugas.

Mode PID

`pidMode`

Tipe: String

Nilai Valid: `host` | `task`

Wajib: Tidak

Namspace proses yang akan digunakan untuk kontainer dalam tugas. Nilai yang valid adalah `host` atau `task`. Misalnya, pemantauan sidecar mungkin `pidMode` perlu mengakses informasi tentang kontainer lain yang berjalan dalam tugas yang sama.

Jika `host` ditentukan, semua kontainer dalam tugas yang menentukan mode `host` PID pada instance container yang sama berbagi namespace proses yang sama dengan instance Amazon EC2 host.

Jika `task` ditentukan, semua kontainer dalam tugas tertentu berbagi namespace proses yang sama.

Jika tidak ada nilai yang ditentukan, defaultnya adalah namespace pribadi untuk setiap kontainer.

Jika mode `host` PID digunakan, ada risiko tinggi eksposur namespace proses yang tidak diinginkan.

> **Note**
>
> Parameter ini tidak didukung untuk kontainer Windows.

Injeksi kesalahan

`enableFaultInjection`

Jenis: Boolean

Nilai Valid: `true` | `false`

Wajib: Tidak

Jika parameter ini disetel ke`true`, dalam muatan tugas, Amazon ECS menerima permintaan injeksi kesalahan dari wadah tugas. Secara default, parameter ini diatur ke`false`.

Templat definisi tugas Amazon ECS

Templat ketentuan tugas kosong ditampilkan sebagai berikut. Anda dapat menggunakan template ini untuk membuat definisi tugas Anda, yang kemudian dapat ditempelkan ke area input JSON konsol atau disimpan ke file dan digunakan dengan opsi. AWS CLI `--cli-input-json` Untuk informasi selengkapnya, lihat Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate.

Templat jenis EC2 peluncuran Amazon

```
{
  "family": "",
  "taskRoleArn": "",
  "executionRoleArn": "",
  "networkMode": "none",
  "containerDefinitions": [
    {
      "name": "",
      "image": "",
      "repositoryCredentials": {
        "credentialsParameter": ""
      },
      "cpu": 0,
      "memory": 0,
      "memoryReservation": 0,
      "links": [""],
      "portMappings": [
        {
          "containerPort": 0,
          "hostPort": 0,
          "protocol": "tcp"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      },
      "essential": true,
      "entryPoint": [""],
      "command": [""],
      "environment": [
        {
          "name": "",
          "value": ""
        }
      ],
      "environmentFiles": [
        {
          "value": "",
          "type": "s3"
        }
```

```
      ],
      "mountPoints": [
        {
          "sourceVolume": "",
          "containerPath": "",
          "readOnly": true
        }
      ],
      "volumesFrom": [
        {
          "sourceContainer": "",
          "readOnly": true
        }
      ],
      "linuxParameters": {
        "capabilities": {
          "add": [""],
          "drop": [""]
        },
        "devices": [
          {
            "hostPath": "",
            "containerPath": "",
            "permissions": ["read"]
          }
        ],
        "initProcessEnabled": true,
        "sharedMemorySize": 0,
        "tmpfs": [
          {
            "containerPath": "",
            "size": 0,
            "mountOptions": [""]
          }
        ],
        "maxSwap": 0,
        "swappiness": 0
      },
      "secrets": [
        {
          "name": "",
          "valueFrom": ""
        }
      ],
```

```
      "dependsOn": [
        {
          "containerName": "",
          "condition": "COMPLETE"
        }
      ],
      "startTimeout": 0,
      "stopTimeout": 0,
      "hostname": "",
      "user": "",
      "workingDirectory": "",
      "disableNetworking": true,
      "privileged": true,
      "readonlyRootFilesystem": true,
      "dnsServers": [""],
      "dnsSearchDomains": [""],
      "extraHosts": [
        {
          "hostname": "",
          "ipAddress": ""
        }
      ],
      "dockerSecurityOptions": [""],
      "interactive": true,
      "pseudoTerminal": true,
      "dockerLabels": {
        "KeyName": ""
      },
      "ulimits": [
        {
          "name": "nofile",
          "softLimit": 0,
          "hardLimit": 0
        }
      ],
      "logConfiguration": {
        "logDriver": "splunk",
        "options": {
          "KeyName": ""
        },
        "secretOptions": [
          {
            "name": "",
            "valueFrom": ""
```

```
          }
        ]
      },
      "healthCheck": {
        "command": [""],
        "interval": 0,
        "timeout": 0,
        "retries": 0,
        "startPeriod": 0
      },
      "systemControls": [
        {
          "namespace": "",
          "value": ""
        }
      ],
      "resourceRequirements": [
        {
          "value": "",
          "type": "InferenceAccelerator"
        }
      ],
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "KeyName": ""
        }
      }
    }
  ],
  "volumes": [
    {
      "name": "",
      "host": {
        "sourcePath": ""
      },
      "configuredAtLaunch": true,
      "dockerVolumeConfiguration": {
        "scope": "shared",
        "autoprovision": true,
        "driver": "",
        "driverOpts": {
          "KeyName": ""
        },
```

```
        "labels": {
          "KeyName": ""
        }
      },
      "efsVolumeConfiguration": {
        "fileSystemId": "",
        "rootDirectory": "",
        "transitEncryption": "DISABLED",
        "transitEncryptionPort": 0,
        "authorizationConfig": {
          "accessPointId": "",
          "iam": "ENABLED"
        }
      },
      "fsxWindowsFileServerVolumeConfiguration": {
        "fileSystemId": "",
        "rootDirectory": "",
        "authorizationConfig": {
          "credentialsParameter": "",
          "domain": ""
        }
      }
    }
  ],
  "placementConstraints": [
    {
      "type": "memberOf",
      "expression": ""
    }
  ],
  "requiresCompatibilities": ["EC2"],
  "cpu": "",
  "memory": "",
  "tags": [
    {
      "key": "",
      "value": ""
    }
  ],
  "pidMode": "task",
  "ipcMode": "task",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "",
```

```
    "properties": [
      {
        "name": "",
        "value": ""
      }
    ]
  },
  "inferenceAccelerators": [
    {
      "deviceName": "",
      "deviceType": ""
    }
  ],
  "ephemeralStorage": {
    "sizeInGiB": 0
  },
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "WINDOWS_SERVER_20H2_CORE"
  }
}
```

Templat jenis peluncuran Fargate

Important

Untuk tipe peluncuran Fargate, Anda harus menyertakan `operatingSystemFamily` parameter dengan salah satu nilai berikut:

- `LINUX`
- `WINDOWS_SERVER_2019_FULL`
- `WINDOWS_SERVER_2019_CORE`
- `WINDOWS_SERVER_2022_FULL`
- `WINDOWS_SERVER_2022_CORE`

```
{
    "family": "",
    "runtimePlatform": {"operatingSystemFamily": ""},
```

```
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "awsvpc",
    "platformFamily": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {"credentialsParameter": ""},
            "cpu": 0,
            "memory": 0,
            "memoryReservation": 0,
            "links": [""],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [""],
            "command": [""],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
```

```
                "sourceContainer": "",
                "readOnly": true
            }
        ],
        "linuxParameters": {
            "capabilities": {
                "add": [""],
                "drop": [""]
            },
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": ["read"]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [""]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "HEALTHY"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
        "user": "",
```

```
            "workingDirectory": "",
            "disableNetworking": true,
            "privileged": true,
            "readonlyRootFilesystem": true,
            "dnsServers": [""],
            "dnsSearchDomains": [""],
            "extraHosts": [
                {
                    "hostname": "",
                    "ipAddress": ""
                }
            ],
            "dockerSecurityOptions": [""],
            "interactive": true,
            "pseudoTerminal": true,
            "dockerLabels": {"KeyName": ""},
            "ulimits": [
                {
                    "name": "msgqueue",
                    "softLimit": 0,
                    "hardLimit": 0
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {"KeyName": ""},
                "secretOptions": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ]
            },
            "healthCheck": {
                "command": [""],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
                    "value": ""
```

```
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "GPU"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentd",
                "options": {"KeyName": ""}
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {"sourcePath": ""},
            "configuredAtLaunch":true,
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {"KeyName": ""},
                "labels": {"KeyName": ""}
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            }
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
```

```
            "value": ""
        }
    ],
    "ephemeralStorage": {"sizeInGiB": 0},
    "pidMode": "task",
    "ipcMode": "none",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ]
}
```

Anda dapat membuat template definisi tugas ini menggunakan AWS CLI perintah berikut.

```
aws ecs register-task-definition --generate-cli-skeleton
```

Contoh definisi tugas Amazon ECS

Anda dapat menyalin contoh dan cuplikan untuk mulai membuat definisi tugas Anda sendiri.

Anda dapat menyalin contoh, lalu menempelkannya saat Anda menggunakan opsi Konfigurasi melalui JSON di konsol. Pastikan untuk menyesuaikan contoh, seperti menggunakan ID akun Anda. Anda dapat menyertakan cuplikan dalam definisi tugas JSON Anda. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol dan Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate.

Untuk contoh definisi tugas lainnya, lihat AWS Contoh Definisi Tugas pada GitHub.

Topik

- Peladen Web
- splunkdriver log
- fluentddriver log
- gelfdriver log
- Beban kerja pada instance eksternal
- Peran IAM definisi gambar dan tugas Amazon ECR
- Entrypoint dengan perintah
- Dependensi kontainer
- Volume dalam definisi tugas
- Ketentuan tugas sampel Windows

Peladen Web

Berikut ini adalah contoh definisi tugas menggunakan wadah Linux pada jenis peluncuran Fargate yang mengatur server web:

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "public.ecr.aws/docker/library/httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
```

```
                "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

Berikut ini adalah contoh definisi tugas menggunakan wadah Windows pada jenis peluncuran Fargate yang mengatur server web:

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
```

```
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

`splunk`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `splunk` log dalam definisi tugas yang mengirimkan log ke layanan jarak jauh. Parameter token Splunk ditetapkan sebagai opsi rahasia karena dapat diperlakukan sebagai data sensitif. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

```
"containerDefinitions": [{
  "logConfiguration": {
   "logDriver": "splunk",
   "options": {
    "splunk-url": "https://cloud.splunk.com:8080",
    "tag": "tag_name",
   },
   "secretOptions": [{
    "name": "splunk-token",
    "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:splunk-token-
KnrBkD"
}],
```

`fluentd`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `fluentd` log dalam definisi tugas yang mengirimkan log ke layanan jarak jauh. Nilai `fluentd-address` ditetapkan sebagai opsi rahasia karena dapat diperlakukan sebagai data sensitif. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "fluentd",
  "options": {
   "tag": "fluentd demo"
  },
  "secretOptions": [{
   "name": "fluentd-address",
   "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:fluentd-address-
KnrBkD"
  }]
 },
 "entryPoint": [],
 "portMappings": [{
             "hostPort": 80,
             "protocol": "tcp",
             "containerPort": 80
             },
             {
  "hostPort": 24224,
  "protocol": "tcp",
  "containerPort": 24224
 }]
}],
```

`gelf`driver log

Cuplikan berikut menunjukkan cara menggunakan driver `gelf` log dalam definisi tugas yang mengirimkan log ke host jarak jauh yang menjalankan Logstash yang mengambil log Gelf sebagai input. Untuk informasi selengkapnya, lihat `logConfiguration`.

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "gelf",
```

```
  "options": {
   "gelf-address": "udp://logstash-service-address:5000",
   "tag": "gelf task demo"
  }
 },
 "entryPoint": [],
 "portMappings": [{
   "hostPort": 5000,
   "protocol": "udp",
   "containerPort": 5000
  },
  {
   "hostPort": 5000,
   "protocol": "tcp",
   "containerPort": 5000
  }
 ]
}],
```

Beban kerja pada instance eksternal

Saat mendaftarkan definisi tugas Amazon ECS, gunakan `requiresCompatibilities` parameter dan tentukan `EXTERNAL` yang memvalidasi bahwa definisi tugas kompatibel untuk digunakan saat menjalankan beban kerja Amazon ECS pada instans eksternal Anda. Jika Anda menggunakan konsol untuk mendaftarkan definisi tugas, Anda harus menggunakan editor JSON. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

> **Important**
>
> Jika tugas Anda memerlukan peran IAM eksekusi tugas, pastikan itu ditentukan dalam definisi tugas.

Ketika Anda men-deploy beban kerja, gunakan tipe peluncuran `EXTERNAL` ketika membuat layanan atau menjalankan tugas mandiri Anda.

Berikut ini merupakan ketentuan tugas contoh.

Linux

```
{
```

```
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "nginx",
  "image": "public.ecr.aws/nginx/nginx:latest",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "nginx"
}
```

Windows

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "windows-container",
  "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019",
  "memory": 256,
  "cpu": 512,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "windows-container"
}
```

Peran IAM definisi gambar dan tugas Amazon ECR

Cuplikan berikut menggunakan gambar Amazon ECR yang disebut `aws-nodejs-sample` dengan `v1` tag dari registri. `123456789012.dkr.ecr.us-west-2.amazonaws.com` Wadah dalam tugas ini mewarisi izin IAM dari peran. `arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole` Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

```
{
    "containerDefinitions": [
        {
            "name": "sample-app",
            "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/aws-nodejs-
sample:v1",
            "memory": 200,
            "cpu": 10,
            "essential": true
        }
    ],
    "family": "example_task_3",
    "taskRoleArn": "arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole"
}
```

Entrypoint dengan perintah

Cuplikan berikut menunjukkan sintaks untuk wadah Docker yang menggunakan titik masuk dan argumen perintah. Kontainer ini menge-ping `example.com` empat kali dan kemudian keluar.

```
{
    "containerDefinitions": [
        {
            "memory": 32,
            "essential": true,
            "entryPoint": ["ping"],
            "name": "alpine_ping",
            "readonlyRootFilesystem": true,
            "image": "alpine:3.4",
            "command": [
                "-c",
                "4",
                "example.com"
            ],
```

```
            "cpu": 16
        }
    ],
    "family": "example_task_2"
}
```

Dependensi kontainer

Cuplikan ini menunjukkan sintaks untuk definisi tugas dengan beberapa kontainer di mana dependensi kontainer ditentukan. Dalam definisi tugas berikut, `envoy` wadah harus mencapai status sehat, ditentukan oleh parameter pemeriksaan kesehatan wadah yang diperlukan, sebelum `app` wadah dimulai. Untuk informasi selengkapnya, lihat Dependensi kontainer.

```
{
  "family": "appmesh-gateway",
  "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
  },
  "proxyConfiguration":{
      "type": "APPMESH",
      "containerName": "envoy",
      "properties": [
          {
              "name": "IgnoredUID",
              "value": "1337"
          },
          {
              "name": "ProxyIngressPort",
              "value": "15000"
          },
          {
              "name": "ProxyEgressPort",
              "value": "15001"
          },
          {
              "name": "AppPorts",
              "value": "9080"
          },
          {
              "name": "EgressIgnoredIPs",
              "value": "169.254.170.2,169.254.169.254"
          }
      ]
```

```
  },
  "containerDefinitions": [
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
          "name": "ENVOY_LOG_LEVEL",
          "value": "info"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "echo hello"
        ],
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
```

```
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Volume dalam definisi tugas

Gunakan yang berikut ini untuk memahami cara menentukan volume dalam tugas.

- Untuk informasi tentang cara mengonfigurasi volume Amazon EBS, lihatTentukan konfigurasi volume Amazon EBS pada penerapan Amazon ECS.
- Untuk informasi tentang cara mengonfigurasi volume Amazon EFS, lihatMengonfigurasi sistem file Amazon EFS untuk Amazon ECS menggunakan konsol.
- Untuk informasi tentang cara mengkonfigurasi volume FSx untuk Windows File Server, lihatPelajari cara mengonfigurasi FSx sistem file Windows File Server untuk Amazon ECS.
- Untuk informasi tentang cara mengonfigurasi volume Docker, lihatContoh volume Docker untuk Amazon ECS.
- Untuk informasi tentang cara mengonfigurasi bind mount, lihatContoh Bind mount untuk Amazon ECS.

Ketentuan tugas sampel Windows

Berikut ini adalah contoh definisi tugas untuk membantu Anda memulai dengan wadah Windows di Amazon ECS.

Example Aplikasi Contoh Konsol Amazon ECS untuk Windows

Definisi tugas berikut adalah aplikasi sampel konsol Amazon ECS yang diproduksi di wizard yang dijalankan pertama untuk Amazon ECS; itu telah di-porting untuk menggunakan gambar wadah Windows. `microsoft/iis`

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
```

```
      "cpu": 1024,
      "entryPoint":["powershell", "-Command"],
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file -
Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "memory": 1024,
      "essential": true
    }
  ],
  "networkMode": "awsvpc",
  "memory": "1024",
  "cpu": "1024"
}
```

Klaster-klaster Amazon ECS

Cluster Amazon ECS adalah pengelompokan tugas atau layanan yang logis. Selain tugas dan layanan, cluster terdiri dari sumber daya berikut:

- Kapasitas infrastruktur yang dapat menjadi kombinasi dari berikut ini:
  - EC2 Contoh Amazon di cloud AWS
  - Tanpa server (AWS Fargate) di awan AWS
  - Mesin virtual lokal (VM) atau server
- Jaringan (VPC dan subnet) tempat tugas dan layanan Anda berjalan

  Saat Anda menggunakan EC2 instans Amazon untuk kapasitas, subnet dapat berada di Availability Zones, Local Zones, Wavelength Zones, atau. AWS Outposts
- Namespace opsional

  Namespace digunakan untuk service-to-service komunikasi dengan Service Connect.
- Opsi pemantauan

  CloudWatch Wawasan Kontainer datang dengan biaya tambahan dan merupakan layanan yang dikelola sepenuhnya. Secara otomatis mengumpulkan, mengumpulkan, dan merangkum metrik dan log Amazon ECS.

Berikut ini adalah konsep umum tentang cluster Amazon ECS.

- Anda membuat cluster untuk memisahkan sumber daya Anda.
- Cluster Wilayah AWS spesifik.
- Cluster dapat berada di salah satu negara bagian berikut.

  AKTIF

  Cluster siap menerima tugas dan, jika berlaku, Anda dapat mendaftarkan instance kontainer dengan cluster.

  PENYEDIAAN

  Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas sedang dibuat.

PEMBATALAN PENYEDIAAN

Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas sedang dihapus.

FAILED

Cluster memiliki penyedia kapasitas yang terkait dengannya dan sumber daya yang dibutuhkan untuk penyedia kapasitas gagal dibuat.

TIDAK AKTIF

Cluster telah dihapus. Klaster dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Perilaku ini dapat berubah di masa depan, jadi pastikan Anda tidak bergantung pada `INACTIVE` cluster yang bertahan.

- Anda dapat menggunakan jenis instans yang berbeda untuk jenis EC2 peluncuran atau penyedia kapasitas grup Auto Scaling. Sebuah instance hanya dapat didaftarkan ke satu cluster pada satu waktu.
- Anda dapat membatasi akses ke cluster dengan membuat kebijakan IAM kustom. Untuk informasi, lihat Contoh cluster Amazon ECS bagian diContoh kebijakan berbasis identitas untuk Amazon Elastic Container Service.
- Anda dapat menggunakan Service Auto Scaling untuk menskalakan tugas Fargate. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.
- Anda dapat mengonfigurasi namespace Service Connect default untuk sebuah klaster. Setelah Anda menetapkan namespace Service Connect default, layanan baru apa pun yang dibuat di klaster dapat ditambahkan sebagai layanan klien di namespace dengan mengaktifkan Service Connect. Tidak diperlukan konfigurasi tambahan. Untuk informasi selengkapnya, lihat Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek.

Penyedia kapasitas

Penyedia kapasitas Amazon ECS mengelola penskalaan infrastruktur untuk tugas di cluster Anda. Setiap cluster dapat memiliki satu atau lebih penyedia kapasitas dan strategi penyedia kapasitas opsional. Anda dapat menetapkan strategi penyedia kapasitas default ke cluster. Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan strategi penyedia kapasitas default klaster atau strategi penyedia kapasitas yang mengganti tugas default. Strategi penyedia kapasitas default cluster hanya berlaku jika Anda tidak menentukan jenis peluncuran, atau

strategi penyedia kapasitas untuk tugas atau layanan Anda. Jika Anda memberikan salah satu dari parameter ini, strategi default tidak digunakan.

Untuk Fargate, Anda tidak perlu membuat atau mengelola kapasitas. Anda hanya perlu mengaitkan salah satu penyedia kapasitas yang telah ditentukan sebelumnya berikut dengan cluster:

- Fargate
- Spot Fargate

Bila Anda menggunakan EC2 instance untuk kapasitas Anda, Anda menggunakan grup Auto Scaling untuk mengelola EC2 instans. Auto Scaling membantu memastikan bahwa Anda memiliki jumlah EC2 instans yang benar yang tersedia untuk menangani pemuatan aplikasi.

Cluster dapat berisi campuran tugas yang di-host di AWS Fargate, EC2 instans Amazon, atau instans eksternal. Tugas dapat berjalan di Fargate atau EC2 infrastruktur sebagai tipe peluncuran atau strategi penyedia kapasitas. Jika Anda menggunakan EC2 sebagai tipe peluncuran, Amazon ECS tidak melacak dan menskalakan kapasitas grup EC2 Auto Scaling Amazon. Untuk informasi selengkapnya tentang tipe peluncuran, lihat Jenis peluncuran Amazon ECS.

Sebuah cluster dapat berisi campuran penyedia kapasitas grup Auto Scaling dan penyedia kapasitas Fargate. Strategi penyedia kapasitas hanya dapat berisi penyedia kapasitas grup Auto Scaling atau penyedia kapasitas Fargate.

Cluster Amazon ECS untuk Fargate

Penyedia kapasitas Amazon ECS mengelola penskalaan infrastruktur untuk tugas di cluster Anda. Setiap cluster dapat memiliki satu atau lebih penyedia kapasitas dan strategi penyedia kapasitas opsional. Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan strategi penyedia kapasitas default kluster atau strategi penyedia kapasitas yang mengganti tugas default.

Ketika Anda menjalankan tugas Anda AWS Fargate, Anda tidak perlu membuat atau mengelola kapasitas. Anda hanya perlu mengaitkan salah satu penyedia kapasitas yang telah ditentukan sebelumnya berikut dengan cluster:

- Fargate
- Spot Fargate

Dengan Amazon ECS pada penyedia AWS Fargate kapasitas, Anda dapat menggunakan kapasitas Fargate dan Fargate Spot dengan tugas Amazon ECS Anda.

Dengan Fargate Spot, Anda dapat menjalankan tugas Amazon ECS toleran interupsi dengan tarif yang didiskon dibandingkan dengan harga Fargate. Spot Fargate menjalankan tugas pada kapasitas komputasi cadangan. Ketika AWS membutuhkan kapasitas kembali, tugas Anda terganggu dengan peringatan dua menit.

Saat tugas yang menggunakan penyedia kapasitas Fargate dan Fargate Spot dihentikan, peristiwa perubahan status tugas dikirim ke Amazon. EventBridge Alasan berhenti menggambarkan penyebabnya. Untuk informasi selengkapnya, lihat Acara perubahan status tugas Amazon ECS.

Sebuah cluster dapat berisi campuran penyedia kapasitas grup Fargate dan Auto Scaling. Namun, strategi penyedia kapasitas hanya dapat berisi penyedia kapasitas grup Fargate atau Auto Scaling, tetapi tidak keduanya. Untuk informasi selengkapnya, lihat Penyedia Kapasitas Grup Auto Scaling.

Pertimbangkan hal berikut saat menggunakan penyedia kapasitas:

- Anda harus mengaitkan penyedia kapasitas dengan klaster sebelum mengaitkannya dengan strategi penyedia kapasitas.
- Anda dapat menentukan maksimal 20 penyedia kapasitas untuk strategi penyedia kapasitas.
- Anda tidak dapat memperbarui layanan menggunakan penyedia kapasitas grup Auto Scaling untuk menggunakan penyedia kapasitas Fargate. Kebalikannya juga terjadi.
- Dalam strategi penyedia kapasitas, jika tidak ada `weight` nilai yang ditentukan untuk penyedia kapasitas di konsol, maka nilai default 1 digunakan. Jika menggunakan API atau AWS CLI, nilai default `0` digunakan.
- Ketika beberapa penyedia kapasitas ditentukan dalam strategi penyedia kapasitas, setidaknya salah satu penyedia kapasitas harus memiliki nilai bobot yang lebih besar dari nol. Setiap penyedia kapasitas dengan bobot nol tidak digunakan untuk menempatkan tugas. Jika Anda menentukan beberapa penyedia kapasitas dalam strategi dengan bobot nol yang sama, maka `CreateService` tindakan apa pun `RunTask` atau yang menggunakan strategi penyedia kapasitas gagal.
- Dalam strategi penyedia kapasitas, hanya satu penyedia kapasitas yang dapat memiliki nilai dasar yang ditentukan. Jika tidak ada nilai dasar yang ditentukan, nilai default nol digunakan.
- Sebuah cluster dapat berisi campuran penyedia kapasitas grup Auto Scaling dan penyedia kapasitas Fargate. Namun, strategi penyedia kapasitas hanya dapat berisi grup Auto Scaling atau penyedia kapasitas Fargate, tetapi tidak keduanya.

- Cluster dapat berisi campuran layanan dan tugas mandiri yang menggunakan penyedia kapasitas dan jenis peluncuran. Layanan dapat diperbarui untuk menggunakan strategi penyedia kapasitas daripada jenis peluncuran. Namun, Anda harus memaksa penerapan baru saat melakukannya.

Pemberitahuan penghentian Fargate Spot

Selama periode permintaan yang sangat tinggi, kapasitas Fargate Spot mungkin tidak tersedia. Hal ini dapat menyebabkan tugas Fargate Spot tertunda. Ketika ini terjadi, layanan Amazon ECS mencoba lagi meluncurkan tugas hingga kapasitas yang diperlukan tersedia. Fargate tidak menggantikan kapasitas Spot dengan kapasitas on-demand.

Ketika tugas yang menggunakan kapasitas Fargate Spot dihentikan karena gangguan Spot, peringatan dua menit dikirim sebelum tugas dihentikan. Peringatan dikirim sebagai peristiwa perubahan status tugas ke Amazon EventBridge dan sebagai sinyal SIGTERM ke tugas yang sedang berjalan. Jika Anda menggunakan Fargate Spot sebagai bagian dari layanan, maka dalam skenario ini penjadwal layanan menerima sinyal interupsi dan mencoba meluncurkan tugas tambahan di Fargate Spot jika ada kapasitas yang tersedia. Layanan dengan hanya satu tugas terganggu sampai kapasitas tersedia. Untuk informasi lebih lanjut tentang shutdown yang anggun, lihat Penutupan anggun dengan ECS.

Untuk memastikan bahwa kontainer Anda keluar dengan anggun sebelum tugas berhenti, Anda dapat mengonfigurasi hal berikut:

- Nilai `stopTimeout` selama `120` detik atau kurang dapat ditentukan dalam ketentuan kontainer yang digunakan oleh tugas. `stopTimeout`Nilai defaultnya adalah 30 detik. Anda dapat menentukan `stopTimeout` nilai yang lebih panjang untuk memberi diri Anda lebih banyak waktu antara saat peristiwa perubahan status tugas diterima dan titik waktu ketika penampung dihentikan secara paksa. Untuk informasi selengkapnya, lihat Waktu habis kontainer.
- Sinyal SIGTERM harus diterima dari dalam kontainer untuk melakukan tindakan pembersihan. Kegagalan untuk memproses sinyal ini mengakibatkan tugas menerima sinyal SIGKILL setelah dikonfigurasi `stopTimeout` dan dapat mengakibatkan kehilangan data atau korupsi.

Berikut ini adalah cuplikan dari peristiwa perubahan status tugas. Cuplikan ini menampilkan alasan berhenti dan kode berhenti untuk gangguan Fargate Spot.

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
```

```
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:task/b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
  ],
  "detail": {
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "STOPPED",
    "lastStatus": "RUNNING",
    "stoppedReason": "Your Spot Task was interrupted.",
    "stopCode": "SpotInterruption",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6fEXAMPLE",
    ...
  }
}
```

Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa perubahan status tugas Amazon ECS. Anda dapat secara opsional menentukan cluster di `detail` lapangan. Melakukannya berarti Anda akan menerima peristiwa perubahan status tugas untuk cluster itu. Untuk informasi selengkapnya tentang membuat EventBridge aturan, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "clusterArn": [
            "arn:aws:ecs:us-west-2:111122223333:cluster/default"
        ]
    }
}
```

Membuat cluster Amazon ECS untuk jenis peluncuran Fargate

Anda membuat klaster untuk menentukan infrastruktur tugas dan layanan yang dijalankan.

Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan menetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”. Konsol Amazon ECS menciptakan sumber daya yang dibutuhkan oleh cluster Amazon ECS dengan membuat AWS CloudFormation tumpukan.

Konsol secara otomatis mengaitkan penyedia kapasitas Fargate dan Fargate Spot dengan cluster.

Anda dapat mengubah opsi berikut:

- Tambahkan namespace ke cluster.

  Namespace memungkinkan layanan yang Anda buat di cluster dapat terhubung ke layanan lain di namespace tanpa konfigurasi tambahan. Untuk informasi selengkapnya, lihat Interkoneksi layanan Amazon ECS.
- Aktifkan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan, atau Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan”.

  Pada tanggal 2 Desember 2024, AWS merilis Container Insights dengan peningkatan observabilitas untuk Amazon ECS. Versi ini mendukung peningkatan observabilitas untuk cluster Amazon ECS menggunakan jenis peluncuran Amazon dan EC2 Fargate. Setelah mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang disempurnakan di Amazon ECS, Container Insights secara otomatis mengumpulkan telemetri infrastruktur terperinci dari tingkat cluster hingga level container di lingkungan Anda dan menampilkan data Anda di dasbor yang menampilkan berbagai metrik dan dimensi. Anda kemudian dapat menggunakan out-of-the-box dasbor ini di konsol Container Insights untuk lebih memahami kesehatan dan kinerja container Anda, dan untuk mengurangi masalah lebih cepat dengan mengidentifikasi anomali.

  Kami menyarankan Anda menggunakan Wawasan Kontainer dengan observabilitas yang ditingkatkan, bukan Wawasan Kontainer karena ini memberikan visibilitas mendetail di lingkungan penampung Anda, sehingga mengurangi waktu rata-rata untuk resolusi.
- Tambahkan tag untuk membantu Anda mengidentifikasi klaster Anda.

- Tetapkan AWS KMS kunci untuk penyimpanan terkelola Anda. Untuk informasi tentang cara membuat kunci, lihat Membuat kunci KMS di Panduan AWS Key Management Service Pengguna.
- Tetapkan AWS KMS kunci untuk penyimpanan sementara Fargate Anda. Untuk informasi tentang cara membuat kunci, lihat Membuat kunci KMS di Panduan AWS Key Management Service Pengguna.

Prosedur

Untuk membuat cluster baru (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, konfigurasikan berikut ini:
   - Untuk nama Cluster, masukkan nama unik.

     Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
   - (Opsional) Agar namespace yang digunakan untuk Service Connect berbeda dari nama cluster, untuk Namespace, masukkan nama yang unik.
6. (Opsional) Gunakan Wawasan Kontainer, perluas Pemantauan, lalu pilih salah satu opsi berikut:
   - Untuk menggunakan Wawasan Kontainer yang direkomendasikan dengan kemampuan observasi yang ditingkatkan, pilih Wawasan Kontainer dengan observabilitas yang ditingkatkan.
   - Untuk menggunakan Container Insights, pilih Container Insights.
7. (Opsional), Di bawah Enkripsi, Anda dapat mengonfigurasi hal berikut:
   - Enkripsi data Anda di penyimpanan sementara Fargate. Di bawah Enkripsi, untuk penyimpanan fana Fargate, masukkan ARN AWS KMS kunci yang ingin Anda gunakan untuk mengenkripsi data penyimpanan fana Fargate.
   - Enkripsi data pada penyimpanan terkelola. Di bawah Enkripsi, untuk penyimpanan Terkelola, masukkan ARN AWS KMS kunci yang ingin Anda gunakan untuk mengenkripsi data penyimpanan terkelola.

8. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
9. Pilih Create (Buat).

Langkah selanjutnya

Setelah Anda membuat cluster, Anda dapat membuat definisi tugas untuk aplikasi Anda dan kemudian menjalankannya sebagai tugas mandiri, atau sebagai bagian dari layanan. Untuk informasi selengkapnya, lihat berikut ini:

- Definisi tugas Amazon ECS
- Menjalankan aplikasi sebagai tugas Amazon ECS
- Membuat penyebaran pembaruan bergulir Amazon ECS

Penyedia kapasitas Amazon ECS untuk jenis EC2 peluncuran

Saat menggunakan EC2 instans Amazon untuk kapasitas Anda, Anda menggunakan grup Auto Scaling untuk mengelola instans EC2 Amazon yang terdaftar di klaster mereka. Auto Scaling membantu memastikan bahwa Anda memiliki jumlah EC2 instans Amazon yang benar yang tersedia untuk menangani pemuatan aplikasi.

Anda dapat menggunakan fitur penskalaan terkelola agar Amazon ECS mengelola tindakan penskalaan dan penskalaan grup Auto Scaling, atau Anda dapat mengelola tindakan penskalaan sendiri. Untuk informasi selengkapnya, lihat Secara otomatis mengelola kapasitas Amazon ECS dengan penskalaan otomatis cluster.

Kami menyarankan Anda membuat grup Auto Scaling kosong baru. Jika Anda menggunakan grup Auto Scaling yang ada, EC2 instans Amazon apa pun yang terkait dengan grup yang sudah berjalan dan terdaftar ke klaster Amazon ECS sebelum grup Auto Scaling digunakan untuk membuat

penyedia kapasitas mungkin tidak terdaftar dengan benar pada penyedia kapasitas. Ini dapat menyebabkan masalah saat menggunakan penyedia kapasitas dalam strategi penyedia kapasitas. Gunakan `DescribeContainerInstances` untuk mengonfirmasi apakah instance kontainer dikaitkan dengan penyedia kapasitas atau tidak.

> **Note**
>
> Untuk membuat grup Auto Scaling kosong, atur hitungan yang diinginkan ke nol. Setelah Anda membuat penyedia kapasitas dan menghubungkannya dengan cluster, Anda kemudian dapat menskalakannya.
> Saat Anda menggunakan konsol Amazon ECS, Amazon ECS membuat template EC2 peluncuran Amazon dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation Mereka diawali dengan`EC2ContainerService-<ClusterName>`. Anda dapat menggunakan grup Auto Scaling sebagai penyedia kapasitas untuk cluster tersebut.

Sebaiknya gunakan pengurasan instans terkelola untuk memungkinkan penghentian EC2 instans Amazon secara anggun yang tidak akan mengganggu beban kerja Anda. Fitur ini aktif secara default. Untuk informasi selengkapnya, lihat Hentikan beban kerja Amazon ECS dengan aman yang berjalan pada instans EC2

Pertimbangkan hal berikut saat menggunakan penyedia kapasitas grup Auto Scaling di konsol:

- Grup Auto Scaling harus memiliki `MaxSize` lebih dari nol untuk diskalakan.
- Grup Auto Scaling tidak dapat memiliki pengaturan pembobotan instance.
- Jika grup Auto Scaling tidak dapat menskalakan untuk mengakomodasi jumlah tugas yang dijalankan, tugas gagal bertransisi di luar status. `PROVISIONING`
- Jangan mengubah sumber daya kebijakan penskalaan yang terkait dengan grup Auto Scaling yang dikelola oleh penyedia kapasitas.
- Jika penskalaan terkelola diaktifkan saat Anda membuat penyedia kapasitas, jumlah yang diinginkan grup Auto Scaling dapat diatur. `0` Saat penskalaan terkelola diaktifkan, Amazon ECS mengelola tindakan penskalaan dan penskalaan grup Auto Scaling.
- Anda harus mengaitkan penyedia kapasitas dengan klaster sebelum mengaitkannya dengan strategi penyedia kapasitas.
- Anda dapat menentukan maksimal 20 penyedia kapasitas untuk strategi penyedia kapasitas.

- Anda tidak dapat memperbarui layanan menggunakan penyedia kapasitas grup Auto Scaling untuk menggunakan penyedia kapasitas Fargate. Kebalikannya juga terjadi.
- Dalam strategi penyedia kapasitas, jika tidak ada `weight` nilai yang ditentukan untuk penyedia kapasitas di konsol, maka nilai default 1 digunakan. Jika menggunakan API atau AWS CLI, nilai default `0` digunakan.
- Ketika beberapa penyedia kapasitas ditentukan dalam strategi penyedia kapasitas, setidaknya salah satu penyedia kapasitas harus memiliki nilai bobot yang lebih besar dari nol. Setiap penyedia kapasitas dengan bobot nol tidak digunakan untuk menempatkan tugas. Jika Anda menentukan beberapa penyedia kapasitas dalam strategi dengan bobot nol yang sama, maka `CreateService` tindakan apa pun `RunTask` atau yang menggunakan strategi penyedia kapasitas gagal.
- Dalam strategi penyedia kapasitas, hanya satu penyedia kapasitas yang dapat memiliki nilai dasar yang ditentukan. Jika tidak ada nilai dasar yang ditentukan, nilai default nol digunakan.
- Sebuah cluster dapat berisi campuran penyedia kapasitas grup Auto Scaling dan penyedia kapasitas Fargate. Namun, strategi penyedia kapasitas hanya dapat berisi grup Auto Scaling atau penyedia kapasitas Fargate, tetapi tidak keduanya.
- Cluster dapat berisi campuran layanan dan tugas mandiri yang menggunakan penyedia kapasitas dan jenis peluncuran. Layanan dapat diperbarui untuk menggunakan strategi penyedia kapasitas daripada jenis peluncuran. Namun, Anda harus memaksa penerapan baru saat melakukannya.
- Amazon ECS mendukung kolam hangat Amazon EC2 Auto Scaling. Kolam hangat adalah sekelompok EC2 instance Amazon pra-inisialisasi yang siap digunakan. Kapan pun aplikasi Anda perlu diskalakan, Amazon EC2 Auto Scaling menggunakan instance yang telah diinisialisasi sebelumnya dari kolam hangat daripada meluncurkan instance dingin. Hal ini memungkinkan untuk setiap proses inisialisasi akhir untuk berjalan sebelum instance ditempatkan ke dalam layanan. Untuk informasi selengkapnya, lihat Mengonfigurasi instans pra-inisialisasi untuk grup Auto Scaling Amazon ECS Anda.

Untuk informasi selengkapnya tentang membuat template peluncuran Amazon EC2 Auto Scaling, lihat templat peluncuran Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling. Untuk informasi selengkapnya tentang membuat grup EC2 Auto Scaling Amazon, lihat grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.

Pertimbangan keamanan instans EC2 penampung Amazon untuk Amazon ECS

Anda harus mempertimbangkan satu instance kontainer dan aksesnya dalam model ancaman Anda. Misalnya, satu tugas yang terpengaruh mungkin dapat memanfaatkan izin IAM dari tugas yang tidak terinfeksi pada instance yang sama.

Kami menyarankan Anda menggunakan yang berikut ini untuk membantu mencegah hal ini:

- Jangan gunakan hak administrator saat menjalankan tugas Anda.
- Tetapkan peran tugas dengan akses paling tidak istimewa ke tugas Anda.

  Agen kontainer secara otomatis membuat token dengan ID kredensyal unik yang digunakan untuk mengakses sumber daya Amazon ECS.
- Untuk mencegah container yang dijalankan oleh tugas yang menggunakan mode `awsvpc` jaringan mengakses informasi kredensi yang diberikan ke profil EC2 instans Amazon, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, tetapkan variabel konfigurasi `ECS_AWSVPC_BLOCK_IMDS` agen ke true dalam file konfigurasi agen dan restart agen.
- Gunakan Amazon GuardDuty Runtime Monitoring untuk mendeteksi ancaman untuk cluster dan container dalam lingkungan Anda AWS . Runtime Monitoring menggunakan agen GuardDuty keamanan yang menambahkan visibilitas runtime ke beban kerja Amazon ECS individual, misalnya, akses file, eksekusi proses, dan koneksi jaringan. Untuk informasi selengkapnya, lihat GuardDutyRuntime Monitoring di Panduan GuardDuty Pengguna.

Membuat cluster Amazon ECS untuk jenis EC2 peluncuran Amazon

Anda membuat klaster untuk menentukan infrastruktur tugas dan layanan yang dijalankan.

Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan menetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”. Konsol Amazon ECS menyediakan cara sederhana untuk membuat sumber daya yang dibutuhkan oleh cluster Amazon ECS dengan membuat AWS CloudFormation tumpukan.

Untuk membuat proses pembuatan cluster semudah mungkin, konsol memiliki pilihan default untuk banyak pilihan yang kami jelaskan di bawah ini. Ada juga panel bantuan yang tersedia untuk porsi besar bagian di konsol yang menyediakan konteks lebih lanjut.

Anda dapat mendaftarkan EC2 instans Amazon saat membuat klaster atau mendaftarkan instance tambahan dengan klaster setelah dibuat.

Anda dapat memodifikasi opsi default berikut:

- Ubah subnet tempat instans Anda diluncurkan
- Mengubah grup keamanan yang digunakan untuk mengontrol lalu lintas ke instance container
- Tambahkan namespace ke cluster.

  Namespace memungkinkan layanan yang Anda buat di cluster dapat terhubung ke layanan lain di namespace tanpa konfigurasi tambahan. Untuk informasi selengkapnya, lihat Interkoneksi layanan Amazon ECS.
- Aktifkan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan, atau Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan”.

  Pada tanggal 2 Desember 2024, AWS merilis Container Insights dengan peningkatan observabilitas untuk Amazon ECS. Versi ini mendukung peningkatan observabilitas untuk cluster Amazon ECS menggunakan jenis peluncuran Amazon dan EC2 Fargate. Setelah mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang disempurnakan di Amazon ECS, Container Insights secara otomatis mengumpulkan telemetri infrastruktur terperinci dari tingkat cluster hingga level container di lingkungan Anda dan menampilkan data Anda di dasbor yang menampilkan berbagai metrik dan dimensi. Anda kemudian dapat menggunakan out-of-the-box dasbor ini di konsol Container Insights untuk lebih memahami kesehatan dan kinerja container Anda, dan untuk mengurangi masalah lebih cepat dengan mengidentifikasi anomali.

  Kami menyarankan Anda menggunakan Wawasan Kontainer dengan observabilitas yang ditingkatkan, bukan Wawasan Kontainer karena ini memberikan visibilitas mendetail di lingkungan penampung Anda, sehingga mengurangi waktu rata-rata untuk resolusi.
- Tetapkan AWS KMS kunci untuk penyimpanan terkelola Anda. Untuk informasi tentang cara membuat kunci, lihat Membuat kunci KMS di Panduan AWS Key Management Service Pengguna.

- Tetapkan AWS KMS kunci untuk penyimpanan sementara Fargate Anda. Untuk informasi tentang cara membuat kunci, lihat Membuat kunci KMS di Panduan AWS Key Management Service Pengguna.
- Tambahkan tag untuk membantu Anda mengidentifikasi klaster Anda.

Opsi grup Auto Scaling

Saat menggunakan EC2 instans Amazon, Anda harus menentukan grup Auto Scaling untuk mengelola infrastruktur tempat tugas dan layanan Anda berjalan.

Bila Anda memilih untuk membuat grup Auto Scaling baru, grup Auto Scaling secara otomatis dikonfigurasi untuk perilaku berikut:

- Amazon ECS mengelola tindakan scale-in dan scale-out dari grup Auto Scaling.
- Amazon ECS tidak akan mencegah EC2 instans Amazon yang berisi tugas dan yang ada dalam grup Auto Scaling dihentikan selama tindakan penskalaan. Untuk informasi selengkapnya, lihat Perlindungan Instance di Panduan AWS Auto Scaling Pengguna.

Anda mengonfigurasi properti grup Auto Scaling berikut yang menentukan jenis dan jumlah instance yang akan diluncurkan untuk grup:

- AMI Amazon ECS yang dioptimalkan.
- Tipe instans.
- Key pair SSH yang membuktikan identitas Anda saat Anda terhubung ke instance. Untuk informasi tentang cara membuat kunci SSH, lihat pasangan EC2 kunci Amazon dan instance Linux di EC2 Panduan Pengguna Amazon.
- Jumlah minimum instans yang akan diluncurkan untuk grup Auto Scaling.
- Jumlah maksimum instans yang dimulai untuk grup Auto Scaling.

  Agar grup dapat ditingkatkan, maksimum harus lebih besar dari 0.

Amazon ECS membuat template peluncuran Amazon EC2 Auto Scaling dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation Nilai yang Anda tentukan untuk AMI, tipe instance, dan key pair SSH adalah bagian dari template peluncuran. Template diawali dengan`EC2ContainerService-<ClusterName>`, yang membuatnya mudah diidentifikasi. Grup Auto Scaling diawali dengan. `<ClusterName>-ECS-Infra-ECSAutoScalingGroup`

Instans yang diluncurkan untuk grup Auto Scaling menggunakan template peluncuran.

Opsi jaringan

Secara default instance diluncurkan ke subnet default untuk Wilayah. Grup keamanan, yang mengontrol lalu lintas ke instans kontainer Anda, saat ini terkait dengan subnet digunakan. Anda dapat mengubah subnet dan grup keamanan untuk instans.

Anda dapat memilih subnet yang ada. Anda dapat menggunakan grup keamanan yang ada, atau membuat yang baru, Saat membuat grup keamanan baru, Anda harus menentukan setidaknya satu aturan masuk.

Aturan masuk menentukan lalu lintas apa yang dapat mencapai instans kontainer Anda dan menyertakan properti berikut:

- Protokol untuk memungkinkan
- Kisaran port untuk memungkinkan
- Lalu lintas masuk (sumber)

Untuk mengizinkan lalu lintas masuk dari alamat tertentu atau blok CIDR, gunakan Custom for Source dengan CIDR yang diizinkan.

Untuk memungkinkan lalu lintas masuk dari semua tujuan, gunakan Anywhere for Source. Ini secara otomatis menambahkan blok CIDR 0.0.0.0/0 dan: :/0 blok IPv4 CIDR. IPv6

Untuk mengizinkan lalu lintas masuk dari komputer lokal Anda, gunakan grup Sumber untuk Sumber. Ini secara otomatis menambahkan alamat IP saat ini dari komputer lokal Anda sebagai sumber yang diizinkan.

Untuk membuat cluster baru (konsol Amazon ECS)

Sebelum Anda mulai, tetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, konfigurasikan berikut ini:

- Untuk nama Cluster, masukkan nama unik.

  Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.

- (Opsional) Agar namespace yang digunakan untuk Service Connect berbeda dari nama cluster, untuk Namespace, masukkan nama yang unik.

6. Tambahkan EC2 instans Amazon ke klaster Anda, perluas Infrastruktur, lalu pilih EC2 instans Amazon.

   Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas:

   a. Untuk menggunakan grup Auto Scaling yang ada, dari grup Auto Scaling (ASG), pilih grup.

   b. Untuk membuat grup Auto Scaling, dari grup Auto Scaling (ASG), pilih Buat grup baru, lalu berikan detail berikut tentang grup:

      - Untuk model Provisioning, pilih apakah akan menggunakan instans On-Demand atau Instans Spot.
      - Jika Anda memilih untuk menggunakan Instans Spot, untuk Strategi Alokasi, pilih kumpulan kapasitas Spot (tipe instans dan Availability Zones) yang digunakan untuk instans.

        Untuk sebagian besar beban kerja, Anda dapat memilih Kapasitas harga dioptimalkan.

        Untuk informasi selengkapnya, lihat Strategi alokasi untuk Instans Spot di EC2 Panduan Pengguna Amazon.
      - Untuk Container misalnya Amazon Machine Image (AMI), pilih AMI Amazon ECS yang dioptimalkan untuk instans grup Auto Scaling.
      - EC2 Misalnya jenis, pilih jenis instance untuk beban kerja Anda.

        Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa.
      - EC2 Misalnya peran, pilih peran instance kontainer yang ada, atau Anda dapat membuat yang baru.

        Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.
      - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling.

- Untuk key pair SSH, pilih pair yang membuktikan identitas Anda saat Anda terhubung ke instance.
- Untuk memungkinkan gambar dan penyimpanan yang lebih besar, untuk ukuran volume Root EBS, masukkan nilai dalam GiB.

7. (Opsional) Untuk mengubah VPC dan subnet, di bawah EC2instans Jaringan untuk Amazon, lakukan salah satu operasi berikut:

   - Untuk menghapus subnet, di bawah Subnet, pilih X untuk setiap subnet yang ingin Anda hapus.
   - Untuk mengubah ke VPC selain VPC default, di bawah VPC, pilih VPC yang ada, dan kemudian di bawah Subnet, pilih subnet.
   - Pilih grup keamanan. Di bawah Grup keamanan, pilih salah satu opsi berikut:
     - Untuk menggunakan grup keamanan yang ada, pilih Gunakan grup keamanan yang ada, lalu pilih grup keamanan.
     - Untuk membuat grup keamanan, pilih Buat grup keamanan baru. Kemudian, pilih Tambahkan aturan untuk setiap aturan masuk.

       Untuk informasi tentang aturan masuk, lihatOpsi jaringan.
   - Untuk secara otomatis menetapkan alamat IP publik ke instans EC2 penampung Amazon Anda, untuk Tetapkan IP publik secara otomatis, pilih salah satu opsi berikut:
     - Gunakan pengaturan subnet — Tetapkan alamat IP publik ke instance ketika subnet yang diluncurkan instance adalah subnet publik.
     - Aktifkan — Tetapkan alamat IP publik ke instans.

8. (Opsional) Gunakan Wawasan Kontainer, perluas Pemantauan, lalu pilih salah satu opsi berikut:

   - Untuk menggunakan Wawasan Kontainer yang direkomendasikan dengan kemampuan observasi yang ditingkatkan, pilih Wawasan Kontainer dengan observabilitas yang ditingkatkan.
   - Untuk menggunakan Container Insights, pilih Container Insights.

9. (Opsional)

   Jika Anda menggunakan Runtime Monitoring dengan opsi manual dan Anda ingin cluster ini dipantau GuardDuty, pilih Tambahkan tag dan lakukan hal berikut:

   - Untuk Key, masukkan **`guardDutyRuntimeMonitoringManaged`**

- Untuk Nilai, masukkan **`true`**.

10. (Opsional) Enkripsi data pada penyimpanan terkelola. Di bawah Enkripsi, untuk penyimpanan Terkelola, masukkan ARN AWS KMS kunci yang ingin Anda gunakan untuk mengenkripsi data penyimpanan terkelola.
11. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

    [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

    - Untuk Kunci, masukkan nama kunci.
    - Untuk Nilai, masukkan nilai kunci.

    [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
12. Pilih Create (Buat).

Langkah selanjutnya

Setelah Anda membuat cluster, Anda dapat membuat definisi tugas untuk aplikasi Anda dan kemudian menjalankannya sebagai tugas mandiri, atau sebagai bagian dari layanan. Untuk informasi selengkapnya, lihat berikut ini:

- Definisi tugas Amazon ECS
- Menjalankan aplikasi sebagai tugas Amazon ECS
- Membuat penyebaran pembaruan bergulir Amazon ECS

Secara otomatis mengelola kapasitas Amazon ECS dengan penskalaan otomatis cluster

Amazon ECS dapat mengelola penskalaan EC2 instans Amazon yang terdaftar ke klaster Anda. Ini disebut sebagai penskalaan otomatis cluster Amazon ECS. Anda mengaktifkan penskalaan terkelola saat membuat penyedia kapasitas grup Auto Scaling Amazon ECS. Kemudian, Anda menetapkan persentase target (the`targetCapacity`) untuk pemanfaatan instance dalam grup Auto Scaling ini. Amazon ECS membuat dua CloudWatch metrik khusus dan kebijakan penskalaan pelacakan target untuk grup Auto Scaling Anda. Amazon ECS kemudian mengelola tindakan scale-in dan scale-out berdasarkan pemanfaatan sumber daya yang digunakan tugas Anda.

Untuk setiap penyedia kapasitas grup Auto Scaling yang terkait dengan kluster, Amazon ECS membuat dan mengelola sumber daya berikut:

- CloudWatch Alarm nilai metrik rendah
- CloudWatch Alarm nilai metrik tinggi
- Kebijakan penskalaan pelacakan target

> **Note**
>
> Amazon ECS membuat kebijakan penskalaan pelacakan target dan menempelkannya ke grup Auto Scaling. Untuk memperbarui kebijakan penskalaan pelacakan target, perbarui pengaturan penskalaan terkelola penyedia kapasitas, daripada memperbarui kebijakan penskalaan secara langsung.

Saat Anda menonaktifkan penskalaan terkelola atau memisahkan penyedia kapasitas dari kluster, Amazon ECS akan menghapus CloudWatch metrik dan sumber daya kebijakan penskalaan pelacakan target.

Amazon ECS menggunakan metrik berikut untuk menentukan tindakan apa yang harus diambil:

`CapacityProviderReservation`

Persentase instance kontainer yang digunakan untuk penyedia kapasitas tertentu. Amazon ECS menghasilkan metrik ini.

Amazon ECS menetapkan `CapacityProviderReservation` nilai ke angka antara 0-100. Amazon ECS menggunakan rumus berikut untuk mewakili rasio berapa banyak kapasitas yang tersisa dalam grup Auto Scaling. Kemudian, Amazon ECS menerbitkan metrik ke. CloudWatch Untuk informasi selengkapnya tentang cara penghitungan metrik, lihat Deep Dive di Auto Scaling Amazon ECS Cluster.

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

`DesiredCapacity`

Jumlah kapasitas untuk grup Auto Scaling. Metrik ini tidak dipublikasikan ke CloudWatch.

Amazon ECS menerbitkan `CapacityProviderReservation` metrik ke CloudWatch dalam namespace. `AWS/ECS/ManagedScaling` `CapacityProviderReservation`Metrik menyebabkan salah satu tindakan berikut terjadi:

`CapacityProviderReservation`Nilainya sama dengan **`targetCapacity`**

Grup Auto Scaling tidak perlu menskalakan atau memperkecil skala. Target persentase pemanfaatan telah tercapai.

`CapacityProviderReservation`Nilai lebih besar dari **`targetCapacity`**

Ada lebih banyak tugas menggunakan persentase kapasitas yang lebih tinggi daripada `targetCapacity` persentase Anda. Peningkatan nilai `CapacityProviderReservation` metrik menyebabkan CloudWatch alarm terkait bertindak. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Grup Auto Scaling menggunakan nilai ini untuk meluncurkan EC2 instance, dan kemudian mendaftarkannya ke cluster.

Ketika `targetCapacity` adalah nilai default 100%, tugas baru berada dalam `PENDING` status selama scale-out karena tidak ada kapasitas yang tersedia pada instance untuk menjalankan tugas. Setelah instans baru mendaftar dengan ECS, tugas-tugas ini akan dimulai pada instans baru.

`CapacityProviderReservation`Nilainya kurang dari **`targetCapacity`**

Ada lebih sedikit tugas yang menggunakan persentase kapasitas yang lebih rendah daripada `targetCapacity` persentase Anda dan setidaknya ada satu contoh yang dapat dihentikan. Nilai `CapacityProviderReservation` metrik yang menurun menyebabkan CloudWatch alarm terkait bertindak. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Grup Auto Scaling menggunakan nilai ini untuk menghentikan instance EC2 kontainer, dan kemudian membatalkan pendaftarannya dari cluster.

Grup Auto Scaling mengikuti kebijakan penghentian grup untuk menentukan instance mana yang dihentikan terlebih dahulu selama peristiwa penskalaan. Selain itu, ini menghindari instance dengan pengaturan perlindungan skala masuk instance diaktifkan. Penskalaan otomatis cluster dapat mengelola instance mana yang memiliki setelan perlindungan penskalaan instans jika Anda mengaktifkan perlindungan terminasi terkelola. Untuk informasi selengkapnya tentang perlindungan terminasi terkelola, lihatKontrol instans Amazon ECS berakhir. Untuk informasi selengkapnya tentang cara grup Auto Scaling menghentikan instans, lihat Mengontrol instans Auto Scaling mana yang dihentikan selama penskalaan di Panduan Pengguna Amazon Auto Scaling. EC2

Pertimbangkan hal berikut saat menggunakan penskalaan otomatis cluster:

- Jangan mengubah atau mengelola kapasitas yang diinginkan untuk grup Auto Scaling yang terkait dengan penyedia kapasitas dengan kebijakan penskalaan apa pun selain yang dikelola Amazon ECS.
- Saat Amazon ECS keluar dari 0 instans, Amazon ECS akan secara otomatis meluncurkan 2 instans.
- Amazon ECS menggunakan peran IAM `AWSServiceRoleForECS` terkait layanan untuk izin yang diperlukan untuk memanggil atas nama Anda. AWS Auto Scaling Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Saat menggunakan penyedia kapasitas dengan grup Auto Scaling, pengguna, grup, atau peran yang membuat penyedia kapasitas memerlukan izin. `autoscaling:CreateOrUpdateTags` Ini karena Amazon ECS menambahkan tag ke grup Auto Scaling saat mengaitkannya dengan penyedia kapasitas.

  > **Important**
  >
  > Pastikan alat apa pun yang Anda gunakan tidak menghapus `AmazonECSManaged` tag dari grup Auto Scaling. Jika tag ini dihapus, Amazon ECS tidak dapat mengelola penskalaan.

- Penskalaan otomatis cluster tidak mengubah MinimumCapacityatau MaximumCapacityuntuk grup. Agar kelompok dapat diskalakan, nilai untuk MaximumCapacityharus lebih besar dari nol.
- Saat Auto Scaling (penskalaan terkelola) diaktifkan, penyedia kapasitas hanya dapat dihubungkan ke satu cluster pada saat yang bersamaan. Jika penyedia kapasitas Anda telah mengelola penskalaan dimatikan, Anda dapat mengaitkannya dengan beberapa cluster.
- Saat penskalaan terkelola dimatikan, penyedia kapasitas tidak akan menskalakan atau memperkecil skala. Anda dapat menggunakan strategi penyedia kapasitas untuk menyeimbangkan tugas Anda antara penyedia kapasitas.
- `binpack`Strategi adalah strategi yang paling efisien dalam hal kapasitas.
- Ketika kapasitas target kurang dari 100%, strategi penempatan yang perlu menggunakan `binpack` strategi harus memiliki urutan yang lebih tinggi daripada `spread` strategi. Ini mencegah penyedia kapasitas dari skala hingga setiap tugas memiliki instance khusus atau batasnya tercapai.

Aktifkan penskalaan otomatis cluster

Anda dapat mengaktifkan penskalaan otomatis cluster dengan menggunakan Konsol atau. AWS CLI

Saat Anda membuat klaster untuk jenis EC2 peluncuran menggunakan konsol, Amazon ECS membuat grup Auto Scaling atas nama Anda dan menetapkan kapasitas target. Untuk informasi selengkapnya, lihat Membuat cluster Amazon ECS untuk jenis EC2 peluncuran Amazon.

Anda juga dapat membuat grup Auto Scaling, lalu menetapkannya ke cluster. Untuk informasi selengkapnya, lihat Memperbarui penyedia kapasitas Amazon ECS.

Saat Anda menggunakan AWS CLI, setelah Anda membuat cluster

1. Sebelum Anda membuat penyedia kapasitas, Anda perlu membuat grup Auto Scaling. Untuk informasi selengkapnya, lihat grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.
2. Gunakan `put-cluster-capacity-providers` untuk memodifikasi penyedia kapasitas cluster. Untuk informasi selengkapnya, lihat Mengaktifkan penskalaan otomatis cluster Amazon ECS.

Optimalkan penskalaan otomatis cluster Amazon ECS

Pelanggan yang menjalankan Amazon ECS di Amazon EC2 dapat memanfaatkan penskalaan otomatis cluster untuk mengelola penskalaan grup Auto Scaling EC2 Amazon. Dengan penskalaan otomatis cluster, Anda dapat mengonfigurasi Amazon ECS untuk menskalakan grup Auto Scaling Anda secara otomatis, dan hanya fokus menjalankan tugas Anda. Amazon ECS memastikan grup Auto Scaling masuk dan keluar sesuai kebutuhan tanpa memerlukan intervensi lebih lanjut. Penyedia kapasitas Amazon ECS digunakan untuk mengelola infrastruktur di klaster Anda dengan memastikan ada cukup instance kontainer untuk memenuhi permintaan aplikasi Anda. Untuk mempelajari cara kerja penskalaan otomatis cluster di bawah kap, lihat Deep Dive di Auto Scaling Amazon ECS Cluster.

Penskalaan otomatis cluster bergantung pada integrasi CloudWatch berbasis dengan grup Auto Scaling untuk menyesuaikan kapasitas cluster. Oleh karena itu memiliki latensi inheren yang terkait dengan

- Menerbitkan CloudWatch metrik,
- Waktu yang dibutuhkan metrik `CapacityProviderReservation` untuk melanggar CloudWatch alarm (tinggi dan rendah)

- Waktu yang dibutuhkan oleh EC2 instans Amazon yang baru diluncurkan untuk pemanasan. Anda dapat mengambil tindakan berikut untuk membuat penskalaan otomatis cluster lebih responsif untuk penerapan yang lebih cepat:

Ukuran penskalaan langkah penyedia kapasitas

Penyedia kapasitas Amazon ECS akan grow/shrink menggunakan instans kontainer untuk memenuhi permintaan aplikasi Anda. Jumlah minimum instans yang akan diluncurkan Amazon ECS diatur ke 1 secara default. Ini dapat menambah waktu tambahan untuk penerapan Anda, jika beberapa instance diperlukan untuk menempatkan tugas Anda yang tertunda. Anda dapat meningkatkan `minimumScalingStepSize`melalui Amazon ECS API untuk meningkatkan jumlah minimum instans yang diskalakan atau dikeluarkan Amazon ECS sekaligus. A `maximumScalingStepSize`yang terlalu rendah dapat membatasi berapa banyak instance kontainer yang diskalakan masuk atau keluar sekaligus, yang dapat memperlambat penerapan Anda.

> **Note**
>
> Konfigurasi ini saat ini hanya tersedia melalui `CreateCapacityProvider`atau `UpdateCapacityProvider` APIs.

Periode pemanasan contoh

Periode pemanasan instans adalah periode waktu setelah EC2 instans Amazon yang baru diluncurkan dapat berkontribusi pada CloudWatch metrik untuk grup Auto Scaling. Setelah periode pemanasan yang ditentukan berakhir, instance dihitung ke metrik agregat grup Auto Scaling, dan penskalaan otomatis cluster dilanjutkan dengan iterasi perhitungan berikutnya untuk memperkirakan jumlah instans yang diperlukan.

Nilai default untuk `instanceWarmupPeriod`adalah 300 detik, yang dapat Anda konfigurasikan ke nilai yang lebih rendah melalui `CreateCapacityProvider`atau `UpdateCapacityProvider` APIs untuk penskalaan yang lebih responsif. Kami menyarankan Anda menetapkan nilainya menjadi lebih dari 60 detik sehingga Anda dapat menghindari penyediaan berlebih.

Kapasitas cadangan

Jika penyedia kapasitas Anda tidak memiliki instans kontainer yang tersedia untuk menempatkan tugas, maka perlu meningkatkan (skala) kapasitas klaster dengan meluncurkan EC2 instans Amazon

dengan cepat, dan menunggu hingga boot sebelum dapat meluncurkan kontainer pada mereka. Ini dapat secara signifikan menurunkan tingkat peluncuran tugas. Anda memiliki dua opsi di sini.

Dalam hal ini, memiliki EC2 kapasitas Amazon cadangan yang sudah diluncurkan dan siap menjalankan tugas akan meningkatkan tingkat peluncuran tugas yang efektif. Anda dapat menggunakan `Target Capacity` konfigurasi untuk menunjukkan bahwa Anda ingin mempertahankan kapasitas cadangan di cluster Anda. Misalnya, dengan menetapkan `Target Capacity` 80%, Anda menunjukkan bahwa cluster Anda membutuhkan kapasitas cadangan 20% setiap saat. Kapasitas cadangan ini dapat memungkinkan tugas mandiri apa pun segera diluncurkan, memastikan peluncuran tugas tidak dibatasi. Trade-off untuk pendekatan ini adalah potensi peningkatan biaya untuk menjaga kapasitas cluster cadangan.

Pendekatan alternatif yang dapat Anda pertimbangkan adalah menambahkan ruang kepala ke layanan Anda, bukan ke penyedia kapasitas. Ini berarti bahwa alih-alih mengurangi `Target Capacity` konfigurasi untuk meluncurkan kapasitas cadangan, Anda dapat meningkatkan jumlah replika di layanan Anda dengan memodifikasi metrik penskalaan pelacakan target atau ambang batas penskalaan langkah dari penskalaan otomatis layanan. Perhatikan bahwa pendekatan ini hanya akan membantu untuk beban kerja runcing, tetapi tidak akan berpengaruh saat Anda menerapkan layanan baru dan beralih dari 0 ke N tugas untuk pertama kalinya. Untuk informasi selengkapnya tentang kebijakan penskalaan terkait, lihat Kebijakan Penskalaan Pelacakan Target atau Kebijakan Penskalaan Langkah di Panduan Pengembang Layanan Amazon Elastic Container.

Perilaku penskalaan terkelola Amazon ECS

Jika Anda memiliki penyedia kapasitas grup Auto Scaling yang menggunakan penskalaan terkelola, Amazon ECS memperkirakan jumlah instans optimal yang akan ditambahkan ke klaster Anda dan menggunakan nilainya untuk menentukan berapa banyak instans yang akan diminta atau dirilis.

Perilaku penskalaan keluar yang terkelola

Amazon ECS memilih penyedia kapasitas untuk setiap tugas dengan mengikuti strategi penyedia kapasitas dari layanan, tugas mandiri, atau default cluster. Amazon ECS mengikuti langkah-langkah lainnya untuk penyedia kapasitas tunggal.

Tugas tanpa strategi penyedia kapasitas diabaikan oleh penyedia kapasitas. Tugas yang tertunda yang tidak memiliki strategi penyedia kapasitas tidak akan menyebabkan penyedia kapasitas untuk meningkatkan skala. Tugas atau layanan tidak dapat menetapkan strategi penyedia kapasitas jika tugas atau layanan tersebut menetapkan jenis peluncuran.

Berikut ini menjelaskan perilaku penskalaan keluar secara lebih rinci.

- Kelompokkan semua tugas penyediaan untuk penyedia kapasitas ini sehingga setiap kelompok memiliki persyaratan sumber daya yang sama persis.
- Bila Anda menggunakan beberapa tipe instans dalam grup Auto Scaling, tipe instans dalam grup Auto Scaling akan diurutkan berdasarkan parameternya. Parameter ini termasuk vCPU, memori, antarmuka jaringan elastis (ENIs), port, dan. GPUs Jenis instance terkecil dan terbesar untuk setiap parameter dipilih. Untuk informasi selengkapnya tentang cara memilih jenis instans, lihatInstans EC2 kontainer Amazon untuk Amazon ECS.

  > **Important**
  >
  > Jika sekelompok tugas memiliki persyaratan sumber daya yang lebih besar dari jenis instans terkecil di grup Auto Scaling, maka grup tugas tersebut tidak dapat dijalankan dengan penyedia kapasitas ini. Penyedia kapasitas tidak menskalakan grup Auto Scaling. Tugas tetap di `PROVISIONING` negara bagian.
  > Untuk mencegah tugas tetap berada dalam `PROVISIONING` status, kami sarankan Anda membuat grup Auto Scaling terpisah dan penyedia kapasitas untuk persyaratan sumber daya minimum yang berbeda. Saat menjalankan tugas atau membuat layanan, tambahkan saja penyedia kapasitas ke strategi penyedia kapasitas yang dapat menjalankan tugas pada jenis instans terkecil di grup Auto Scaling. Untuk parameter lain, Anda dapat menggunakan batasan penempatan

- Untuk setiap grup tugas, Amazon ECS menghitung jumlah instance yang diperlukan untuk menjalankan tugas yang tidak ditempatkan. Perhitungan ini menggunakan `binpack` strategi. Strategi ini memperhitungkan vCPU, memori, antarmuka jaringan elastis (ENI), port, dan GPUs persyaratan tugas. Ini juga memperhitungkan ketersediaan sumber daya dari EC2 instans Amazon. Nilai untuk jenis instans terbesar diperlakukan sebagai jumlah instans terhitung maksimum. Nilai untuk jenis instance terkecil digunakan sebagai perlindungan. Jika jenis instance terkecil tidak dapat menjalankan setidaknya satu instance tugas, perhitungan menganggap tugas sebagai tidak kompatibel. Akibatnya, tugas tersebut dikecualikan dari perhitungan scale-out. Ketika semua tugas tidak kompatibel dengan jenis instance terkecil, penskalaan otomatis cluster berhenti dan `CapacityProviderReservation` nilainya tetap pada `targetCapacity` nilainya.
- Amazon ECS menerbitkan `CapacityProviderReservation` metrik sehubungan CloudWatch dengan `minimumScalingStepSize` jika salah satu dari berikut ini adalah kasusnya.
  - Jumlah instans maksimum yang dihitung kurang dari ukuran langkah penskalaan minimum.
  - Nilai yang lebih rendah dari jumlah instans yang dihitung `maximumScalingStepSize` atau maksimum.

- CloudWatch alarm menggunakan `CapacityProviderReservation` metrik untuk penyedia kapasitas. Ketika `CapacityProviderReservation` metrik lebih besar dari `targetCapacity` nilainya, alarm juga meningkatkan grup `DesiredCapacity` Auto Scaling. `targetCapacity`Nilainya adalah pengaturan penyedia kapasitas yang dikirim ke CloudWatch alarm selama fase aktivasi penskalaan otomatis cluster.

  `targetCapacity`Defaultnya adalah 100%.
- Grup Auto Scaling meluncurkan instance tambahan. EC2 Untuk mencegah penyediaan berlebih, Auto Scaling memastikan bahwa kapasitas EC2 instans yang baru diluncurkan distabilkan sebelum meluncurkan instance baru. Auto Scaling memeriksa apakah semua instance yang ada telah melewati `instanceWarmupPeriod` (sekarang dikurangi waktu peluncuran instance). Penskalaan diblokir untuk instance yang ada di dalam. `instanceWarmupPeriod`

  Jumlah detik default untuk instance yang baru diluncurkan untuk pemanasan adalah 300.

Untuk informasi selengkapnya, lihat Menyelam mendalam tentang penskalaan otomatis klaster Amazon ECS.

Pertimbangan skala

Pertimbangkan hal berikut untuk proses scale-out:

- Meskipun ada beberapa kendala penempatan, kami menyarankan Anda hanya menggunakan kendala penempatan `distinctInstance` tugas. Ini mencegah proses scale-out berhenti karena Anda menggunakan batasan penempatan yang tidak kompatibel dengan instance sampel.
- Penskalaan terkelola berfungsi paling baik jika grup Auto Scaling Anda menggunakan jenis instans yang sama atau serupa.
- Ketika proses scale-out diperlukan dan tidak ada instance container yang sedang berjalan, Amazon ECS selalu melakukan scale-out ke dua instance pada awalnya, dan kemudian melakukan proses scale-out atau scale-in tambahan. Setiap scale-out tambahan menunggu periode pemanasan instance. Untuk proses scale-in, Amazon ECS menunggu 15 menit setelah proses scale-out sebelum memulai proses scale-in setiap saat.
- Langkah scale-out kedua harus menunggu sampai `instanceWarmupPeriod` habis masa berlakunya, yang mungkin mempengaruhi batas skala keseluruhan. Jika Anda perlu mengurangi waktu ini, pastikan waktu tersebut `instanceWarmupPeriod` cukup besar untuk EC2 meluncurkan dan memulai agen Amazon ECS (yang mencegah penyediaan berlebih).

- Penskalaan otomatis cluster mendukung Konfigurasi Peluncuran, Template Peluncuran, dan beberapa jenis instans di grup Auto Scaling penyedia kapasitas. Anda juga dapat menggunakan pemilihan tipe instans berbasis atribut tanpa beberapa jenis instance.
- Saat menggunakan grup Auto Scaling dengan instans Sesuai Permintaan dan beberapa jenis instans atau Instans Spot, tempatkan jenis instans yang lebih besar lebih tinggi dalam daftar prioritas dan jangan tentukan bobot. Menentukan bobot tidak didukung saat ini. Untuk informasi selengkapnya, lihat grup Auto Scaling dengan beberapa jenis instans di AWS Auto Scaling Panduan Pengguna.
- Amazon ECS kemudian meluncurkan`minimumScalingStepSize`, jika jumlah instans maksimum yang dihitung kurang dari ukuran langkah penskalaan minimum, atau yang lebih rendah dari nilai hitungan instans terhitung maksimum `maximumScalingStepSize` atau maksimum.
- Jika layanan Amazon ECS atau `run-task` meluncurkan tugas dan instans wadah penyedia kapasitas tidak memiliki sumber daya yang cukup untuk memulai tugas, Amazon ECS membatasi jumlah tugas dengan status ini untuk setiap klaster dan mencegah tugas apa pun melebihi batas ini. Untuk informasi selengkapnya, lihat Kuota layanan.

Perilaku scale-in yang dikelola

Amazon ECS memantau instans kontainer untuk setiap penyedia kapasitas dalam klaster. Saat instance container tidak menjalankan tugas apa pun, instance container dianggap kosong dan Amazon ECS memulai proses scale-in.

CloudWatch alarm scale-in memerlukan 15 titik data (15 menit) sebelum proses scale-in untuk grup Auto Scaling dimulai. Setelah proses penskalaan dimulai hingga Amazon ECS perlu mengurangi jumlah instans kontainer yang terdaftar, grup Auto Scaling menetapkan `DesireCapacity` nilainya menjadi lebih besar dari satu instans dan kurang dari 50% setiap menit.

Saat Amazon ECS meminta scale-out (bila `CapacityProviderReservation` lebih besar dari 100) saat proses scale-in sedang berlangsung, proses scale-in dihentikan dan dimulai dari awal jika diperlukan.

Berikut ini menjelaskan perilaku scale-in secara lebih rinci:

1. Amazon ECS menghitung jumlah instance kontainer yang kosong. Sebuah instance container dianggap kosong bahkan ketika tugas daemon sedang berjalan.
2. Amazon ECS menetapkan `CapacityProviderReservation` nilai ke angka antara 0-100 yang menggunakan rumus berikut untuk mewakili rasio seberapa besar kelompok Auto Scaling

harus relatif terhadap seberapa besar sebenarnya, dinyatakan sebagai persentase. Kemudian, Amazon ECS menerbitkan metrik ke. CloudWatch Untuk informasi selengkapnya tentang cara penghitungan metrik, lihat Deep Dive di Auto Scaling Amazon ECS Cluster

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

3. `CapacityProviderReservation`Metrik menghasilkan CloudWatch alarm. Alarm ini memperbarui `DesiredCapacity` nilai untuk grup Auto Scaling. Kemudian, salah satu tindakan berikut terjadi:
   - Jika Anda tidak menggunakan penghentian terkelola penyedia kapasitas, grup Auto Scaling akan memilih EC2 instans menggunakan kebijakan penghentian grup Auto Scaling dan mengakhiri instans hingga jumlah instans mencapai. EC2 `DesiredCapacity` Instance kontainer kemudian dideregistrasi dari cluster.
   - Jika semua instance container menggunakan proteksi terminasi terkelola, Amazon ECS akan menghapus proteksi scale-in pada instance container yang kosong. Grup Auto Scaling kemudian akan dapat menghentikan instance. EC2 Instance kontainer kemudian dideregistrasi dari cluster.

Kontrol instans Amazon ECS berakhir

> **Important**
>
> Anda harus mengaktifkan perlindungan penskalaan instans Auto Scaling pada grup Auto Scaling untuk menggunakan fitur proteksi terminasi terkelola dari penskalaan otomatis cluster.

Perlindungan terminasi terkelola memungkinkan penskalaan otomatis cluster untuk mengontrol instance mana yang dihentikan. Saat Anda menggunakan perlindungan penghentian terkelola, Amazon ECS hanya menghentikan EC2 instans yang tidak menjalankan tugas Amazon ECS. Tugas yang dijalankan oleh layanan yang menggunakan strategi `DAEMON` penjadwalan diabaikan dan sebuah instance dapat dihentikan oleh penskalaan otomatis cluster bahkan ketika instance menjalankan tugas-tugas ini. Ini karena semua instance di cluster menjalankan tugas-tugas ini.

Amazon ECS pertama-tama mengaktifkan opsi perlindungan skala masuk instans untuk EC2 instans di grup Auto Scaling. Kemudian, Amazon ECS menempatkan tugas pada instans. Ketika semua

tugas non-daemon dihentikan pada sebuah instance, Amazon ECS memulai proses scale-in dan mematikan perlindungan scale-in untuk instance tersebut. EC2 Grup Auto Scaling kemudian dapat menghentikan instance.

Auto Scaling instance scale-in protection mengontrol EC2 instance mana yang dapat dihentikan oleh Auto Scaling. Instans dengan fitur scale-in diaktifkan tidak dapat dihentikan selama proses scale-in. Untuk informasi selengkapnya tentang perlindungan penskalaan instans Auto Scaling, lihat Menggunakan perlindungan penskalaan instans di Panduan Pengguna Penskalaan Otomatis EC2 Amazon.

Anda dapat mengatur `targetCapacity` persentase sehingga Anda memiliki kapasitas cadangan. Ini membantu tugas future diluncurkan lebih cepat karena grup Auto Scaling tidak harus meluncurkan lebih banyak instance. Amazon ECS menggunakan nilai kapasitas target untuk mengelola CloudWatch metrik yang dibuat layanan. Amazon ECS mengelola CloudWatch metrik. Grup Auto Scaling diperlakukan sebagai kondisi tunak sehingga tidak diperlukan tindakan penskalaan. Nilainya bisa dari 0-100%. Misalnya, untuk mengonfigurasi Amazon ECS agar kapasitas bebas 10% di atas yang digunakan oleh tugas Amazon ECS, tetapkan nilai kapasitas target menjadi 90%. Pertimbangkan hal berikut saat menetapkan `targetCapacity` nilai pada penyedia kapasitas.

- `targetCapacity`Nilai kurang dari 100% mewakili jumlah kapasitas gratis (EC2 instans Amazon) yang perlu ada di cluster. Kapasitas bebas berarti tidak ada tugas yang berjalan.
- Batasan penempatan seperti Availability Zones, tanpa tambahan memaksa `binpack` Amazon ECS untuk akhirnya menjalankan satu tugas untuk setiap instance, yang mungkin bukan perilaku yang diinginkan.

Anda harus mengaktifkan perlindungan penskalaan instans Auto Scaling pada grup Auto Scaling untuk menggunakan perlindungan terminasi terkelola. Jika Anda tidak mengaktifkan perlindungan skala dalam, maka mengaktifkan perlindungan terminasi terkelola dapat menyebabkan perilaku yang tidak diinginkan. Misalnya, Anda mungkin memiliki instance yang terjebak dalam keadaan menguras. Untuk informasi selengkapnya, lihat Menggunakan perlindungan penskalaan instans di Panduan Pengguna Amazon Auto EC2 Scaling.

Saat Anda menggunakan perlindungan penghentian dengan penyedia kapasitas, jangan lakukan tindakan manual apa pun, seperti melepaskan instance, pada grup Auto Scaling yang terkait dengan penyedia kapasitas. Tindakan manual dapat mematahkan operasi skala dalam penyedia kapasitas. Jika Anda melepaskan instance dari grup Auto Scaling, Anda juga perlu membatalkan pendaftaran instans terpisah dari cluster Amazon ECS.

Memperbarui perlindungan penghentian terkelola untuk penyedia kapasitas Amazon ECS

Bila Anda menggunakan proteksi terminasi terkelola, Anda perlu memperbarui pengaturan untuk penyedia kapasitas yang ada.

Konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman cluster, pilih tab Infrastructure.
4. Pilih penyedia kapasitas.
5. Pilih Perbarui untuk mengubah pengaturan penyedia kapasitas.
6. Di bawah pengaturan grup Auto Scaling, alihkan perlindungan terminasi terkelola untuk mengaktifkan atau menonaktifkan fitur.
7. Pilih Perbarui.

AWS CLI

Anda dapat memperbarui setelan proteksi terminasi terkelola penyedia kapasitas menggunakan `update-capacity-provider` perintah:

Untuk mengaktifkan perlindungan terminasi terkelola:

```
aws ecs update-capacity-provider \
  --capacity-provider CapacityProviderName \
  --auto-scaling-group-provider
 "managedScaling={status=ENABLED,targetCapacity=70,minimumScalingStepSize=1,maximumScalingStepS
```

Untuk menonaktifkan perlindungan terminasi terkelola:

```
aws ecs update-capacity-provider \
  --capacity-provider CapacityProviderName \
  --auto-scaling-group-provider
 "managedScaling={status=ENABLED,targetCapacity=70,minimumScalingStepSize=1,maximumScalingStepS
```

> **Note**
>
> Mungkin perlu beberapa menit agar perubahan diterapkan di seluruh cluster Anda. Saat mengaktifkan perlindungan terminasi terkelola, instance yang sudah menjalankan tugas akan

dilindungi dari peristiwa penskalaan. Saat menonaktifkan perlindungan terminasi terkelola, bendera perlindungan akan dihapus dari instans selama siklus manajemen penyedia kapasitas ECS berikutnya.

Konsol untuk menjalankan tugas

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman cluster, pilih tab Tasks.
4. Pilih tugas.
5. Di bawah Konfigurasi, alihkan perlindungan terminasi terkelola untuk mengaktifkan atau menonaktifkan fitur.
6. Pilih Konfigurasi perlindungan skala tugas.

   Kotak dialog Configure task scale-in protection ditampilkan

   a. Di bawah Perlindungan skala tugas, aktifkan Aktifkan.
   b. Untuk Kedaluwarsa dalam hitungan menit, masukkan jumlah menit sebelum perlindungan penskalaan tugas berakhir.
   c. Pilih Perbarui

Mengaktifkan penskalaan otomatis cluster Amazon ECS

Anda mengaktifkan penskalaan otomatis cluster sehingga Amazon ECS mengelola penskalaan EC2 instans Amazon yang terdaftar ke klaster Anda.

Jika Anda ingin menggunakan konsol untuk mengaktifkan penskalaan otomatis Cluster, lihat lihatMembuat penyedia kapasitas untuk Amazon ECS.

Sebelum memulai, buat grup Auto Scaling dan penyedia kapasitas. Untuk informasi selengkapnya, lihat the section called “Penyedia kapasitas untuk jenis EC2 peluncuran”.

Untuk mengaktifkan penskalaan otomatis cluster, Anda mengaitkan penyedia kapasitas dengan cluster, Kemudian Anda mengaktifkan penskalaan otomatis cluster.

1. Gunakan `put-cluster-capacity-providers` perintah untuk mengaitkan satu atau lebih penyedia kapasitas dengan cluster.

Untuk menambah penyedia AWS Fargate kapasitas, sertakan FARGATE dan penyedia FARGATE_SPOT kapasitas dalam permintaan. Untuk informasi selengkapnya, lihat put-cluster-capacity-providers dalam AWS CLI Referensi Perintah.

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers CapacityProviderName FARGATE FARGATE_SPOT \
  --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
```

Untuk menambahkan grup Auto Scaling untuk jenis EC2 peluncuran, sertakan nama grup Auto Scaling dalam permintaan. Untuk informasi selengkapnya, lihat put-cluster-capacity-providers dalam AWS CLI Referensi Perintah.

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers CapacityProviderName \
  --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
```

2. Gunakan describe-clusters perintah untuk memverifikasi bahwa asosiasi berhasil. Untuk informasi selengkapnya, lihat describe-clusters dalam AWS CLI Referensi Perintah.

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

3. Gunakan update-capacity-provider perintah untuk mengaktifkan penskalaan otomatis terkelola untuk penyedia kapasitas. Untuk informasi selengkapnya, lihat update-capacity-provider dalam AWS CLI Referensi Perintah.

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=ENABLED
```

Mematikan penskalaan otomatis cluster Amazon ECS

Anda menonaktifkan penskalaan otomatis cluster saat Anda membutuhkan kontrol yang lebih terperinci dari EC2 instance yang terdaftar ke klaster Anda,

Untuk menonaktifkan penskalaan otomatis klaster untuk kluster, Anda dapat memisahkan penyedia kapasitas dengan penskalaan terkelola yang diaktifkan dari klaster atau memperbarui penyedia kapasitas untuk menonaktifkan penskalaan terkelola.

Putuskan hubungan penyedia kapasitas

Gunakan langkah-langkah berikut untuk memisahkan penyedia kapasitas dengan cluster.

1. Gunakan `put-cluster-capacity-providers` perintah untuk memisahkan penyedia kapasitas grup Auto Scaling dengan cluster. Cluster dapat menjaga hubungan dengan penyedia AWS Fargate kapasitas. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy '[]'
   ```

   Gunakan `put-cluster-capacity-providers` perintah untuk memisahkan penyedia kapasitas grup Auto Scaling dengan cluster. Untuk informasi selengkapnya, lihat `put-cluster-capacity-providers` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers [] \
     --default-capacity-provider-strategy '[]'
   ```

2. Gunakan `describe-clusters` perintah untuk memverifikasi bahwa disasosiasi berhasil. Untuk informasi selengkapnya, lihat `describe-clusters` dalam AWS CLI Referensi Perintah.

   ```
   aws ecs describe-clusters \
     --cluster ClusterName \
     --include ATTACHMENTS
   ```

Matikan penskalaan terkelola untuk penyedia kapasitas

Gunakan langkah-langkah berikut untuk menonaktifkan penskalaan terkelola untuk penyedia kapasitas.

- Gunakan `update-capacity-provider` perintah untuk mematikan penskalaan otomatis terkelola untuk penyedia kapasitas. Untuk informasi selengkapnya, lihat `update-capacity-provider` dalam AWS CLI Referensi Perintah.

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=DISABLED
```

Membuat penyedia kapasitas untuk Amazon ECS

Setelah pembuatan cluster selesai, Anda dapat membuat penyedia kapasitas baru (grup Auto Scaling) untuk jenis peluncuran EC2 . Penyedia kapasitas membantu mengelola dan menskalakan infrastruktur untuk aplikasi Anda.

Sebelum Anda membuat penyedia kapasitas, Anda perlu membuat grup Auto Scaling. Untuk informasi selengkapnya, lihat grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.

Untuk membuat penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada *name* halaman Cluster:, pilih Infrastruktur, lalu pilih Buat.
5. Pada halaman Buat penyedia kapasitas, konfigurasikan opsi berikut.

    a. Di bawah Detail dasar, untuk nama penyedia Kapasitas, masukkan nama penyedia kapasitas unik.

    b. Di bawah grup Auto Scaling, untuk grup Menggunakan Auto Scaling yang ada, pilih grup Auto Scaling.

    c. (Opsional) Untuk mengonfigurasi kebijakan penskalaan, di bawah Kebijakan penskalaan, konfigurasikan opsi berikut.

    - Agar Amazon ECS mengelola tindakan scale-in dan scale-out, pilih Aktifkan penskalaan terkelola.
    - Untuk mencegah EC2 instance dengan menjalankan tugas Amazon ECS agar tidak dihentikan, pilih Aktifkan perlindungan penskalaan.

- Untuk Menetapkan kapasitas target, masukkan nilai target untuk CloudWatch metrik yang digunakan dalam kebijakan penskalaan pelacakan target yang dikelola Amazon ECS.

6. Pilih Buat.

Memperbarui penyedia kapasitas Amazon ECS

Bila Anda menggunakan grup Auto Scaling sebagai penyedia kapasitas, Anda dapat mengubah kebijakan penskalaan grup.

Untuk memperbarui penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada *name* halaman Cluster:, pilih Infrastruktur, lalu pilih Perbarui.
5. Pada halaman Buat penyedia kapasitas, konfigurasikan opsi berikut.
   - Di bawah grup Auto Scaling, di bawah kebijakan Penskalaan, konfigurasikan opsi berikut.
     - Agar Amazon ECS mengelola tindakan scale-in dan scale-out, pilih Aktifkan penskalaan terkelola.
     - Untuk mencegah EC2 instans yang menjalankan tugas Amazon ECS dihentikan, pilih Aktifkan perlindungan penskalaan.
     - Untuk Menetapkan kapasitas target, masukkan nilai target untuk CloudWatch metrik yang digunakan dalam kebijakan penskalaan pelacakan target yang dikelola Amazon ECS.
6. Pilih Perbarui.

Menghapus penyedia kapasitas Amazon ECS

Jika Anda selesai menggunakan penyedia kapasitas grup Auto Scaling, Anda dapat menghapusnya. Setelah grup dihapus, penyedia kapasitas grup Auto Scaling beralih ke status. `INACTIVE` Penyedia kapasitas dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Namun, perilaku ini dapat berubah di masa depan, jadi sebaiknya Anda tidak mengandalkan penyedia kapasitas `INACTIVE` yang tersimpan. Sebelum penyedia kapasitas grup Auto Scaling dihapus, penyedia kapasitas harus dihapus dari strategi penyedia kapasitas dari semua layanan. Anda dapat menggunakan `UpdateService` API atau alur kerja layanan pembaruan

di konsol Amazon ECS untuk menghapus penyedia kapasitas dari strategi penyedia kapasitas layanan. Gunakan opsi Paksa penerapan baru untuk memastikan bahwa tugas apa pun yang menggunakan kapasitas EC2 instans Amazon yang disediakan oleh penyedia kapasitas dialihkan untuk menggunakan kapasitas dari penyedia kapasitas yang tersisa.

Untuk menghapus penyedia kapasitas untuk cluster (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada ***name*** halaman Cluster:, pilih Infrastruktur, grup Auto Scaling, lalu pilih Hapus.
5. Di kotak konfirmasi, masukkan hapus ***Auto Scaling group name***
6. Pilih Hapus.

Hentikan beban kerja Amazon ECS dengan aman yang berjalan pada instans EC2

Pengurasan instans terkelola memfasilitasi penghentian instans Amazon EC2 yang anggun. Hal ini memungkinkan beban kerja Anda berhenti dengan aman dan dijadwal ulang ke instans non-terminating. Pemeliharaan dan pembaruan infrastruktur dilakukan tanpa khawatir akan gangguan pada beban kerja. Dengan menggunakan pengeringan instans terkelola, Anda menyederhanakan alur kerja manajemen infrastruktur yang memerlukan penggantian EC2 instans Amazon sambil memastikan ketahanan dan ketersediaan aplikasi Anda.

Pengurasan instans terkelola Amazon ECS berfungsi dengan penggantian instans grup Auto Scaling. Berdasarkan penyegaran instans dan masa pakai instans maksimum, pelanggan dapat memastikan bahwa mereka tetap mematuhi OS terbaru dan mandat keamanan untuk kapasitas mereka.

Pengurasan instans terkelola hanya dapat digunakan dengan penyedia kapasitas Amazon ECS. Anda dapat mengaktifkan pengurasan instans terkelola saat membuat atau memperbarui penyedia kapasitas grup Auto Scaling menggunakan konsol Amazon ECS AWS CLI, atau SDK.

Peristiwa berikut dicakup oleh pengeringan instans terkelola Amazon ECS.

- Penyegaran instans grup Auto Scaling - Gunakan penyegaran instans untuk melakukan penggantian bergulir EC2 instans Amazon Anda di grup Auto Scaling alih-alih melakukannya secara manual dalam batch. Ini berguna ketika Anda perlu mengganti sejumlah besar instance. Penyegaran instance dimulai melalui EC2 konsol Amazon atau `StartInstanceRefresh`

API. Pastikan Anda memilih `Replace` perlindungan Scale-in saat menelepon `StartInstanceRefresh` jika Anda menggunakan proteksi terminasi terkelola.

- Masa pakai instans maksimum - Anda dapat menentukan masa pakai maksimum saat mengganti instance grup Auto Scaling. Hal ini berguna untuk menjadwalkan instance penggantian berdasarkan kebijakan keamanan internal atau kepatuhan.
- Penskalaan grup Auto Scaling - Berdasarkan kebijakan penskalaan dan tindakan penskalaan terjadwal, grup Auto Scaling mendukung penskalaan otomatis instans. Dengan menggunakan grup Auto Scaling sebagai penyedia kapasitas Amazon ECS, Anda dapat menskalakan instans grup Auto Scaling saat tidak ada tugas yang berjalan di dalamnya.
- Pemeriksaan kesehatan grup Auto Scaling - Grup Auto Scaling mendukung banyak pemeriksaan kesehatan untuk mengelola penghentian kejadian yang tidak sehat.
- AWS CloudFormation pembaruan tumpukan - Anda dapat menambahkan `UpdatePolicy` atribut ke AWS CloudFormation tumpukan Anda untuk melakukan pembaruan bergulir saat grup berubah.
- Penyeimbangan kembali kapasitas spot - Grup Auto Scaling mencoba mengganti Instans Spot secara proaktif yang memiliki risiko gangguan lebih tinggi berdasarkan pemberitahuan penyeimbangan kapasitas Amazon. EC2 Grup Auto Scaling menghentikan instance lama saat penggantian diluncurkan dan sehat. Pengurasan instans terkelola Amazon ECS menguras Instans Spot dengan cara yang sama seperti menguras Instans Non-Spot.
- Gangguan spot - Instans Spot diakhiri dengan pemberitahuan dua menit. Pengurasan instans yang dikelola Amazon ECS menempatkan instance dalam status menguras sebagai respons.

Kait siklus hidup EC2 Auto Scaling Amazon dengan pengurasan instans terkelola

Kait siklus hidup grup Auto Scaling memungkinkan pelanggan membuat solusi yang dipicu oleh peristiwa tertentu dalam siklus hidup instance dan melakukan tindakan kustom saat peristiwa tertentu terjadi. Grup Auto Scaling memungkinkan hingga 50 kait. Beberapa kait terminasi dapat ada dan dilakukan secara paralel, dan grup Auto Scaling menunggu semua kait selesai sebelum mengakhiri sebuah instance.

Selain penghentian kait yang dikelola Amazon ECS, Anda juga dapat mengonfigurasi kait penghentian siklus hidup Anda sendiri. Kait siklus hidup memilikidefault action, dan sebaiknya `continue` setelan sebagai default untuk memastikan kait lain, seperti hook terkelola Amazon ECS, tidak terpengaruh oleh kesalahan apa pun dari kait khusus.

Jika Anda telah mengonfigurasi hook siklus hidup penghentian grup Auto Scaling dan juga mengaktifkan pengurasan instans terkelola Amazon ECS, kedua kait siklus hidup akan dilakukan.

Pengaturan waktu relatif, bagaimanapun, tidak dijamin. Kait siklus hidup memiliki `default action` pengaturan untuk menentukan tindakan yang akan diambil saat batas waktu berlalu. Jika terjadi kegagalan, kami sarankan untuk menggunakan `continue` sebagai hasil default di kait khusus Anda. Ini memastikan kait lain, terutama kait terkelola Amazon ECS, tidak terpengaruh oleh kesalahan apa pun di hook siklus hidup kustom Anda. Hasil alternatif `abandon` menyebabkan semua kait lainnya dilewati dan harus dihindari. Untuk informasi selengkapnya tentang kait siklus hidup grup Auto Scaling, lihat kait siklus hidup Auto Scaling Amazon di EC2 Panduan Pengguna Penskalaan Otomatis Amazon. EC2

Tugas dan pengeringan instans terkelola

Pengurasan instans terkelola Amazon ECS menggunakan fitur pengurasan yang ada yang ditemukan dalam instance kontainer. Fitur pengurasan instans kontainer melakukan penggantian dan penghentian untuk tugas replika yang termasuk dalam layanan Amazon ECS. Tugas mandiri, seperti yang dipanggil oleh`RunTask`, yang berada dalam `RUNNING` status `PENDING` atau tetap tidak terpengaruh. Anda harus menunggu ini untuk menyelesaikan atau menghentikannya secara manual. Instance kontainer tetap dalam `DRAINING` status sampai semua tugas dihentikan atau 48 jam telah berlalu. Tugas daemon adalah yang terakhir berhenti setelah semua tugas replika berhenti.

Pengurasan instans terkelola dan perlindungan terminasi terkelola

Pengurasan instans terkelola berfungsi meskipun penghentian terkelola dinonaktifkan. Untuk informasi tentang perlindungan penghentian terkelola, lihatKontrol instans Amazon ECS berakhir.

Tabel berikut merangkum perilaku untuk kombinasi yang berbeda dari terminasi terkelola dan pengeringan terkelola.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Diaktifkan	Diaktifkan	Amazon ECS melindungi EC2 instans Amazon yang menjalankan tugas agar tidak

Pengakhiran terkelola	Pengeringan terkelola	Hasil
		dihentikan oleh peristiwa penskalaan. Setiap instance yang mengalami penghentian, seperti yang tidak memiliki perlindungan terminasi yang ditetapkan, telah menerima interupsi Spot, atau dipaksa oleh penyegaran instans akan terkuras dengan baik.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Nonaktif	Diaktifkan	Amazon ECS tidak melindungi EC2 instans Amazon yang menjalankan tugas agar tidak diskalakan. Namun, setiap contoh yang sedang dihentikan akan dikeringkan dengan baik.

Pengakhiran terkelola	Pengeringan terkelola	Hasil
Diaktifkan	Dinonaktifkan	Amazon ECS melindungi EC2 instans Amazon yang menjalankan tugas agar tidak dihentikan oleh peristiwa penskalaan. Namun, instance masih dapat dihentikan oleh gangguan Spot atau penyegaran instans paksa, atau jika mereka tidak menjalankan tugas apa pun. Amazon ECS tidak melakukan

Pengakhiran terkelola	Pengeringan terkelola	Hasil
		pengeringan yang baik untuk instans ini, dan meluncurkan tugas layanan pengganti setelah berhenti.
Nonaktif	Nonaktif	EC2 Instans Amazon dapat diskalakan atau dihentikan kapan saja, bahkan jika mereka menjalankan tugas Amazon ECS. Amazon ECS akan meluncurkan tugas layanan pengganti setelah berhenti.

Pengurasan instans terkelola dan pengeringan Instans Spot

Dengan pengeringan Instans Spot, Anda dapat menyetel variabel lingkungan `ECS_ENABLE_SPOT_INSTANCE_DRAINING` di agen Amazon ECS yang memungkinkan Amazon ECS menempatkan instans dalam status pengurasan sebagai respons terhadap gangguan Spot dua menit. Pengurasan instans terkelola Amazon ECS memfasilitasi penutupan instans EC2 Amazon yang mengalami penghentian secara anggun karena berbagai alasan, bukan hanya gangguan Spot. Misalnya, Anda dapat menggunakan penyeimbangan ulang kapasitas EC2 Auto Scaling Amazon untuk secara proaktif mengganti Instans Spot dengan risiko interupsi yang tinggi, dan pengurasan instans terkelola melakukan shutdown yang anggun pada Instans Spot yang diganti. Saat Anda menggunakan pengurasan instans terkelola, Anda tidak perlu mengaktifkan pengurasan instans Spot secara terpisah, sehingga `ECS_ENABLE_SPOT_INSTANCE_DRAINING` data pengguna grup Auto Scaling berlebihan. Untuk informasi selengkapnya tentang pengeringan Instans Spot, lihatInstans Spot.

Cara kerja pengeringan instans terkelola EventBridge

Acara pengeringan instans terkelola Amazon ECS dipublikasikan ke Amazon EventBridge, dan Amazon ECS membuat aturan EventBridge terkelola di bus default akun Anda untuk mendukung pengeringan instans terkelola. Anda dapat memfilter peristiwa ini ke AWS layanan lain seperti Lambda, Amazon SNS, dan Amazon SQS untuk memantau dan memecahkan masalah.

- Amazon EC2 Auto Scaling mengirimkan peristiwa ke EventBridge saat hook siklus hidup dipanggil.
- Pemberitahuan gangguan spot dipublikasikan ke. EventBridge
- Amazon ECS menghasilkan pesan kesalahan yang dapat Anda ambil melalui konsol Amazon ECS dan. APIs
- EventBridge memiliki mekanisme coba lagi yang dibangun sebagai mitigasi untuk kegagalan sementara.

Mengonfigurasi penyedia kapasitas Amazon ECS untuk mematikan instans dengan aman

Anda dapat mengaktifkan pengurasan instans terkelola saat membuat atau memperbarui penyedia kapasitas grup Auto Scaling menggunakan konsol Amazon ECS dan. AWS CLI

> **Note**
>
> Pengurasan instans terkelola aktif secara default saat Anda membuat penyedia kapasitas.

Berikut ini adalah contoh penggunaan AWS CLI untuk membuat penyedia kapasitas dengan pengeringan instans terkelola diaktifkan dan memungkinkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada.

Buat penyedia kapasitas dengan pengurasan instans terkelola diaktifkan

Untuk membuat penyedia kapasitas dengan pengurasan instans terkelola diaktifkan, gunakan `create-capacity-provider` perintah. Atur parameter `managedDraining` ke `ENABLED`.

```
aws ecs create-capacity-provider \
--name capacity-provider \
--auto-scaling-group-provider '{
  "autoScalingGroupArn": "asg-arn",
  "managedScaling": {
    "status": "ENABLED",
    "targetCapacity": 100,
    "minimumScalingStepSize": 1,
    "maximumScalingStepSize": 1
  },
  "managedDraining": "ENABLED",
  "managedTerminationProtection": "ENABLED",
}'
```

Respons:

```
{
    "capacityProvider": {
        "capacityProviderArn": "capacity-provider-arn",
        "name": "capacity-provider",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1
            },
            "managedTerminationProtection": "ENABLED"
            "managedDraining": "ENABLED"
        }
    }
```

```
}
```

Aktifkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada

Aktifkan pengeringan instans terkelola untuk penyedia kapasitas kluster yang ada menggunakan `update-capacity-provider` perintah. Anda melihat bahwa `managedDraining` saat ini mengatakan `DISABLED` dan `updateStatus` berkata`UPDATE_IN_PROGRESS`.

```
aws ecs update-capacity-provider \
--name cp-draining \
--auto-scaling-group-provider '{
  "managedDraining": "ENABLED"
}
```

Respons:

```
{
    "capacityProvider": {
        "capacityProviderArn": "cp-draining-arn",
        "name": "cp-draining",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-draining-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1,
                "instanceWarmupPeriod": 300
            },
            "managedTerminationProtection": "DISABLED",
            "managedDraining": "DISABLED" // before update
        },
        "updateStatus": "UPDATE_IN_PROGRESS", // in progress and need describe again to
 find out the result
        "tags": [
        ]
    }
}
```

Gunakan `describe-clusters` perintah dan sertakan`ATTACHMENTS`. `status`Lampiran pengeringan instance terkelola adalah`PRECREATED`, dan keseluruhannya `attachmentsStatus` adalah`UPDATING`.

```
aws ecs describe-clusters --clusters cluster-name --include ATTACHMENTS
```

Respons:

```
{
    "clusters": [
        {
            ...

            "capacityProviders": [
                "cp-draining"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                # new precreated managed draining attachment
                {
                    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "type": "managed_draining",
                    "status": "PRECREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "cp-draining"
                        },
                        {
                            "name": "autoScalingLifecycleHookName",
                            "value": "ecs-managed-draining-termination-hook"
                        }
                    ]
                },

                ...

            ],
            "attachmentsStatus": "UPDATING"
        }
    ],
    "failures": []
```

```
}
```

Ketika pembaruan selesai, gunakan`describe-capacity-providers`, dan Anda lihat `managedDraining` sekarang`ENABLED`.

```
aws ecs describe-capacity-providers --capacity-providers cp-draining
```

Respons:

```
{
    "capacityProviders": [
        {
            "capacityProviderArn": "cp-draining-arn",
            "name": "cp-draining",
            "status": "ACTIVE",
            "autoScalingGroupProvider": {
                "autoScalingGroupArn": "asg-draning-arn",
                "managedScaling": {
                    "status": "ENABLED",
                    "targetCapacity": 100,
                    "minimumScalingStepSize": 1,
                    "maximumScalingStepSize": 1,
                    "instanceWarmupPeriod": 300
                },
                "managedTerminationProtection": "DISABLED",
                "managedDraining": "ENABLED" // successfully update
            },
            "updateStatus": "UPDATE_COMPLETE",
            "tags": []
        }
    ]
}
```

Pemecahan masalah pengurasan instans Amazon ECS Terkelola

Anda mungkin perlu memecahkan masalah dengan pengeringan instans terkelola. Berikut ini adalah contoh masalah dan resolusi yang mungkin Anda temui saat menggunakannya.

Instans tidak berakhir setelah melebihi masa pakai instans maksimum saat menggunakan penskalaan otomatis.

Jika instans Anda tidak berakhir bahkan setelah mencapai dan melebihi masa pakai instans maksimum saat menggunakan grup penskalaan otomatis, itu mungkin karena mereka dilindungi dari penskalaan. Anda dapat mematikan penghentian terkelola dan mengizinkan pengeringan terkelola untuk menangani daur ulang instans.

Membuat sumber daya untuk penskalaan otomatis cluster Amazon ECS menggunakan AWS Management Console

Pelajari cara membuat sumber daya untuk penskalaan otomatis cluster menggunakan. AWS Management Console Di mana sumber daya memerlukan nama, kami menggunakan awalan `ConsoleTutorial` untuk memastikan mereka semua memiliki nama unik dan membuatnya mudah ditemukan.

Topik

- Prasyarat
- Langkah 1: Buat cluster Amazon ECS
- Langkah 2: Daftarkan definisi tugas
- Langkah 3: Jalankan tugas
- Langkah 4: Verifikasi
- Langkah 5: Bersihkan

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Peran IAM instans penampung Amazon ECS dibuat. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.
- Peran IAM terkait layanan Amazon ECS dibuat. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Peran IAM terkait layanan Auto Scaling dibuat. Untuk informasi selengkapnya, lihat Peran Tertaut Layanan untuk Auto EC2 Scaling Amazon di Panduan Pengguna Amazon Auto EC2 Scaling.
- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Langkah 1: Buat cluster Amazon ECS

Gunakan langkah-langkah berikut untuk membuat cluster Amazon ECS.

Amazon ECS membuat template peluncuran Amazon EC2 Auto Scaling dan grup Auto Scaling atas nama Anda sebagai bagian dari tumpukan. AWS CloudFormation

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih Create cluster.
3. Di bawah konfigurasi Cluster, untuk nama Cluster, masukkan`ConsoleTutorial-cluster`.
4. Di bawah Infrastruktur, hapus AWS Fargate (tanpa server), lalu pilih instans Amazon. EC2 Selanjutnya, konfigurasikan grup Auto Scaling yang bertindak sebagai penyedia kapasitas.
   - Di bawah grup Auto Scaling (ASG). Pilih Buat ASG baru, lalu berikan detail berikut tentang grup:
     - Untuk Sistem Operasi/Arsitektur, pilih Amazon Linux 2.
     - EC2 Misalnya jenis, pilih t3.nano.
     - Untuk Kapasitas, masukkan jumlah minimum dan jumlah maksimum instans yang akan diluncurkan di grup Auto Scaling.
5. (Opsional) Untuk mengelola tag cluster, memperluas Tag, dan kemudian melakukan salah satu operasi berikut:

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tag] Pilih Hapus di sebelah kanan Kunci dan Nilai tag.
6. Pilih Buat.

Langkah 2: Daftarkan definisi tugas

Sebelum Anda dapat menjalankan tugas di klaster Anda, Anda harus mendaftarkan ketentuan tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah ketentuan tugas sederhana yang menggunakan citra `amazonlinux` dari Docker Hub dan hanya

tidur. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, tempel konten berikut.

```
{
    "family": "ConsoleTutorial-taskdef",
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
            "memory": 20,
            "essential": true,
            "command": [
                "sh",
                "-c",
                "sleep infinity"
            ]
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
```

5. Pilih Buat.

Langkah 3: Jalankan tugas

Setelah Anda mendaftarkan ketentuan tugas dari akun Anda, Anda dapat menjalankan tugas-tugas di klaster Anda. Untuk tutorial ini, Anda menjalankan lima instans dari ketentuan tugas `ConsoleTutorial-taskdef` di perangkat klaster `ConsoleTutorial-cluster` Anda.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih ConsoleTutorial-cluster.
3. Di bawah Tugas, pilih Jalankan tugas baru.
4. Di bagian Lingkungan, di bawah Opsi komputasi, pilih Strategi penyedia kapasitas.

5. Di bawah konfigurasi Deployment, untuk tipe Aplikasi, pilih Tugas.
6. Pilih ConsoleTutorial-taskdef dari daftar dropdown Keluarga.
7. Di bawah Tugas yang diinginkan, masukkan 5.
8. Pilih Buat.

Langkah 4: Verifikasi

Pada titik ini dalam tutorial, Anda harus memiliki cluster dengan lima tugas berjalan dan grup Auto Scaling dengan penyedia kapasitas. Penyedia kapasitas mengaktifkan penskalaan terkelola Amazon ECS.

Kami dapat memverifikasi bahwa semuanya berfungsi dengan baik dengan melihat CloudWatch metrik, pengaturan grup Auto Scaling, dan terakhir jumlah tugas cluster Amazon ECS.

Untuk melihat CloudWatch metrik untuk klaster Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bilah navigasi di bagian atas layar, pilih Wilayah .
3. Pada panel navigasi, di bawah Metrik, pilih Semua metrik.
4. Pada halaman Semua metrik, di bawah tab Browse, pilih`AWS/ECS/ManagedScaling`.
5. Pilih CapacityProviderName, ClusterName.
6. Pilih kotak centang yang sesuai dengan `ConsoleTutorial-cluster` ClusterName.
7. Di bawah tab Metrik grafik, ubah Periode menjadi 30 detik dan Statistik ke Maksimum.

   Nilai yang ditampilkan dalam grafik menunjukkan nilai kapasitas target untuk penyedia kapasitas. Ini harus dimulai pada `100`, yang merupakan kapasitas target persen yang kami atur. Anda harus melihatnya menaikkan skala hingga `200`, yang akan memicu alarm kebijakan penskalaan pelacakan target. Alarm kemudian akan memicu grup Auto Scaling untuk skala keluar.

Gunakan langkah-langkah berikut untuk melihat detail grup Auto Scaling untuk mengonfirmasi bahwa tindakan penskalaan telah terjadi.

Untuk memverifikasi grup Auto Scaling yang diperkecil

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Pada bilah navigasi di bagian atas layar, pilih Wilayah .

3. Pada panel navigasi, di bawah Auto Scaling, pilih Grup Auto Scaling.
4. Pilih grup `ConsoleTutorial-cluster` Auto Scaling yang dibuat dalam tutorial ini. Lihat nilai di bawah Kapasitas yang diinginkan dan lihat instance di bawah tab Manajemen instans untuk mengonfirmasi grup Anda diskalakan menjadi dua instance.

Gunakan langkah-langkah berikut untuk melihat kluster Amazon ECS untuk mengonfirmasi bahwa EC2 instans Amazon telah terdaftar di klaster dan tugas Anda dialihkan ke status. `RUNNING`

Untuk memverifikasi instance dalam grup Auto Scaling

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih `ConsoleTutorial-cluster` cluster.
4. Pada tab Tugas, konfirmasikan bahwa Anda melihat lima tugas dalam `RUNNING` status.

Langkah 5: Bersihkan

Setelah Anda menyelesaikan tutorial ini, bersihkan sumber daya yang terkait dengannya untuk menghindari biaya yang tidak Anda gunakan. Menghapus penyedia kapasitas dan ketentuan tugas tidak didukung, tetapi tidak ada biaya yang terkait dengan sumber daya ini.

Untuk membersihkan sumber daya tutorial

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih ConsoleTutorial-cluster.
4. Pada halaman ConsoleTutorial-cluster, pilih tab Tasks, lalu pilih Stop, Stop all.
5. Pada panel navigasi, silakan pilih Klaster.
6. Pada halaman Clusters, pilih ConsoleTutorial-cluster.
7. Di kanan atas halaman, pilih Hapus klaster.
8. Di kotak konfirmasi, masukkan delete ConsoleTutorial-cluster dan pilih Delete.
9. Hapus grup Auto Scaling menggunakan langkah-langkah berikut.
    a. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
    b. Pada bilah navigasi di bagian atas layar, pilih Wilayah .

c. Pada panel navigasi, di bawah Auto Scaling, pilih Grup Auto Scaling.

d. Pilih grup **`ConsoleTutorial-cluster`** Auto Scaling, lalu pilih Tindakan.

e. Dari menu Tindakan, pilih Hapus. Masukkan hapus di kotak konfirmasi dan kemudian pilih Hapus.

Instans EC2 kontainer Amazon untuk Amazon ECS

Instans penampung Amazon ECS adalah EC2 instans Amazon yang menjalankan agen penampung Amazon ECS dan terdaftar ke klaster. Saat Anda menjalankan tugas dengan Amazon ECS menggunakan jenis EC2 peluncuran, Jenis peluncuran eksternal, atau penyedia kapasitas grup Auto Scaling, tugas Anda ditempatkan pada instance container aktif Anda. Anda bertanggung jawab atas manajemen dan pemeliharaan instans kontainer.

Meskipun Anda dapat membuat AMI EC2 instans Amazon sendiri yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, Amazon ECS yang dioptimalkan AMIs telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur. AWS Ini adalah cara termudah bagi Anda untuk memulai dan mendapatkan kontainer Anda yang berjalan pada AWS dengan cepat.

Saat Anda membuat klaster menggunakan konsol, Amazon ECS membuat template peluncuran untuk instans Anda dengan AMI terbaru yang terkait dengan sistem operasi yang dipilih.

Saat Anda menggunakan AWS CloudFormation untuk membuat klaster, parameter SSM adalah bagian dari template EC2 peluncuran Amazon untuk instance grup Auto Scaling. Anda dapat mengonfigurasi template untuk menggunakan parameter Systems Manager dinamis untuk menentukan apa yang akan digunakan oleh Amazon ECS Optimized AMI. Parameter ini memastikan bahwa setiap kali Anda menerapkan tumpukan itu akan memeriksa untuk melihat apakah ada pembaruan yang tersedia yang perlu diterapkan ke EC2 instance. Untuk contoh cara menggunakan parameter Systems Manager, lihat Membuat klaster Amazon ECS dengan Amazon ECS AMI Amazon Linux 2023 yang dioptimalkan Amazon ECS di AWS CloudFormation Panduan Pengguna.

- Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan
- Mengambil metadata AMI Amazon ECS yang dioptimalkan Bottlerocket
- Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS

Anda dapat memilih dari jenis instance yang kompatibel dengan aplikasi Anda. Dengan instance yang lebih besar, Anda dapat meluncurkan lebih banyak tugas secara bersamaan. Dengan contoh

yang lebih kecil, Anda dapat meningkatkan skala dengan cara yang lebih halus untuk menghemat biaya. Anda tidak perlu memilih satu jenis EC2 instans Amazon yang sesuai dengan semua aplikasi di cluster Anda. Sebagai gantinya, Anda dapat membuat beberapa grup Auto Scaling di mana setiap grup memiliki jenis instans yang berbeda. Kemudian, Anda dapat membuat penyedia EC2 kapasitas Amazon untuk masing-masing grup ini.

Gunakan pedoman berikut untuk menentukan tipe keluarga instans dan tipe instans yang akan digunakan:

- Hilangkan jenis instans atau keluarga instans yang tidak memenuhi persyaratan spesifik aplikasi Anda. Misalnya, jika aplikasi Anda memerlukan GPU, Anda dapat mengecualikan jenis instance apa pun yang tidak memiliki GPU.
- Pertimbangkan persyaratan termasuk throughput jaringan dan penyimpanan.
- Pertimbangkan CPU dan memori. Sebagai aturan umum, CPU dan memori harus cukup besar untuk menampung setidaknya satu replika tugas yang ingin Anda jalankan.

Instans Spot

Kapasitas spot dapat memberikan penghematan biaya yang signifikan dibandingkan instans sesuai permintaan. Kapasitas spot adalah kelebihan kapasitas yang harganya jauh lebih rendah daripada kapasitas sesuai permintaan atau cadangan. Kapasitas spot cocok untuk pemrosesan batch dan beban kerja pembelajaran mesin, serta lingkungan pengembangan dan pementasan. Secara lebih umum, ini cocok untuk semua beban kerja yang mentolerir downtime sementara.

Pahami bahwa konsekuensi berikut karena kapasitas Spot mungkin tidak tersedia sepanjang waktu.

- Selama periode permintaan yang sangat tinggi, kapasitas Spot mungkin tidak tersedia. Hal ini dapat menyebabkan peluncuran instans Amazon EC2 Spot tertunda. Dalam acara ini, layanan Amazon ECS mencoba lagi meluncurkan tugas, dan grup Amazon Auto EC2 Scaling juga mencoba meluncurkan instans lagi, hingga kapasitas yang diperlukan tersedia. Amazon EC2 tidak mengganti kapasitas Spot dengan kapasitas sesuai permintaan.
- Ketika permintaan keseluruhan untuk kapasitas meningkat, Instans Spot dan tugas dapat dihentikan hanya dengan peringatan dua menit. Setelah peringatan dikirim, tugas harus memulai shutdown yang teratur jika perlu sebelum instance dihentikan sepenuhnya. Ini membantu meminimalkan kemungkinan kesalahan. Untuk informasi lebih lanjut tentang shutdown yang anggun, lihat Penutupan anggun dengan ECS.

Untuk membantu meminimalkan kekurangan kapasitas Spot, pertimbangkan rekomendasi berikut:

- Gunakan beberapa Wilayah dan Availability Zone - Kapasitas spot bervariasi menurut Wilayah dan Availability Zone. Anda dapat meningkatkan ketersediaan Spot dengan menjalankan beban kerja di beberapa Wilayah dan Availability Zone. Jika memungkinkan, tentukan subnet di semua Availability Zone di Wilayah tempat Anda menjalankan tugas dan instance.
- Gunakan beberapa jenis EC2 instans Amazon - Saat Anda menggunakan Kebijakan Instance Campuran dengan Amazon EC2 Auto Scaling, beberapa jenis instans akan diluncurkan ke Grup Auto Scaling. Ini memastikan bahwa permintaan kapasitas Spot dapat dipenuhi saat dibutuhkan. Untuk memaksimalkan keandalan dan meminimalkan kompleksitas, gunakan tipe instans dengan jumlah CPU dan memori yang kira-kira sama dalam Kebijakan Instans Campuran Anda. Instance ini dapat berasal dari generasi yang berbeda, atau varian dari tipe instance dasar yang sama. Perhatikan bahwa mereka mungkin datang dengan fitur tambahan yang mungkin tidak Anda perlukan. Contoh daftar semacam itu dapat mencakup m4.large, m5.large, m5a.large, m5d.large, m5n.large, m5dn.large, dan m5ad.large. Untuk informasi selengkapnya, lihat grup Auto Scaling dengan beberapa jenis instans dan opsi pembelian di Panduan Pengguna Amazon Auto EC2 Scaling.
- Gunakan strategi alokasi Spot yang dioptimalkan kapasitas - Dengan Amazon EC2 Spot, Anda dapat memilih antara strategi alokasi kapasitas dan biaya yang dioptimalkan. Jika Anda memilih strategi yang dioptimalkan kapasitas saat meluncurkan instans baru, Amazon EC2 Spot memilih jenis instans dengan ketersediaan terbesar di Availability Zone yang dipilih. Ini membantu mengurangi kemungkinan bahwa instance dihentikan segera setelah diluncurkan.

Untuk informasi tentang cara mengonfigurasi pemberitahuan penghentian spot pada instance container Anda, lihat:

- Mengonfigurasi instans container Amazon ECS Linux untuk menerima pemberitahuan Instans Spot
- Mengonfigurasi instans penampung Amazon ECS Windows untuk menerima pemberitahuan Instans Spot

Linux Amazon ECS yang dioptimalkan AMIs

> **Important**
>
> Amazon ECS yang dioptimalkan Amazon Linux 2 AMI mencapai end-of-life pada 30 Juni 2026, mencerminkan tanggal EOL yang sama dari sistem operasi Amazon Linux 2 hulu

(untuk informasi lebih lanjut, lihat Amazon Linux 2). FAQs Kami mendorong pelanggan untuk meningkatkan aplikasi mereka untuk menggunakan Amazon Linux 2023, yang mencakup dukungan jangka panjang hingga 2028. Untuk informasi tentang migrasi dari Amazon Linux 2 ke Amazon Linux 2023, lihat Memigrasi dari AMI Amazon Linux 2 Amazon ECS yang dioptimalkan ke AMI Amazon Linux 2023 yang dioptimalkan Amazon ECS.

Amazon ECS menyediakan Amazon ECS yang dioptimalkan AMIs yang telah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Kami menyarankan Anda menggunakan Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon untuk instans Amazon EC2 Anda. Meluncurkan instans penampung Anda dari AMI terbaru yang dioptimalkan Amazon ECS memastikan bahwa Anda menerima pembaruan keamanan saat ini dan versi agen penampung. Untuk informasi tentang cara meluncurkan instance, lihatMeluncurkan instans penampung Amazon ECS Linux.

Saat Anda membuat klaster menggunakan konsol, Amazon ECS membuat template peluncuran untuk instans Anda dengan AMI terbaru yang terkait dengan sistem operasi yang dipilih.

Saat Anda menggunakan AWS CloudFormation untuk membuat klaster, parameter SSM adalah bagian dari template EC2 peluncuran Amazon untuk instance grup Auto Scaling. Anda dapat mengonfigurasi template untuk menggunakan parameter Systems Manager dinamis untuk menentukan apa yang akan digunakan oleh Amazon ECS Optimized AMI. Parameter ini memastikan bahwa setiap kali Anda menerapkan tumpukan itu akan memeriksa untuk melihat apakah ada pembaruan yang tersedia yang perlu diterapkan ke EC2 instance. Untuk contoh cara menggunakan parameter Systems Manager, lihat Membuat klaster Amazon ECS dengan Amazon ECS AMI Amazon Linux 2023 yang dioptimalkan Amazon ECS di AWS CloudFormation Panduan Pengguna.

Jika Anda perlu menyesuaikan AMI Amazon ECS yang dioptimalkan, lihat Amazon ECS Optimized AMI Build Recipes di. GitHub

Varian berikut dari AMI Amazon ECS yang dioptimalkan tersedia untuk EC2 instans Amazon Anda dengan sistem operasi Amazon Linux 2023.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023	Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI	Amazon Linux 2023 adalah generasi berikutnya dari	Secara default, Amazon ECS Amazon Linux 2023 AMI yang

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
		Amazon Linux dari AWS. Untuk sebagian besar kasus, disarankan untuk meluncurkan EC2 instans Amazon Anda untuk beban kerja Amazon ECS Anda. Untuk informasi selengkapnya, lihat Apa itu Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.	dioptimalkan Amazon dikirimkan dengan volume root 30-GiB tunggal. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI adalah`xfs`, dan Docker menggunak an driver penyimpan an. `overlay2` Untuk informasi selengkap nya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023 (arm64)	Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI	Berdasarkan Amazon Linux 2023, AMI ini direkomendasikan untuk digunakan saat meluncurkan EC2 instans Amazon Anda, yang didukung oleh Prosesor AWS Graviton/Graviton 2/ Graviton 3/Graviton 4 berbasis ARM, untuk beban kerja Amazon ECS Anda. Untuk informasi selengkap nya, lihat Spesifikasi untuk instans tujuan EC2 umum Amazon dalam panduan Jenis EC2 Instans Amazon.	Secara default, Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon dikirimkan dengan volume root 30-GiB tunggal. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI adalah`xfs`, dan Docker menggunak an driver penyimpan an. `overlay2` Untuk informasi selengkap nya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2023 (Neuron)	Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI	Berdasarkan Amazon Linux 2023, ini AMIis untuk instans Amazon EC2 Inf1, Trn1, atau Inf2. Muncul pra-konfi gurasi dengan driver AWS Inferentia dan AWS Trainium dan runtime AWS Neuron untuk Docker yang membuat menjalank an beban kerja inferensi pembelaja ran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja pembelajaran mesin AWS Neuron. Amazon ECS yang dioptimalkan Amazon Linux 2023 (Neuron) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	Secara default, Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon dikirimkan dengan volume root 30-GiB tunggal. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI adalah`xfs`, dan Docker menggunak an driver penyimpan an. `overlay2` Untuk informasi selengkap nya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
GPU Amazon Linux 2023	Amazon ECS dioptimalkan Amazon Linux 2023 GPU AMI	Berdasarkan Amazon Linux 2023, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans EC2 berbasis GPU Amazon untuk beban kerja Amazon ECS Anda. Muncul pra-konfigurasi dengan driver kernel NVIDIA dan runtime GPU Docker yang membuat beban kerja berjalan yang memanfaatkan Amazon ECS. GPUs Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.	Secara default, Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon dikirimkan dengan volume root 30-GiB tunggal. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI adalah`xfs`, dan Docker menggunak an driver penyimpan an. `overlay2` Untuk informasi selengkap nya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Varian berikut dari AMI Amazon ECS yang dioptimalkan tersedia untuk EC2 instans Amazon Anda dengan sistem operasi Amazon Linux 2.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2	Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini digunakan saat meluncurkan EC2 instans Amazon Anda dan Anda ingin menggunakan kernel Linux 5.10 alih-alih kernel 4.14 untuk beban kerja Amazon ECS Anda. Kernel 5.10 AMI Amazon Linux 2 yang dioptimalkan Amazon ECS tidak disertakan dengan AWS CLI yang sudah diinstal sebelumnya.	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimalkan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimalkan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncuran untuk meningkatkan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Sistem file default untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI adalah`xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2	Amazon ECS yang dioptimalkan Amazon Linux 2 AMI	Ini untuk beban kerja Amazon ECS Anda. Amazon ECS yang dioptimalkan Amazon Linux 2 AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimalkan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimalkan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncuran untuk meningkatkan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (arm64)	Amazon ECS yang dioptimalkan Amazon Linux 2 kernel 5.10 (arm64) AMI	Berdasarkan Amazon Linux 2, AMI ini untuk EC2 instans Amazon Anda, yang didukung oleh AWS Graviton/Graviton 2/ Graviton 3/Graviton 4 Prosesor berbasis ARM, dan Anda ingin menggunak an kernel Linux 5.10 alih-alih kernel Linux 4.14 untuk beban kerja Amazon ECS Anda. Untuk informasi selengkapnya, lihat Spesifikasi untuk instans tujuan EC2 umum Amazon di panduan Jenis EC2 Instans Amazon. Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimal kan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimal kan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (arm64)	Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI	Berdasarkan Amazon Linux 2, AMI ini digunakan saat meluncurkan EC2 instans Amazon Anda, yang didukung oleh AWS Graviton/ Graviton 2/Graviton 3/ Graviton 4 Prosesor berbasis ARM, untuk beban kerja Amazon ECS Anda. Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI tidak disertakan dengan yang sudah diinstal sebelumnya. AWS CLI	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimal kan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimal kan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (GPU)	Kernel Amazon ECS GPU yang dioptimal kan 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans EC2 berbasis GPU Amazon Anda dengan kernel Linux 5.10 untuk beban kerja Amazon ECS Anda. Muncul pra-konfi gurasi dengan driver kernel NVIDIA dan runtime GPU Docker yang membuat beban kerja berjalan yang memanfaat kan Amazon ECS. GPUs Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimal kan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimal kan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (GPU)	AMI Amazon ECS GPU yang dioptimal kan	Berdasarkan Amazon Linux 2, AMI ini direkomendasikan untuk digunakan saat meluncurkan instans EC2 berbasis GPU Amazon Anda dengan kernel Linux 4.14 untuk beban kerja Amazon ECS Anda. Muncul pra-konfi gurasi dengan driver kernel NVIDIA dan runtime GPU Docker yang membuat beban kerja berjalan yang memanfaat kan Amazon ECS. GPUs Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimal kan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimal kan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (Neuron)	Amazon ECS mengoptimalkan kernel Amazon Linux 2 (Neuron) 5.10 AMI	Berdasarkan Amazon Linux 2, AMI ini untuk instans Amazon EC2 Inf1, Trn1 atau Inf2. Muncul pra-konfigurasi dengan AWS Inferentia dengan kernel Linux 5.10 dan driver AWS Trainium dan runtime AWS Neuron untuk Docker yang membuat menjalankan beban kerja inferensi pembelajaran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja pembelajaran mesin AWS Neuron. Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI tidak disertakan dengan yang AWS CLI sudah diinstal sebelumnya.	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimalkan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimalkan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncuran untuk meningkatkan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
Amazon Linux 2 (Neuron)	Amazon ECS dioptimalkan Amazon Linux 2 (Neuron) AMI	Berdasarkan Amazon Linux 2, AMI ini untuk instans Amazon EC2 Inf1, Trn1 atau Inf2. Muncul pra-konfi gurasi dengan driver AWS Inferentia dan AWS Trainium dan runtime AWS Neuron untuk Docker yang membuat menjalank an beban kerja inferensi pembelaja ran mesin lebih mudah di Amazon ECS. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja pembelajaran mesin AWS Neuron. Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) AMI tidak disertaka n dengan yang AWS CLI sudah diinstal sebelumnya.	Secara default, Amazon ECS berbasis Amazon Linux 2 yang dioptimal kan (Amazon ECS yang dioptimalkan AMIs Amazon Linux 2 AMI, Amazon ECS Amazon Linux 2 (arm64) AMI yang dioptimalkan oleh Amazon ECS, dan AMI yang dioptimal kan untuk GPU Amazon ECS) dikirimkan dengan satu volume root 30-GiB. Anda dapat mengubah ukuran volume root 30 GiB pada waktu peluncura n untuk meningkat kan penyimpanan yang tersedia pada instans kontainer Anda. Penyimpanan ini digunakan untuk sistem operasi dan untuk citra Docker serta metadata. Sistem file default untuk Amazon ECS yang dioptimalkan

Sistem operasi	AMI	Deskripsi	Konfigurasi penyimpanan
			Amazon Linux 2 AMI adalah `xfs`, dan Docker menggunakan driver penyimpanan. `overlay2` Untuk informasi selengkapnya, lihat Gunakan driver penyimpanan OverlayFS dalam dokumentasi Docker.

Amazon ECS menyediakan changelog untuk varian Linux dari AMI yang dioptimalkan Amazon ECS. GitHub Untuk informasi lebih lanjut, lihat Changelog.

Varian Linux dari AMI Amazon ECS yang dioptimalkan menggunakan Amazon Linux 2 AMI atau Amazon Linux 2023 AMI sebagai basis mereka. Anda dapat mengambil nama AMI untuk setiap varian dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan. Catatan rilis Amazon Linux 2 AMI juga tersedia. Untuk informasi selengkapnya, lihat catatan rilis Amazon Linux 2. Catatan rilis Amazon Linux 2023 juga tersedia. Untuk informasi lebih lanjut lihat, catatan rilis Amazon Linux 2023.

Halaman-halaman berikut memberikan informasi tambahan tentang perubahan:

- Sumber catatan rilis AMI pada GitHub
- Catatan rilis Docker Engine dalam dokumentasi Docker
- Dokumentasi Driver NVIDIA dalam dokumentasi NVIDIA
- Changelog agen Amazon ECS aktif GitHub

  Kode sumber untuk `ecs-init` aplikasi dan skrip dan konfigurasi untuk pengemasan agen sekarang menjadi bagian dari repositori agen. Untuk versi `ecs-init` dan kemasan yang lebih lama, lihat Amazon ecs-init changelog di GitHub

Menerapkan pembaruan keamanan ke AMI Amazon ECS yang dioptimalkan

Amazon ECS yang dioptimalkan AMIs berdasarkan Amazon Linux berisi versi yang disesuaikan. cloud-init Cloud-initadalah paket yang digunakan untuk mem-bootstrap gambar Linux di lingkungan komputasi awan dan melakukan tindakan yang diinginkan saat meluncurkan instance. Secara default, semua Amazon ECS yang dioptimalkan AMIs berdasarkan Amazon Linux yang dirilis sebelum 12 Juni 2024 memiliki semua pembaruan keamanan "Kritis" dan "Penting" yang diterapkan pada saat peluncuran instance.

Dimulai dengan rilis 12 Juni 2024 dari Amazon ECS yang dioptimalkan AMIs berdasarkan Amazon Linux 2, perilaku default tidak akan lagi mencakup pembaruan paket saat peluncuran. Sebagai gantinya, kami menyarankan Anda memperbarui ke AMI baru yang dioptimalkan Amazon ECS saat rilis tersedia. Amazon ECS yang dioptimalkan AMIs dirilis ketika ada pembaruan keamanan yang tersedia atau perubahan AMI dasar. Ini akan memastikan Anda menerima versi paket terbaru dan pembaruan keamanan, dan bahwa versi paket tidak dapat diubah melalui peluncuran instance. Untuk informasi selengkapnya tentang mengambil AMI terbaru yang dioptimalkan Amazon ECS, lihat. Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan

Kami merekomendasikan untuk mengotomatiskan lingkungan Anda untuk memperbarui ke AMI baru saat tersedia. Untuk informasi tentang opsi yang tersedia, lihat Amazon ECS memungkinkan pengelolaan EC2 kapasitas yang lebih mudah, dengan pengurasan instans terkelola.

Untuk terus menerapkan pembaruan keamanan "Kritis" dan "Penting" secara manual pada versi AMI, Anda dapat menjalankan perintah berikut di EC2 instans Amazon Anda.

```
yum update --security
```

Jika ingin mengaktifkan kembali pembaruan keamanan saat peluncuran, Anda dapat menambahkan baris berikut ke `#cloud-config` bagian data pengguna cloud-init saat meluncurkan instans Amazon Anda. EC2 Untuk informasi selengkapnya, lihat Menggunakan cloud-init di Amazon Linux 2 di Panduan Pengguna Amazon Linux.

```
#cloud-config
repo_upgrade: security
```

Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan

Anda dapat mengambil metadata AMI Amazon ECS yang dioptimalkan secara terprogram. Metadata mencakup nama AMI, versi agen penampung Amazon ECS, dan versi runtime Amazon ECS yang mencakup versi Docker.

Saat Anda membuat klaster menggunakan konsol, Amazon ECS membuat template peluncuran untuk instans Anda dengan AMI terbaru yang terkait dengan sistem operasi yang dipilih.

Saat Anda menggunakan AWS CloudFormation untuk membuat klaster, parameter SSM adalah bagian dari template EC2 peluncuran Amazon untuk instance grup Auto Scaling. Anda dapat mengonfigurasi template untuk menggunakan parameter Systems Manager dinamis untuk menentukan apa yang akan digunakan oleh Amazon ECS Optimized AMI. Parameter ini memastikan bahwa setiap kali Anda menerapkan tumpukan itu akan memeriksa untuk melihat apakah ada pembaruan yang tersedia yang perlu diterapkan ke EC2 instance. Untuk contoh cara menggunakan parameter Systems Manager, lihat Membuat klaster Amazon ECS dengan Amazon ECS AMI Amazon Linux 2023 yang dioptimalkan Amazon ECS di AWS CloudFormation Panduan Pengguna.

ID AMI, nama gambar, sistem operasi, versi agen kontainer, nama gambar sumber, dan versi runtime untuk setiap varian Amazon ECS yang dioptimalkan AMIs dapat diambil secara terprogram dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParametersdan GetParametersByPath.

> **Note**
>
> Pengguna administratif Anda harus memiliki izin IAM berikut untuk mengambil metadata AMI Amazon ECS yang dioptimalkan. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM.
>
> - ssm: GetParameters
> - ssm: GetParameter
> - ssm: GetParametersByPath

Systems Manager Parameter Menyimpan format parameter

Berikut ini adalah format nama parameter untuk setiap varian AMI Amazon ECS yang dioptimalkan.

Linux Amazon ECS dioptimalkan AMIs

- Metadata AMI Amazon Linux 2023:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/<version>
```

- Amazon Linux 2023 (arm64) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/<version>
```

- Amazon Linux 2023 (Neuron) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/<version>
```

- Amazon Linux 2023 (GPU) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/gpu/<version>
```

Amazon Linux 2 AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/<version>
```

- Amazon Linux 2 kernel 5.10 metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/<version>
```

- Amazon Linux 2 (arm64) AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/arm64/<version>
```

- Amazon Linux 2 kernel 5.10 (arm64) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/arm64/<version>
```

- Metadata kernel 5.10 AMI yang dioptimalkan oleh Amazon ECS GPU:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/gpu/<version>
```

- Amazon Linux 2 (GPU) AMI metadata:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/<version>
```

- Amazon ECS mengoptimalkan metadata kernel Amazon Linux 2 (Neuron) 5.10 AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/inf/<version>
```

- Amazon Linux 2 (Neuron) metadata AMI:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/inf/<version>
```

Format nama parameter berikut mengambil ID gambar dari Amazon ECS terbaru yang dioptimalkan Amazon ECS yang dioptimalkan Amazon Linux 2 AMI dengan menggunakan sub-parameter. `image_id`

```
/aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

Format nama parameter berikut mengambil metadata versi AMI tertentu yang dioptimalkan Amazon ECS dengan menentukan nama AMI.

- Metadata Amazon Linux 2 AMI Amazon ECS yang dioptimalkan oleh Amazon ECS:

```
/aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

> **Note**
>
> Semua versi Amazon ECS yang dioptimalkan Amazon Linux 2 AMI tersedia untuk pengambilan. Hanya versi AMI Amazon ECS yang dioptimalkan `amzn-ami-2017.09.l-amazon-ecs-optimized` (Linux) dan yang lebih baru yang dapat diambil.

Contoh

Contoh berikut menunjukkan cara Anda dapat mengambil metadata untuk setiap varian AMI Amazon ECS yang dioptimalkan.

Mengambil metadata AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan

Anda dapat mengambil AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan menggunakan AWS CLI perintah berikut. AWS CLI

Linux Amazon ECS dioptimalkan AMIs

- Untuk Amazon ECS Amazon Linux 2023 yang dioptimalkan oleh Amazon ECS: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64): AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
arm64/recommended --region us-east-1
```

- Untuk Amazon ECS Amazon Linux 2023 (Neuron) yang dioptimalkan oleh Amazon ECS: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
neuron/recommended --region us-east-1
```

- Untuk GPU Amazon Linux 2023 Amazon ECS yang dioptimalkan oleh Amazon: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/gpu/
recommended --region us-east-1
```

- Untuk kernel Amazon Linux 2 yang dioptimalkan Amazon ECS 5.10: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended --region us-east-1
```

- Untuk Amazon ECS Amazon Linux 2 yang dioptimalkan oleh Amazon: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --region us-east-1
```

- Untuk kernel Amazon Linux 2 yang dioptimalkan Amazon ECS 5.10 (arm64): AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/arm64/recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64): AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/arm64/
recommended --region us-east-1
```

- Untuk kernel 5.10 yang dioptimalkan oleh GPU Amazon ECS: AMIs

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/gpu/recommended --region us-east-1
```

- Untuk Amazon ECS AMIs GPU yang dioptimalkan:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 (Neuron) kernel AMIs 5.10:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/inf/recommended --region us-east-1
```

- Untuk Amazon ECS dioptimalkan Amazon Linux 2 (Neuron) AMIs:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended --region us-east-1
```

Mengambil ID gambar dari Amazon ECS terbaru yang dioptimalkan Amazon ECS 2023 AMI yang dioptimalkan

Anda dapat mengambil ID gambar dari ID AMI Amazon Linux 2023 Amazon ECS terbaru yang dioptimalkan oleh Amazon ECS dengan menggunakan sub-parameter. `image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-
linux-2023/recommended/image_id --region us-east-1
```

Untuk mengambil nilai `image_id` saja, Anda dapat meng-kueri nilai parameter tertentu; misalnya:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

Mengambil metadata versi Amazon Linux 2 AMI Amazon ECS tertentu yang dioptimalkan

Ambil metadata versi Amazon Linux Linux AMI yang dioptimalkan Amazon ECS tertentu menggunakan perintah berikut AWS CLI . AWS CLI Ganti nama AMI dengan nama Amazon ECS yang dioptimalkan Amazon Linux AMI untuk diambil.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20200928-x86_64-ebs --region us-east-1
```

Mengambil metadata AMI kernel 5.10 AMI Amazon Linux 2 yang dioptimalkan Amazon ECS menggunakan Systems Manager API GetParametersByPath

Ambil metadata Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon dengan Systems GetParametersByPath Manager API menggunakan perintah AWS CLI berikut.

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/ --region us-east-1
```

Mengambil ID gambar dari kernel Amazon Linux 2 5.10 AMI yang dioptimalkan Amazon ECS terbaru yang direkomendasikan

Anda dapat mengambil ID gambar dari ID AMI 5.10 kernel 5.10 Amazon ECS terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan menggunakan sub-parameter. `image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended/image_id --region us-east-1
```

Untuk mengambil nilai `image_id` saja, Anda dapat meng-kueri nilai parameter tertentu; misalnya:

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

Menggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation

Anda dapat mereferensikan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager.

Contoh Linux

```
Parameters:kernel-5.10
  LatestECSOptimizedAMI:
    Description: AMI ID
```

```
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended/
image_id
```

Bermigrasi dari Amazon Linux 2 ke Amazon Linux 2023 AMI yang dioptimalkan Amazon ECS

Mengikuti Amazon Linux, Amazon ECS mengakhiri dukungan standar untuk Amazon Linux 2 Amazon ECS yang dioptimalkan AMIs efektif 30 Juni 2026. Setelah tanggal ini, versi agen Amazon ECS disematkan dan Amazon Linux 2 Amazon ECS baru yang AMIs dioptimalkan hanya diterbitkan ketika sumber Amazon Linux 2 AMI diperbarui. Complete End of Life (EOL) terjadi pada 30 Juni 2026, setelah itu tidak ada lagi Amazon Linux AMIs 2 Amazon ECS yang dioptimalkan Amazon yang dioptimalkan, bahkan jika sumber AMI diperbarui.

Amazon Linux 2023 menyediakan secure-by-default pendekatan dengan kebijakan keamanan yang telah dikonfigurasi sebelumnya, SELinux dalam mode permisif, mode IMDSv2 -only diaktifkan secara default, waktu boot yang dioptimalkan, dan manajemen paket yang ditingkatkan untuk meningkatkan keamanan dan kinerja.

Ada tingkat kompatibilitas yang tinggi antara Amazon Linux 2 dan Amazon Linux 2023 Amazon ECS yang dioptimalkan AMIs, dan sebagian besar pelanggan akan mengalami minimal-to-zero perubahan dalam beban kerja mereka di antara kedua sistem operasi.

Untuk informasi selengkapnya, lihat Membandingkan Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023 dan 023. AL2 FAQs

Pertimbangan kompatibilitas

Manajemen paket dan pembaruan OS

Tidak seperti versi Amazon Linux sebelumnya, Amazon ECS yang dioptimalkan Amazon Linux 2023 AMIs dikunci ke versi tertentu dari repositori Amazon Linux. Ini melindungi pengguna dari memperbarui paket secara tidak sengaja yang mungkin membawa perubahan yang tidak diinginkan atau melanggar. Untuk informasi selengkapnya, lihat Mengelola repositori dan pembaruan OS di Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Versi kernel Linux

Amazon Linux 2 AMIs didasarkan pada kernel Linux 4.14 dan 5.10, sedangkan Amazon Linux 2023 menggunakan kernel Linux 6.1 dan 6.12. Untuk informasi selengkapnya, lihat Membandingkan kernel Amazon Linux 2 dan Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Perubahan ketersediaan Package

Berikut ini adalah perubahan paket penting di Amazon Linux 2023:

- Beberapa paket biner sumber di Amazon Linux 2 tidak lagi tersedia di Amazon Linux 2023. Untuk informasi selengkapnya, lihat Paket yang dihapus dari Amazon Linux 2023 di Catatan Rilis Amazon Linux 2023.
- Perubahan cara Amazon Linux mendukung versi paket yang berbeda. `amazon-linux-extras`Sistem yang digunakan di Amazon Linux 2 tidak ada di Amazon Linux 2023. Semua paket hanya tersedia di repositori "inti".
- Paket tambahan untuk Enterprise Linux (EPEL) tidak didukung di Amazon Linux 2023. Untuk informasi selengkapnya, lihat Kompatibilitas EPEL di Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.
- Aplikasi 32-bit tidak didukung di Amazon Linux 2023. Untuk informasi selengkapnya, lihat Fitur usang dari Amazon Linux 2 di Panduan Pengguna Amazon Linux 2023.

Grup Kontrol (cgroups) berubah

Control Group (cgroup) adalah fitur kernel Linux untuk mengatur proses secara hierarkis dan mendistribusikan sumber daya sistem di antara mereka. Grup Kontrol digunakan secara ekstensif untuk mengimplementasikan runtime kontainer, dan oleh. `systemd`

Agen Amazon ECS, Docker, dan containerd semuanya mendukung cgroupv1 dan cgroupv2. Agen Amazon ECS dan runtime container mengelola cgroup untuk Anda, sehingga pelanggan Amazon ECS tidak perlu melakukan perubahan apa pun untuk peningkatan cgroup yang mendasarinya.

Untuk detail lebih lanjut tentang cgroupv2, lihat Grup kontrol v2 di Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Perubahan Layanan Metadata Instance (IMDS)

Amazon Linux 2023 memerlukan Layanan Metadata Instans versi 2 (IMDSv2) secara default. IMDSv2 memiliki beberapa manfaat yang membantu meningkatkan postur keamanan. Ini menggunakan metode otentikasi berorientasi sesi yang memerlukan pembuatan token rahasia dalam permintaan HTTP PUT sederhana untuk memulai sesi. Token sesi dapat berlaku di mana saja antara 1 detik dan 6 jam.

Untuk informasi selengkapnya tentang cara transisi dari IMDSv1 ke IMDSv2, lihat Transisi ke menggunakan Layanan Metadata Instans Versi 2 di EC2 Panduan Pengguna Amazon.

Jika Anda ingin menggunakannya IMDSv1, Anda masih dapat melakukannya dengan mengganti pengaturan secara manual menggunakan properti peluncuran opsi metadata instance.

Perubahan pertukaran memori

Pertukaran memori per kontainer tidak didukung di Amazon Linux 2023 dan cgroups v2. Untuk informasi selengkapnya, lihat Mengelola ruang memori swap kontainer di Amazon ECS.

Perubahan validasi FIPS

Amazon Linux 2 disertifikasi di bawah FIPS 140-2 dan Amazon Linux 2023 disertifikasi di bawah FIPS 140-3.

Untuk mengaktifkan mode FIPS di Amazon Linux 2023, instal paket yang diperlukan di EC2 instans Amazon Anda dan ikuti langkah-langkah konfigurasi menggunakan petunjuk dalam Aktifkan Mode FIPS di Amazon Linux 2023 di Panduan Pengguna Amazon Linux 2023.

Dukungan instans yang dipercepat

Amazon Linux 2023 yang dioptimalkan oleh Amazon ECS AMIs mendukung jenis instans berakselerasi Neuron dan GPU. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

Membangun kustom AMIs

Meskipun kami merekomendasikan untuk beralih ke Amazon ECS yang didukung dan diterbitkan secara resmi yang dioptimalkan untuk AMIs Amazon Linux 2023, Anda dapat terus membangun Amazon Linux 2 Amazon ECS khusus yang dioptimalkan AMIs menggunakan skrip build open-source yang digunakan untuk membangun varian Linux dari AMI Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Skrip pembuatan AMI Linux AMI yang dioptimalkan Amazon ECS.

Strategi migrasi

Sebaiknya buat dan implementasikan rencana migrasi yang mencakup pengujian aplikasi menyeluruh. Bagian berikut menguraikan strategi migrasi yang berbeda berdasarkan cara Anda mengelola infrastruktur Amazon ECS Anda.

Bermigrasi dengan penyedia kapasitas Amazon ECS

1. Buat penyedia kapasitas baru dengan templat peluncuran baru. Ini harus mereferensikan grup Auto Scaling dengan template peluncuran yang mirip dengan yang sudah ada, tetapi alih-alih

AMI Amazon Linux 2 Amazon ECS yang dioptimalkan, itu harus menentukan salah satu varian Amazon Linux 2023. Tambahkan penyedia kapasitas baru ini ke cluster Amazon ECS Anda yang sudah ada.

2. Perbarui strategi penyedia kapasitas default cluster Anda untuk menyertakan penyedia kapasitas Amazon Linux 2 yang ada dan penyedia kapasitas Amazon Linux 2023 yang baru. Mulailah dengan bobot yang lebih tinggi pada penyedia Amazon Linux 2 dan bobot yang lebih rendah pada penyedia Amazon Linux 2023 (misalnya, Amazon Linux 2: berat 80, Amazon Linux 2023: berat 20). Hal ini menyebabkan Amazon ECS mulai menyediakan instans Amazon Linux 2023 saat tugas baru dijadwalkan. Verifikasi bahwa instans terdaftar dengan benar dan bahwa tugas dapat berjalan dengan sukses pada instance baru.
3. Secara bertahap sesuaikan bobot penyedia kapasitas dalam strategi default cluster Anda, tingkatkan bobot untuk penyedia Amazon Linux 2023 sambil mengurangi bobot penyedia Amazon Linux 2 dari waktu ke waktu (misalnya, 60/40, lalu 40/60, lalu 20/80). Anda juga dapat memperbarui strategi penyedia kapasitas layanan individual untuk memprioritaskan instans Amazon Linux 2023. Pantau penempatan tugas untuk memastikan berhasil berjalan di instans Amazon Linux 2023.
4. Secara opsional menguras instans penampung Amazon Linux 2 untuk mempercepat migrasi tugas. Jika Anda memiliki kapasitas penggantian Amazon Linux 2023 yang memadai, Anda dapat menguras instans penampung Amazon Linux 2 secara manual melalui konsol Amazon ECS atau AWS CLI untuk mempercepat transisi tugas Anda dari Amazon Linux 2 ke Amazon Linux 2023. Setelah migrasi selesai, hapus penyedia kapasitas Amazon Linux 2 dari klaster Anda dan hapus grup Auto Scaling terkait.

Bermigrasi dengan grup Amazon EC2 Auto Scaling

1. Buat grup Amazon EC2 Auto Scaling baru dengan template peluncuran baru. Ini harus mirip dengan template peluncuran Anda yang ada, tetapi alih-alih AMI Amazon Linux 2 Amazon ECS yang dioptimalkan, itu harus menentukan salah satu varian Amazon Linux 2023. Grup Auto Scaling baru ini dapat meluncurkan instance ke cluster Anda yang ada.
2. Tingkatkan grup Auto Scaling sehingga Anda mulai memiliki instans Amazon Linux 2023 yang mendaftar ke klaster Anda. Verifikasi bahwa instans terdaftar dengan benar dan bahwa tugas dapat berjalan dengan sukses pada instance baru.
3. Setelah tugas Anda diverifikasi untuk berfungsi di Amazon Linux 2023, tingkatkan grup Auto Scaling Amazon Linux 2023 sambil secara bertahap mengurangi grup Auto Scaling Amazon Linux 2, hingga Anda benar-benar mengganti semua instans Amazon Linux 2.

4. Jika Anda memiliki kapasitas penggantian Amazon Linux 2023 yang memadai, Anda mungkin ingin menguras instance container secara eksplisit untuk mempercepat transisi tugas Anda dari Amazon Linux 2 ke Amazon Linux 2023. Untuk informasi selengkapnya, lihat Menguras instans kontainer Amazon ECS.

Migrasi dengan instans yang dikelola secara manual

1. Luncurkan secara manual (atau sesuaikan skrip yang diluncurkan) EC2 instans Amazon baru menggunakan Amazon ECS Amazon Linux 2023 AMI yang dioptimalkan Amazon ECS alih-alih Amazon Linux 2. Pastikan instans ini menggunakan grup keamanan, subnet, peran IAM, dan konfigurasi klaster yang sama dengan instans Amazon Linux 2 yang ada. Instans harus secara otomatis mendaftar ke cluster Amazon ECS Anda yang ada saat diluncurkan.
2. Pastikan instans Amazon Linux 2023 baru berhasil mendaftar ke cluster Amazon ECS Anda dan berada dalam status. `ACTIVE` Uji bahwa tugas dapat dijadwalkan dan dijalankan dengan benar pada instance baru ini dengan menunggu penempatan tugas alami atau secara manual stopping/starting beberapa tugas untuk memicu penjadwalan ulang.
3. Ganti instans Amazon Linux 2 Anda secara bertahap dengan meluncurkan instans Amazon Linux 2023 tambahan sesuai kebutuhan, lalu menguras dan menghentikan instans Amazon Linux 2 secara manual satu per satu. Anda dapat menguras instans melalui konsol Amazon ECS dengan menyetel instans ke `DRAINING` status, yang akan berhenti menempatkan tugas baru di dalamnya dan memungkinkan tugas yang ada selesai atau dijadwalkan ulang di tempat lain.

Skrip pembuatan AMI Linux AMI yang dioptimalkan Amazon ECS

Amazon ECS telah membuka skrip build yang digunakan untuk membangun varian Linux dari AMI Amazon ECS yang dioptimalkan. Skrip build ini sekarang tersedia di GitHub. Untuk informasi lebih lanjut, lihat amazon-ecs-amidi GitHub.

Jika Anda perlu menyesuaikan AMI Amazon ECS yang dioptimalkan, lihat Amazon ECS Optimized AMI Build Recipies aktif. GitHub

Repositori skrip build menyertakan template HashiCorppengemas dan skrip build untuk menghasilkan setiap varian Linux dari AMI Amazon ECS yang dioptimalkan. Skrip ini adalah sumber kebenaran untuk build AMI yang dioptimalkan Amazon ECS, sehingga Anda dapat mengikuti repositori GitHub untuk memantau perubahan pada kami. AMIs Misalnya, mungkin Anda ingin AMI Anda sendiri menggunakan versi Docker yang sama dengan yang digunakan tim Amazon ECS untuk AMI resmi.

Untuk informasi selengkapnya, lihat repositori Amazon ECS AMI di aws/ on. amazon-ecs-ami GitHub

Untuk membangun AMI Linux Amazon ECS yang dioptimalkan

1. Kloning `aws/amazon-ecs-ami` GitHub repo.

```
git clone https://github.com/aws/amazon-ecs-ami.git
```

2. Tambahkan variabel lingkungan untuk AWS Wilayah yang akan digunakan saat membuat AMI. Ganti `us-west-2` nilainya dengan Region yang akan digunakan.

```
export REGION=us-west-2
```

3. Makefile disediakan untuk membangun AMI. Dari direktori root repositori kloning, gunakan salah satu perintah berikut, sesuai dengan varian Linux dari AMI Amazon ECS yang dioptimalkan yang ingin Anda buat.

   - Amazon ECS yang dioptimalkan Amazon Linux 2 AMI

   ```
   make al2
   ```

   - Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI

   ```
   make al2arm
   ```

   - AMI Amazon ECS GPU yang dioptimalkan

   ```
   make al2gpu
   ```

   - Amazon ECS mengoptimalkan Amazon Linux 2 (Neuron) AMI

   ```
   make al2inf
   ```

   - Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI

   ```
   make al2023
   ```

   - Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI

   ```
   make al2023arm
   ```

- Amazon ECS mengoptimalkan Amazon Linux 2023 (Neuron) AMI

```
make al2023neu
```

Amazon ECS dioptimalkan Bottlerocket AMIs

Bottlerocketadalah sistem operasi open-source Linux berbasis yang dibangun AWS untuk menjalankan kontainer pada mesin virtual atau host bare metal. BottlerocketAMI yang dioptimalkan Amazon ECS aman dan hanya menyertakan jumlah minimum paket yang diperlukan untuk menjalankan kontainer. Ini meningkatkan penggunaan sumber daya, mengurangi permukaan serangan keamanan, dan membantu menurunkan overhead manajemen. BottlerocketAMI juga terintegrasi dengan Amazon ECS untuk membantu mengurangi overhead operasional yang terlibat dalam memperbarui instance kontainer dalam sebuah cluster.

Bottlerocketberbeda dari Amazon Linux dengan cara berikut:

- Bottlerockettidak termasuk manajer paket, dan perangkat lunaknya hanya dapat dijalankan sebagai wadah. Pembaruan Bottlerocket keduanya diterapkan dan dapat diputar kembali dalam satu langkah, yang mengurangi kemungkinan kesalahan pembaruan.
- Mekanisme utama untuk mengelola Bottlerocket host adalah dengan penjadwal kontainer. Tidak seperti Amazon Linux, masuk ke Bottlerocket instance individual dimaksudkan untuk menjadi operasi yang jarang terjadi hanya untuk tujuan debugging dan pemecahan masalah tingkat lanjut.

Untuk informasi selengkapnyaBottlerocket, lihat dokumentasi dan rilis diGitHub.

Ada varian Bottlerocket AMI Amazon ECS yang dioptimalkan untuk kernel 6.1 dan kernel 5.10.

Varian berikut menggunakan kernel 6.1:

- `aws-ecs-2`
- `aws-ecs-2-nvidia`

Varian berikut menggunakan kernel 5.10:

- `aws-ecs-1`
- `aws-ecs-1-nvidia`

Untuk informasi selengkapnya tentang `aws-ecs-1-nvidia` varian, lihat Mengumumkan dukungan GPU NVIDIA untuk Bottlerocket Amazon ECS.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Bottlerocket AMI dengan Amazon ECS.

- Bottlerocketmendukung EC2 instans Amazon dengan `x86_64` dan `arm64` prosesor. BottlerocketAMI tidak direkomendasikan untuk digunakan dengan EC2 instans Amazon dengan chip Inferentia.
- Bottlerocketgambar tidak menyertakan server SSH atau shell. Namun, Anda dapat menggunakan alat out-of-band manajemen untuk mendapatkan akses administrator SSH dan melakukan bootstrap. Untuk informasi lebih lanjut, lihat bagian ini di bottlerocket README.md di: GitHub
  - Eksplorasi
  - Kontainer admin
- Secara default, Bottlerocket memiliki wadah kontrol yang diaktifkan. Container ini menjalankan AWS Systems Manager agen yang dapat Anda gunakan untuk menjalankan perintah atau memulai sesi shell di EC2 Bottlerocket instance Amazon. Untuk informasi selengkapnya, lihat Menyiapkan Pengelola Sesi di Panduan AWS Systems Manager Pengguna.

- Bottlerocket dioptimalkan untuk beban kerja kontainer dan memiliki fokus pada keamanan. Bottlerocket tidak menyertakan manajer paket dan tidak dapat diubah. Untuk informasi tentang fitur dan panduan keamanan, lihat Fitur Keamanan dan Panduan Keamanan di GitHub.
- Mode `awsvpc` jaringan didukung untuk versi AMI `1.1.0` Bottlerocket atau yang lebih baru.
- App Mesh dalam definisi tugas didukung untuk versi AMI `1.15.0` Bottlerocket atau yang lebih baru.
- Parameter definisi `initProcessEnabled` tugas didukung untuk versi AMI `1.19.0` Bottlerocket atau yang lebih baru.
- Bottlerocket AMIs Juga tidak mendukung layanan dan fitur berikut:
  - ECS Di Mana Saja
  - Layanan Connect
  - Amazon EFS dalam mode terenkripsi
  - Amazon EFS dalam mode `awsvpc` jaringan

- Volume Amazon EBS tidak dapat dipasang
- Akselerator Elastic Inference

Mengambil metadata AMI Amazon ECS yang dioptimalkan Bottlerocket

Anda dapat mengambil ID Amazon Machine Image (AMI) untuk Amazon ECS yang dioptimalkan AMIs dengan menanyakan Parameter Store API. AWS Systems Manager Dengan menggunakan parameter ini, Anda tidak perlu mencari AMI yang dioptimalkan Amazon ECS secara manual. IDs Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParameter. Pengguna yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon ECS yang dioptimalkan.

`aws-ecs-2`Varian AMI Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-2` Bottlerocket stabil terbaru Wilayah AWS dengan dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Wilayah yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console— Anda dapat menanyakan ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameter. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64-bit (`x86_64`):

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-2/x86_64/latest/image_id/description?region=region#
```

- Untuk arsitektur Arm (`arm64`) 64-bit:

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-2/arm64/latest/image_id/description?region=region#
```

`aws-ecs-2-nvidia`Varian AMI Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-2-nvdia` Bottlerocket stabil terbaru berdasarkan Wilayah dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Wilayah yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console— Anda dapat meminta ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameter. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64 bit (`x86_64`):

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id/description?region=region#
```

- Untuk arsitektur Arm (`arm64`) 64 bit:

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id/description?region=region#
```

`aws-ecs-1`Varian AMI Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-1` Bottlerocket stabil terbaru Wilayah AWS dengan dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Wilayah yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console— Anda dapat menanyakan ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameter. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64-bit (`x86_64`):

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/x86_64/latest/image_id/description
```

  - Untuk arsitektur Arm (`arm64`) 64-bit:

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/arm64/latest/image_id/description
```

`aws-ecs-1-nvidia`Varian AMI Bottlerocket

Anda dapat mengambil varian AMI `aws-ecs-1-nvdia` Bottlerocket stabil terbaru berdasarkan Wilayah dan arsitektur dengan atau. AWS CLI AWS Management Console

- AWS CLI— Anda dapat mengambil ID gambar dari Bottlerocket AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dengan perintah AWS CLI berikut dengan menggunakan subparameter. `image_id` Ganti *`region`* dengan kode Wilayah yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub. Untuk mengambil versi selain yang terbaru, ganti `latest` dengan nomor versi.
  - Untuk arsitektur 64-bit (`x86_64`):

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console— Anda dapat meminta ID AMI yang dioptimalkan Amazon ECS yang direkomendasikan menggunakan URL di. AWS Management Console URL membuka konsol Amazon EC2 Systems Manager dengan nilai ID untuk parameter. Di URL berikut, ganti *`region`* dengan kode Region yang Anda inginkan ID AMI. Untuk informasi tentang dukungan Wilayah AWS, lihat Menemukan AMI di GitHub.
  - Untuk arsitektur 64 bit (`x86_64`):

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/x86_64/latest/image_id/description?region=region#
    ```

  - Untuk arsitektur Arm (`arm64`) 64 bit:

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/arm64/latest/image_id/description?region=region#
    ```

Langkah selanjutnya

Untuk tutorial terperinci tentang cara memulai dengan sistem Bottlerocket operasi di Amazon ECS, lihat Menggunakan AMI Bottlerocket dengan Amazon ECS aktif dan Memulai Bottlerocket dan Amazon ECS GitHub di situs blog. AWS

Untuk informasi tentang cara meluncurkan instance Bottlerocket, lihat Meluncurkan Bottlerocket instance untuk Amazon ECS

Meluncurkan Bottlerocket instance untuk Amazon ECS

Anda dapat meluncurkan Bottlerocket instance sehingga Anda dapat menjalankan beban kerja kontainer Anda.

Anda dapat menggunakan AWS CLI untuk meluncurkan Bottlerocket instance.

1. Buat file yang dipanggil`userdata.toml`. File ini digunakan untuk data pengguna contoh. Ganti *`cluster-name`* dengan nama klaster Anda.

   ```
   [settings.ecs]
   cluster = "cluster-name"
   ```

2. Gunakan salah satu perintah yang disertakan the section called “Mengambil metadata AMI Amazon ECS yang dioptimalkan Bottlerocket” untuk mendapatkan ID Bottlerocket AMI. Anda menggunakan ini pada langkah berikut.
3. Jalankan perintah berikut untuk meluncurkan Bottlerocket instance. Ingatlah untuk mengganti parameter berikut:

   - Ganti *`subnet`* dengan ID subnet pribadi atau publik tempat instans Anda akan diluncurkan.
   - Ganti *`bottlerocket_ami`* dengan ID AMI dari langkah sebelumnya.
   - Ganti *`t3.large`* dengan jenis instance yang ingin Anda gunakan.
   - Ganti *`region`* dengan kode Region.

   ```
   aws ec2 run-instances --key-name ecs-bottlerocket-example \
      --subnet-id subnet \
      --image-id bottlerocket_ami \
      --instance-type t3.large \
      --region region \
      --tag-specifications
    'ResourceType=instance,Tags=[{Key=bottlerocket,Value=example}]' \
   ```

```
    --user-data file://userdata.toml \
    --iam-instance-profile Name=ecsInstanceRole
```

4. Jalankan perintah berikut untuk memverifikasi bahwa instance container terdaftar ke cluster. Saat Anda menjalankan perintah ini, ingatlah untuk mengganti parameter berikut:

   - Ganti *cluster* dengan nama klaster Anda.
   - Ganti *region* dengan kode Wilayah Anda.

```
aws ecs list-container-instances --cluster cluster-name --region region
```

Untuk panduan terperinci tentang cara memulai dengan sistem Bottlerocket operasi di Amazon ECS, lihat Menggunakan AMI Bottlerocket dengan Amazon ECS aktif GitHub dan Memulai dan Amazon ECS di situs blogBottlerocket. AWS

Manajemen instans penampung Amazon ECS Linux

Saat Anda menggunakan EC2 instans untuk beban kerja Amazon ECS, Anda bertanggung jawab untuk mempertahankan instans

Prosedur manajemen

- Meluncurkan instans penampung Amazon ECS Linux
- Bootstrapping instance container Amazon ECS Linux untuk meneruskan data
- Mengonfigurasi instans container Amazon ECS Linux untuk menerima pemberitahuan Instans Spot
- Menjalankan skrip saat Anda meluncurkan instans penampung Amazon ECS Linux
- Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux
- Memesan memori instans kontainer Amazon ECS Linux
- Mengelola instans penampung Amazon ECS dari jarak jauh menggunakan AWS Systems Manager
- Menggunakan proxy HTTP untuk instans penampung Amazon ECS Linux
- Mengonfigurasi instans pra-inisialisasi untuk grup Auto Scaling Amazon ECS Anda
- Memperbarui agen kontainer Amazon ECS

Setiap versi agen penampung Amazon ECS mendukung set fitur yang berbeda dan menyediakan perbaikan bug dari versi sebelumnya. Jika memungkinkan, kami selalu menyarankan untuk

menggunakan versi terbaru dari agen kontainer Amazon ECS. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Untuk melihat fitur dan penyempurnaan mana yang disertakan dengan setiap rilis agen, lihat https://github.com/aws/ amazon-ecs-agent /releases.

> **Important**
>
> Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
> Versi agen Amazon ECS `1.20.0` dan yang lebih baru telah menghentikan dukungan untuk versi Docker yang lebih lama dari. `18.01.0`

Meluncurkan instans penampung Amazon ECS Linux

Anda dapat membuat instans penampung Amazon ECS menggunakan konsol Amazon EC2 .

Anda dapat meluncurkan instance dengan berbagai metode termasuk EC2 konsol Amazon, AWS CLI, dan SDK. Untuk informasi tentang metode lain untuk meluncurkan instance, lihat Meluncurkan instance Anda di Panduan EC2 Pengguna Amazon.

Untuk informasi selengkapnya tentang panduan peluncuran, lihat Meluncurkan instance menggunakan wizard instans peluncuran baru di Panduan EC2 Pengguna Amazon.

Sebelum memulai, selesaikan langkah-langkah di Siapkan untuk menggunakan Amazon ECS.

Anda dapat menggunakan EC2 wizard Amazon baru untuk meluncurkan instance. Wizard peluncuran instans menentukan parameter peluncuran yang diperlukan untuk meluncurkan sebuah instans.

Parameter untuk konfigurasi instans

- Prosedur
- Nama dan tanda
- Aplikasi dan Gambar OS (Gambar Mesin Amazon)
- Jenis instans
- Pasangan kunci (login)
- Pengaturan jaringan

- Mengonfigurasi penyimpanan
- Detail lanjutan

Prosedur

Sebelum memulai, selesaikan langkah-langkah di Siapkan untuk menggunakan Amazon ECS.

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Di bilah navigasi di bagian atas layar, AWS Wilayah saat ini ditampilkan (misalnya, AS Timur (Ohio)). Pilih Wilayah tempat instans akan diluncurkan.
3. Dari dasbor EC2 konsol Amazon, pilih Launch instance.

Nama dan tanda

Nama instans adalah tanda, di mana kuncinya adalah Name, dan nilainya adalah nama yang Anda tentukan. Anda dapat menandai instance, volume, dan grafik elastis. Untuk Instans Spot, Anda hanya dapat menandai permintaan Instans Spot.

Menentukan nama instans dan tanda tambahan bersifat opsional.

- Untuk Nama, masukkan nama deskriptif untuk instans tersebut. Jika Anda tidak menentukan nama, instans dapat diidentifikasi berdasarkan ID-nya, yang secara otomatis dihasilkan saat Anda meluncurkan instans tersebut.
- Untuk menambahkan tanda tambahan, pilih Tambahkan tanda tambahan. Pilih Tambahkan tanda, lalu masukkan kunci dan nilai, lalu pilih jenis sumber daya yang akan diberi tanda. Pilih Tambah tanda lagi untuk setiap tanda tambahan yang akan ditambahkan.

Aplikasi dan Gambar OS (Gambar Mesin Amazon)

Amazon Machine Image (AMI) berisi informasi yang diperlukan untuk membuat instans. Misalnya, AMI mungkin berisi perangkat lunak yang diperlukan untuk bertindak sebagai server web, seperti Apache, dan situs web Anda.

Gunakan bilah Pencarian untuk menemukan AMI yang dioptimalkan Amazon ECS yang dioptimalkan yang diterbitkan oleh. AWS

1. Masukkan salah satu istilah berikut di bilah Pencarian.
   - **`ami-ecs`**

- Nilai AMI Amazon ECS yang dioptimalkan.

  Untuk Amazon ECS terbaru yang dioptimalkan AMIs dan nilainya, lihat AMI yang dioptimalkan oleh Linux Amazon ECS.

2. Tekan Enter.
3. Pada halaman Pilih Gambar Mesin Amazon (AMI), pilih AMIs tab AWS Marketplace.
4. Dari panel hasil Refine kiri, pilih Amazon Web Services sebagai Publisher.
5. Pilih Pilih pada baris AMI yang ingin Anda gunakan.

   Atau, pilih Batal (di kanan atas) untuk kembali ke wizard instance peluncuran tanpa memilih AMI. AMI default akan dipilih. Pastikan AMI memenuhi persyaratan yang diuraikan di Linux yang dioptimalkan Amazon ECS. AMIs

Jenis instans

Tipe instans mendefinisikan konfigurasi perangkat keras dan ukuran instans. Tipe instans yang lebih besar memiliki lebih banyak CPU dan memori. Untuk informasi selengkapnya, lihat Tipe instans.

- Untuk Tipe instans, pilih tipe instans untuk instans tersebut.

  Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

Pasangan kunci (login)

Untuk Nama pasangan kunci, pilih pasangan kunci yang ada, atau pilih Buat pasangan kunci baru untuk membuat yang baru.

> **Important**
>
> Jika Anda memilih opsi Lanjutkan tanpa pasangan kunci (Tidak direkomendasikan), Anda tidak akan dapat terhubung ke instans tersebut, kecuali Anda memilih sebuah AMI yang dikonfigurasi agar pengguna dapat masuk dengan cara lain.

Pengaturan jaringan

Konfigurasikan pengaturan jaringan, sesuai keperluan.

- Platform jaringan: Pilih Virtual Private Cloud (VPC), lalu tentukan subnet di bagian Network interface.
- VPC: Pilih VPC yang ada untuk membuat grup keamanan.
- Subnet: Anda dapat meluncurkan sebuah instans di subnet yang terkait dengan Zona Ketersediaan, Local Zone, Wavelength Zone, atau Outpost.

  Untuk meluncurkan instans di Zona Ketersediaan, pilih subnet tempat Anda akan meluncurkan instans. Untuk membuat subnet baru, pilih Buat subnet baru untuk membuka konsol Amazon VPC. Setelah selesai, kembali ke wizard peluncuran instans dan pilih ikon Segarkan untuk memuat subnet Anda dalam daftar.

  Untuk meluncurkan instans di Local Zone, pilih subnet yang Anda buat di Local Zone.

  Untuk meluncurkan sebuah instans di Outpost, pilih subnet di VPC yang Anda kaitkan dengan Outpost.
- Auto-assign IP Publik: Jika instans Anda harus dapat diakses dari internet, verifikasi bahwa bidang Auto-assign Public IP diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

  > **Note**
  >
  > Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini bisa melalui titik akhir VPC antarmuka atau melalui instance container Anda yang memiliki alamat IP publik.
  > Untuk informasi selengkapnya tentang titik akhir VPC antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink
  > Jika Anda tidak memiliki antarmuka VPC endpoint yang dikonfigurasi dan instance container Anda tidak memiliki alamat IP publik, maka mereka harus menggunakan terjemahan alamat jaringan (NAT) untuk menyediakan akses ini. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC dan dalam panduan ini. Menggunakan proxy HTTP untuk instans penampung Amazon ECS Linux
- Firewall (grup keamanan): Gunakan grup keamanan untuk menentukan aturan firewall untuk instance container Anda. Aturan ini menentukan lalu lintas jaringan masuk yang dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan.
  - Untuk memilih grup keamanan yang ada, pilih Pilih grup keamanan yang ada, dan pilih grup keamanan yang Anda buatSiapkan untuk menggunakan Amazon ECS.

Mengonfigurasi penyimpanan

AMI yang Anda pilih mencakup satu atau lebih volume penyimpanan, termasuk volume root. Anda dapat menentukan volume tambahan untuk dilampirkan ke instans.

Anda dapat menggunakan tampilan Sederhana.

- Jenis penyimpanan: Konfigurasikan penyimpanan untuk instance kontainer Anda.

  Jika Anda menggunakan Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memiliki satu volume 30 GiB yang dikonfigurasi, yang dibagi antara sistem operasi dan Docker.

  Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memiliki dua volume yang dikonfigurasi. Volume Root adalah untuk penggunaan sistem operasi, dan volume Amazon EBS kedua (dilampirkan ke`/dev/xvdcz`) adalah untuk penggunaan Docker.

  Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

Detail lanjutan

Untuk Detail lanjutan, perluas bagian untuk melihat kolom dan menentukan parameter tambahan apa pun untuk instans.

- Opsi pembelian: Pilih Minta Instans Spot untuk meminta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

  > **Note**
  >
  > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

  .

- Profil instans IAM: Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

> **Important**
>
> Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.

- Data pengguna: Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data EC2 pengguna Amazon dijalankan hanya satu kali, saat instance pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:
  - Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, tempelkan skrip berikut ke bidang Data pengguna, ganti *your_cluster_name* dengan nama cluster Anda.

```
#!/bin/bash
echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
```

  - Jika Anda memiliki **ecs.config** file di Amazon S3 dan telah mengaktifkan akses hanya-baca Amazon S3 ke peran instance container, pilih daftar Detail Lanjutan. Kemudian, tempelkan skrip berikut ke bidang Data pengguna, ganti *your_bucket_name* dengan nama bucket Anda untuk menginstal AWS CLI dan menulis file konfigurasi Anda pada waktu peluncuran.

> **Note**
>
> Untuk informasi lebih lanjut tentang konfigurasi ini, lihat Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3.

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

  - Tentukan tanda untuk instans kontainer Anda menggunakan parameter konfigurasi `ECS_CONTAINER_INSTANCE_TAGS`. Ini membuat tag yang terkait dengan Amazon ECS saja, mereka tidak dapat dicantumkan menggunakan Amazon EC2 API.

> **Important**
>
> Jika Anda meluncurkan instance container menggunakan grup EC2 Auto Scaling Amazon, Anda harus menggunakan parameter konfigurasi agen ECS_CONTAINER_INSTANCE_TAGS untuk menambahkan tag. Ini karena cara tag ditambahkan ke EC2 instans Amazon yang diluncurkan menggunakan grup Auto Scaling.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- Tentukan tag untuk instance container Anda, lalu gunakan parameter `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` konfigurasi untuk menyebarkannya dari Amazon EC2 ke Amazon ECS

  Berikut adalah instans dari skrip data pengguna yang akan menyebarkan tanda yang terkait dengan instans kontainer, serta mendaftarkan instans kontainer dengan klaster bernama `your_cluster_name`:

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=your_cluster_name
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

Untuk informasi selengkapnya, lihat Bootstrapping instance container Amazon ECS Linux untuk meneruskan data.

Bootstrapping instance container Amazon ECS Linux untuk meneruskan data

Saat meluncurkan EC2 instans Amazon, Anda dapat meneruskan data pengguna ke EC2 instans. Data dapat digunakan untuk melakukan tugas konfigurasi otomatis umum dan bahkan menjalankan skrip ketika boot instans. Untuk Amazon ECS, kasus penggunaan yang paling umum untuk data

pengguna adalah meneruskan informasi konfigurasi ke daemon Docker dan agen penampung Amazon ECS.

Anda dapat meneruskan beberapa jenis data pengguna ke Amazon EC2, termasuk boothook cloud, skrip shell, dan arahan. `cloud-init` Untuk informasi selengkapnya tentang hal ini dan tipe format lainnya, lihat dokumentasi Cloud-init.

Untuk meneruskan data pengguna saat menggunakan wizard EC2 peluncuran Amazon, lihatMeluncurkan instans penampung Amazon ECS Linux.

Anda dapat mengonfigurasi instance kontainer untuk meneruskan data dalam konfigurasi agen kontainer atau dalam konfigurasi daemon Docker.

Agen kontainer Amazon ECS

Varian Linux dari AMI Amazon ECS yang dioptimalkan mencari data konfigurasi agen dalam `/etc/ecs/ecs.config` file saat agen penampung dimulai. Anda dapat menentukan data konfigurasi ini saat diluncurkan dengan data EC2 pengguna Amazon. Untuk informasi selengkapnya tentang variabel konfigurasi agen penampung Amazon ECS yang tersedia, lihatKonfigurasi agen kontainer Amazon ECS.

Untuk mengatur hanya pada variabel konfigurasi agen tunggal, seperti nama klaster, gunakan echo untuk menyalin variabel ke file konfigurasi:

```
#!/bin/bash
echo "ECS_CLUSTER=MyCluster" >> /etc/ecs/ecs.config
```

Jika Anda memiliki beberapa variabel untuk ditulis pada `/etc/ecs/ecs.config`, gunakan pilihan berikut ini pada format `heredoc`. Format ini menulis segalanya di antara baris dimulai dengan cat dan EOF pada file konfigurasi.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
ECS_LOGLEVEL=debug
ECS_WARM_POOLS_CHECK=true
EOF
```

Untuk mengatur atribut instance kustom, atur variabel `ECS_INSTANCE_ATTRIBUTES` lingkungan.

```
#!/bin/bash
cat <<'EOF' >> ecs.config
ECS_INSTANCE_ATTRIBUTES={"envtype":"prod"}
EOF
```

Daemon docker

Anda dapat menentukan informasi konfigurasi daemon Docker dengan data pengguna Amazon EC2 . Untuk informasi selengkapnya tentang pilihan konfigurasi, lihat dokumentasi daemon Docker.

> **Note**
>
> AWS tidak mendukung konfigurasi Docker khusus, karena terkadang dapat bertentangan dengan perubahan atau fitur Amazon ECS di masa mendatang tanpa peringatan.

Pada contoh di bawah ini, pilihan kustom ditambahkan ke file konfigurasi daemon Docker, `/etc/docker/daemon.json` yang kemudian ditentukan dalam data pengguna saat instans diluncurkan.

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"debug": true}
EOF
systemctl restart docker --no-block
```

Pada contoh di bawah ini, pilihan kustom ditambahkan ke file konfigurasi daemon Docker, `/etc/docker/daemon.json` yang kemudian ditentukan dalam data pengguna saat instans diluncurkan. Contoh ini menunjukkan cara menonaktifkan docker-proxy di file konfigurasi daemon Docker.

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"userland-proxy": false}
EOF
systemctl restart docker --no-block
```

Mengonfigurasi instans container Amazon ECS Linux untuk menerima pemberitahuan Instans Spot

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon

EC2 menyediakan pemberitahuan interupsi dua menit Instans Spot untuk menghentikan dan menghentikan tindakan. Itu tidak memberikan pemberitahuan dua menit untuk tindakan hibernasi. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans, Amazon ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instans dalam status `DRAINING`

> **Important**
>
> Amazon ECS tidak menerima pemberitahuan dari Amazon EC2 saat instans dihapus oleh Auto Scaling Capacity Rebalancing. Untuk informasi selengkapnya, lihat Penyeimbangan Kembali Kapasitas EC2 Auto Scaling Amazon.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di status `PENDING` segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Pengurasan Instans Spot dimatikan secara default.

Anda dapat mengaktifkan pengurasan Instance Spot saat meluncurkan instance. Tambahkan skrip berikut ke dalam bidang data Pengguna. Ganti *`MyCluster`* dengan nama cluster untuk mendaftarkan instance container ke.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Untuk mengaktifkan pengurasan Instance Spot untuk instance kontainer yang ada

1. Connect ke Instans Spot melalui SSH.
2. Edit file `/etc/ecs/ecs.config` dan tambahkan berikut:

   ```
   ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
   ```

3. Mulai ulang layanan `ecs`.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

```
sudo systemctl restart ecs
```

4. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan melakukan kueri pada operasi API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

```
curl http://localhost:51678/v1/metadata
```

Menjalankan skrip saat Anda meluncurkan instans penampung Amazon ECS Linux

Anda mungkin perlu menjalankan kontainer tertentu pada setiap instance kontainer untuk menangani masalah operasi atau keamanan seperti pemantauan, keamanan, metrik, penemuan layanan, atau pencatatan.

Untuk melakukannya, Anda dapat mengonfigurasi instans kontainer Anda untuk memanggil perintah docker run dengan skrip data pengguna saat peluncuran, atau dalam beberapa sistem unit seperti Upstart atau systemd. Meskipun metode ini berfungsi, ia memiliki beberapa kelemahan karena Amazon ECS tidak memiliki pengetahuan tentang wadah dan tidak dapat memantau CPU, memori, port, atau sumber daya lain yang digunakan. Untuk memastikan Amazon ECS dapat memperhitungkan semua sumber daya tugas dengan benar, buat definisi tugas agar container dapat dijalankan pada instance container Anda. Kemudian, gunakan Amazon ECS untuk menempatkan tugas pada waktu peluncuran dengan data EC2 pengguna Amazon.

Skrip data EC2 pengguna Amazon dalam prosedur berikut menggunakan API introspeksi Amazon ECS untuk mengidentifikasi instance container. Kemudian, ia menggunakan AWS CLI dan start-task perintah untuk menjalankan tugas tertentu pada dirinya sendiri selama startup.

Untuk memulai tugas pada waktu peluncuran instans kontainer

1. Modifikasi `ecsInstanceRole` IAM role Anda untuk menambahkan izin untuk Operasi API `StartTask` . Untuk informasi selengkapnya, lihat Memperbarui izin untuk peran dalam Panduan AWS Identity and Access Management Pengguna.
2. Luncurkan satu atau beberapa instans kontainer menggunakan Amazon Linux 2 AMI Amazon ECS yang dioptimalkan Amazon. Luncurkan instance container baru dan gunakan contoh script berikut dalam data EC2 User. Ganti *`your_cluster_name`* dengan cluster untuk instance

container untuk mendaftar ke dan *my_task_def* dengan definisi tugas untuk dijalankan pada instance saat peluncuran.

Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

> **Note**
>
> Konten multibagian MIME di bawah ini menggunakan skrip shell untuk menetapkan nilai konfigurasi dan menginstal paket. Hal ini juga menggunakan tugas sistem untuk memulai tugas setelah layanan ecs berjalan dan API introspeksi tersedia.

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Specify the cluster that the container instance should register into
cluster=your_cluster_name

# Write the cluster configuration variable to the ecs.config file
# (add any other configuration variables here also)
echo ECS_CLUSTER=$cluster >> /etc/ecs/ecs.config

START_TASK_SCRIPT_FILE="/etc/ecs/ecs-start-task.sh"
cat <<- 'EOF' > ${START_TASK_SCRIPT_FILE}
 exec 2>>/var/log/ecs/ecs-start-task.log
 set -x

 # Install prerequisite tools
 yum install -y jq aws-cli

 # Wait for the ECS service to be responsive
 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the container instance ARN and AWS Region from instance metadata
```

```
 instance_arn=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $NF}' )
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster' | awk
 -F/ '{print $NF}' )
 region=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F: '{print $4}')

 # Specify the task definition to run at launch
 task_definition=my_task_def

 # Run the AWS CLI start-task command to start your task on this container instance
 aws ecs start-task --cluster $cluster --task-definition $task_definition --
container-instances $instance_arn --started-by $instance_arn --region $region
EOF

# Write systemd unit file
UNIT="ecs-start-task.service"
cat <<- EOF > /etc/systemd/system/${UNIT}
      [Unit]
      Description=ECS Start Task
      Requires=ecs.service
      After=ecs.service

      [Service]
      Restart=on-failure
      RestartSec=30
      ExecStart=/usr/bin/bash ${START_TASK_SCRIPT_FILE}

      [Install]
      WantedBy=default.target
EOF

# Enable our ecs.service dependent service with `--no-block` to prevent systemd
 deadlock
# See https://github.com/aws/amazon-ecs-agent/issues/1707
systemctl enable --now --no-block "${UNIT}"
----==BOUNDARY==--
```

3. Verifikasi bahwa instans kontainer Anda meluncurkan ke klaster yang benar dan bahwa tugas Anda telah dimulai.

   a. Buka konsol di https://console.aws.amazon.com/ecs/v2.

   b. Dari bilah navigasi, pilih Wilayah tempat klaster Anda berada.

c. Di panel navigasi, pilih Klaster dan pilih klaster yang meng-host instans kontainer Anda.

d. Pada halaman Cluster, pilih Tasks, lalu pilih yor task.

   Setiap instance container yang Anda luncurkan harus menjalankan tugas Anda di atasnya.

   Jika Anda tidak melihat tugas Anda, Anda dapat masuk ke instans kontainer Anda dengan SSH dan memeriksa file `/var/log/ecs/ecs-start-task.log` untuk informasi debugging.

Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux

> **Note**
>
> Fitur ini tidak tersedia di Fargate.

Setiap tugas yang menggunakan mode `awsvpc` jaringan menerima elastic network interface (ENI) sendiri, yang dilampirkan ke instance container yang menghostingnya. Ada batas default untuk jumlah antarmuka jaringan yang dapat dilampirkan ke EC2 instance Amazon, dan antarmuka jaringan utama dihitung sebagai satu. Misalnya, secara default sebuah `c5.large` instance mungkin memiliki hingga tiga yang ENIs melekat padanya. Antarmuka jaringan utama untuk instance dihitung sebagai satu, sehingga Anda dapat melampirkan dua tambahan ENIs ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukanENI, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini.

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan ENI kepadatan menggunakan jenis EC2 instans Amazon yang didukung. Saat Anda menggunakan jenis instans ini dan mengaktifkan setelan `awsvpcTrunking` akun, tambahan ENIs tersedia pada instance container yang baru diluncurkan. Konfigurasi ini memungkinkan Anda untuk menempatkan lebih banyak tugas pada setiap instance kontainer. Untuk menggunakan konsol untuk mengaktifkan fitur, lihatMemodifikasi pengaturan akun Amazon ECS. Untuk menggunakan fitur AWS CLI untuk mengaktifkan fitur, lihatMengelola setelan akun Amazon ECS menggunakan AWS CLI.

Misalnya, `c5.large` contoh dengan `awsvpcTrunking` memiliki ENI batas peningkatan dua belas. Instance kontainer akan memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan “trunk” ke instance container. Jadi, konfigurasi ini mengizinkan Anda meluncurkan sepuluh tugas pada instans kontainer, tidak hanya dua tugas seperti saat ini.

Antarmuka jaringan trunk dikelola sepenuhnya oleh Amazon ECS dan dihapus saat Anda menghentikan atau membatalkan pendaftaran instance container dari cluster. Untuk informasi selengkapnya, lihat Opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan fitur ENI trunking.

- Hanya varian Linux dari AMI Amazon ECS yang dioptimalkan, atau varian Amazon Linux lainnya dengan versi `1.28.1` atau yang lebih baru dari agen kontainer dan versi `1.28.1-2` atau yang lebih baru dari paket ecs-init, yang mendukung peningkatan batas. ENI Jika Anda menggunakan varian Linux terbaru dari AMI Amazon ECS yang dioptimalkan, persyaratan ini akan dipenuhi. Kontainer Windows tidak didukung saat ini.
- Hanya EC2 instans Amazon baru yang diluncurkan setelah mengaktifkan `awsvpcTrunking` menerima peningkatan ENI batas dan antarmuka jaringan trunk. Instans yang diluncurkan sebelumnya tidak menerima fitur ini tanpa memperhatikan tindakan yang dilakukan.
- EC2 Instans Amazon harus menonaktifkan permintaan DNS berbasis sumber daya IPv4 . Untuk menonaktifkan opsi ini, hapus opsi Aktifkan permintaan DNS berbasis sumber daya IPV4 (Catatan) saat Anda membuat instance baru di konsol Amazon. EC2 Untuk menonaktifkan opsi ini menggunakan AWS CLI, gunakan perintah berikut.

  ```
  aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
  ```

- EC2 Instans Amazon di subnet bersama tidak didukung. Instans tersebut akan gagal untuk didaftarkan pada klaster jika instans digunakan.
- Tugas Anda harus menggunakan mode `awsvpc` jaringan dan jenis EC2 peluncuran. Tugas yang menggunakan tipe peluncuran Fargate selalu menerima dedicated ENI terlepas dari berapa banyak yang diluncurkan, jadi fitur ini tidak diperlukan.
- Tugas Anda harus diluncurkan dalam VPC Amazon yang sama dengan instance container Anda. Tugas Anda akan gagal untuk dimulai bersama dengan sebuah kesalahan pada atribut jika tugas tersebut tidak berada dalam VPC yang sama.
- Ketika meluncurkan sebuah instans kontainer yang baru, transisi instans menuju status `REGISTERING` sementara antarmuka jaringan elastis torso telah disediakan untuk instans. Jika pendaftaran gagal, transisi instans menuju status `REGISTRATION_FAILED`. Anda dapat memecahkan masalah pada gagalnya pendaftaran dengan menjelaskan instans kontainer untuk melihat bidang `statusReason` yang menjelaskan alasan pada kegagalan. instans kontainer

kemudian secara manual dapat membatalkan pendaftaran atau mengakhirinya. Setelah instance kontainer berhasil dideregistrasi atau dihentikan, Amazon ECS akan menghapus bagasi. ENI

> **Note**
>
> Amazon ECS memancarkan peristiwa perubahan status instans kontainer yang dapat Anda pantau untuk instance yang bertransisi ke status. `REGISTRATION_FAILED` Untuk informasi selengkapnya, lihat Acara perubahan status instans penampung Amazon ECS.

- Setelah instans kontainer dihentikan, transisi instans menuju status `DEREGISTERING` sementara torso pada antarmuka jaringan elastis telah dicabut. Instans tersebut kemudian melakukan transisi menuju status `INACTIVE`.
- Jika instance kontainer di subnet publik dengan ENI batas yang meningkat dihentikan dan kemudian dimulai ulang, instance kehilangan alamat IP publiknya, dan agen kontainer kehilangan koneksinya.
- Saat Anda mengaktifkan`awsvpcTrunking`, instance container menerima tambahan ENI yang menggunakan grup keamanan default VPC, dan dikelola oleh Amazon ECS.

  VPC default dilengkapi dengan subnet publik di setiap Availability Zone, gateway internet, dan pengaturan untuk mengaktifkan resolusi DNS. Subnet adalah subnet publik, karena tabel rute utama mengirimkan lalu lintas subnet yang ditujukan untuk internet ke gateway internet. Anda dapat membuat subnet default ke subnet privat dengan menghapus rute dari tujuan 0.0.0.0/0 ke gateway internet. Namun, jika Anda melakukan ini, tidak ada instance kontainer yang berjalan di subnet itu yang dapat mengakses internet. Anda dapat menambahkan atau menghapus aturan grup keamanan untuk mengontrol lalu lintas masuk dan keluar dari subnet Anda. Untuk informasi selengkapnya, lihat Aturan grup keamanan di Panduan Pengguna Amazon Virtual Private Cloud.

Prasyarat

Sebelum Anda meluncurkan instance kontainer dengan ENI batas yang meningkat, prasyarat berikut harus diselesaikan.

- Peran terkait layanan untuk Amazon ECS harus dibuat. Peran terkait layanan Amazon ECS memberi Amazon ECS izin untuk melakukan panggilan ke layanan lain AWS atas nama Anda. Peran ini dibuat untuk Anda secara otomatis ketika Anda membuat sebuah klaster, atau jika Anda membuat atau memperbarui layanan di AWS Management Console. Untuk informasi

selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS. Anda juga dapat membuat peran terkait layanan dengan perintah berikut AWS CLI .

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- Peran IAM instans akun atau kontainer Anda harus mengaktifkan pengaturan `awsvpcTrunking` akun. Kami menyarankan Anda membuat 2 container instance role (`ecsInstanceRole`). Anda kemudian dapat mengaktifkan pengaturan `awsvpcTrunking` akun untuk satu peran dan menggunakan peran tersebut untuk tugas yang memerlukan trunking ENI. Untuk informasi tentang peran instance container, lihatPeran IAM instance wadah Amazon ECS.

Setelah prasyarat terpenuhi, Anda dapat meluncurkan instance container baru menggunakan salah satu jenis instans Amazon yang didukung, dan EC2 instance akan memiliki batas yang ditingkatkan. ENI Untuk daftar tipe data yang didukung, lihat Instans yang didukung untuk peningkatan antarmuka jaringan kontainer Amazon ECS. instans kontainer harus memiliki versi `1.28.1` atau yang lebih baru dari agen kontainer dan versi `1.28.1-2` atau yang lebih baru dari paket ecs-init. Jika Anda menggunakan varian Linux terbaru dari AMI Amazon ECS yang dioptimalkan, persyaratan ini akan dipenuhi. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Important

EC2 Instans Amazon harus menonaktifkan permintaan DNS berbasis sumber daya IPv4 . Untuk menonaktifkan opsi ini, pastikan opsi Aktifkan permintaan DNS berbasis sumber daya IPV4 (Catatan) tidak dipilih saat membuat instance baru menggunakan konsol Amazon. EC2 Untuk menonaktifkan opsi ini menggunakan AWS CLI, gunakan perintah berikut.

```
aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-
resource-name-dns-a-record --no-dry-run
```

Untuk melihat instance penampung Anda dengan ENI batas yang ditingkatkan dengan AWS CLI

Setiap instans kontainer memiliki antarmuka jaringan default, yang disebut sebagai antarmuka jaringan torso. Gunakan perintah berikut untuk membuat daftar instance kontainer Anda dengan ENI batas yang ditingkatkan dengan menanyakan `ecs.awsvpc-trunk-id` atribut, yang menunjukkan bahwa ia memiliki antarmuka jaringan trunk.

- daftar-atribut (AWS CLI)

```
aws ecs list-attributes \
      --target-type container-instance \
      --attribute-name ecs.awsvpc-trunk-id \
      --cluster cluster_name \
      --region us-east-1
```

- Dapatkan- ECSAttribute Daftar (AWS Tools for Windows PowerShell)

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-id -Region us-east-1
```

Instans yang didukung untuk peningkatan antarmuka jaringan kontainer Amazon ECS

Berikut ini menunjukkan jenis EC2 instans Amazon yang didukung dan berapa banyak tugas yang menggunakan mode `awsvpc` jaringan yang dapat diluncurkan pada setiap jenis instans sebelum dan sesudah mengaktifkan pengaturan `awsvpcTrunking` akun.

> **Important**
>
> Meskipun jenis instance lain didukung dalam keluarga instance yang sama, tipe `a1.metalc5.metal`,`c5a.8xlarge`,`c5ad.8xlarge`, `c5d.metalm5.metal`,`p3dn.24xlarge`,`r5.metal`,`r5.8xlarge`,, dan `r5d.metal` instance tidak didukung.
> Keluarga`c5n`,`d3`, `d3eng3`,`g3s`,`g4dn`,`i3`,`i3en`,`inf1`,`m5dn`,`m5n`, `m5znmac1`,`r5b`,`r5n`,`r5dn`,`u-12tb1`,`u-6tb1`,`u-9tb1`,, dan `z1d` contoh tidak didukung.

Topik

- Tujuan umum
- Komputasi yang dioptimalkan
- Memori yang dioptimalkan
- Penyimpanan yang dioptimalkan
- Komputasi yang dipercepat
- Komputasi performa tinggi

Tujuan umum

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20
a1.2xlarge	3	40
a1.4xlarge	7	60
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.8xlarge	7	60
m5.12xlarge	7	60
m5.16xlarge	14	120
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40
m5a.4xlarge	7	60
m5a.8xlarge	7	60
m5a.12xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m5a.16xlarge	14	120
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.8xlarge	7	60
m5ad.12xlarge	7	60
m5ad.16xlarge	14	120
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.8xlarge	7	60
m5d.12xlarge	7	60
m5d.16xlarge	14	120
m5d.24xlarge	14	120
m5d.metal	14	120
m6a.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6a.xlarge	3	20
m6a.2xlarge	3	40
m6a.4xlarge	7	60
m6a.8xlarge	7	90
m6a.12xlarge	7	120
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120
m6a.48xlarge	14	120
m6a.metal	14	120
m6g.medium	1	4
m6g.large	2	10
m6g.xlarge	3	20
m6g.2xlarge	3	40
m6g.4xlarge	7	60
m6g.8xlarge	7	60
m6g.12xlarge	7	60
m6g.16xlarge	14	120
m6g.metal	14	120
m6gd.medium	1	4

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6gd.large	2	10
m6gd.xlarge	3	20
m6gd.2xlarge	3	40
m6gd.4xlarge	7	60
m6gd.8xlarge	7	60
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120
m6i.large	2	10
m6i.xlarge	3	20
m6i.2xlarge	3	40
m6i.4xlarge	7	60
m6i.8xlarge	7	90
m6i.12xlarge	7	120
m6i.16xlarge	14	120
m6i.24xlarge	14	120
m6i.32xlarge	14	120
m6i.metal	14	120
m6id.large	2	10
m6id.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6id.2xlarge	3	40
m6id.4xlarge	7	60
m6id.8xlarge	7	90
m6id.12xlarge	7	120
m6id.16xlarge	14	120
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120
m6idn.large	2	10
m6idn.xlarge	3	20
m6idn.2xlarge	3	40
m6idn.4xlarge	7	60
m6idn.8xlarge	7	90
m6idn.12xlarge	7	120
m6idn.16xlarge	14	120
m6idn.24xlarge	14	120
m6idn.32xlarge	15	120
m6idn.metal	15	120
m6in.large	2	10
m6in.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m6in.2xlarge	3	40
m6in.4xlarge	7	60
m6in.8xlarge	7	90
m6in.12xlarge	7	120
m6in.16xlarge	14	120
m6in.24xlarge	14	120
m6in.32xlarge	15	120
m6in.metal	15	120
m7a.medium	1	4
m7a.large	2	10
m7a.xlarge	3	20
m7a.2xlarge	3	40
m7a.4xlarge	7	60
m7a.8xlarge	7	90
m7a.12xlarge	7	120
m7a.16xlarge	14	120
m7a.24xlarge	14	120
m7a.32xlarge	14	120
m7a.48xlarge	14	120
m7a.metal-48xl	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m7g.medium	1	4
m7g.large	2	10
m7g.xlarge	3	20
m7g.2xlarge	3	40
m7g.4xlarge	7	60
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120
m7g.metal	14	120
m7gd.medium	1	4
m7gd.large	2	10
m7gd.xlarge	3	20
m7gd.2xlarge	3	40
m7gd.4xlarge	7	60
m7gd.8xlarge	7	60
m7gd.12xlarge	7	60
m7gd.16xlarge	14	120
m7gd.metal	14	120
m7i.large	2	10
m7i.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m7i.2xlarge	3	40
m7i.4xlarge	7	60
m7i.8xlarge	7	90
m7i.12xlarge	7	120
m7i.16xlarge	14	120
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120
m7i.metal-48xl	14	120
m7i-flex.large	2	4
m7i-flex.xlarge	3	10
m7i-flex.2xlarge	3	20
m7i-flex.4xlarge	7	40
m7i-flex.8xlarge	7	60
m7i-flex.12xlarge	7	120
m7i-flex.16xlarge	14	120
m8g.sedang	1	4
m8g.large	2	10
m8g.xlarge	3	20
m8g.2xlarge	3	40

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
m8g.4xlarge	7	60
m8g.8xlarge	7	60
m8g.12xlarge	7	60
m8g.16xlarge	14	120
m8g.24xlarge	14	120
m8g.48xlarge	14	120
m8g.logam-24xl	14	120
m8g.logam-48xl	14	120
m8gd.sedang	1	4
m8gd.large	2	10
m8gd.xlarge	3	20
m8gd.2xlarge	3	40
m8gd.4xlarge	7	60
m8gd.8xlarge	7	60
m8gd.12xlarge	7	60
m8gd.16xlarge	14	120
m8gd.24xlarge	14	120
m8gd.48xlarge	14	120
m8gd.metal-24xl	14	120
m8gd.metal-48xl	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
mac2.metal	7	12
mac2-m1ultra.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12

Komputasi yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60
c5.12xlarge	7	60
c5.18xlarge	14	120
c5.24xlarge	14	120
c5a.large	2	10
c5a.xlarge	3	20
c5a.2xlarge	3	40
c5a.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c5a.12xlarge	7	60
c5a.16xlarge	14	120
c5a.24xlarge	14	120
c5ad.large	2	10
c5ad.xlarge	3	20
c5ad.2xlarge	3	40
c5ad.4xlarge	7	60
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.12xlarge	7	60
c5d.18xlarge	14	120
c5d.24xlarge	14	120
c6a.large	2	10
c6a.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6a.2xlarge	3	40
c6a.4xlarge	7	60
c6a.8xlarge	7	90
c6a.12xlarge	7	120
c6a.16xlarge	14	120
c6a.24xlarge	14	120
c6a.32xlarge	14	120
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40
c6g.4xlarge	7	60
c6g.8xlarge	7	60
c6g.12xlarge	7	60
c6g.16xlarge	14	120
c6g.metal	14	120
c6gd.medium	1	4
c6gd.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6gd.xlarge	3	20
c6gd.2xlarge	3	40
c6gd.4xlarge	7	60
c6gd.8xlarge	7	60
c6gd.12xlarge	7	60
c6gd.16xlarge	14	120
c6gd.metal	14	120
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60
c6gn.12xlarge	7	60
c6gn.16xlarge	14	120
c6i.large	2	10
c6i.xlarge	3	20
c6i.2xlarge	3	40
c6i.4xlarge	7	60
c6i.8xlarge	7	90

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6i.12xlarge	7	120
c6i.16xlarge	14	120
c6i.24xlarge	14	120
c6i.32xlarge	14	120
c6i.metal	14	120
c6id.large	2	10
c6id.xlarge	3	20
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120
c6id.32xlarge	14	120
c6id.metal	14	120
c6in.large	2	10
c6in.xlarge	3	20
c6in.2xlarge	3	40
c6in.4xlarge	7	60
c6in.8xlarge	7	90

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c6in.12xlarge	7	120
c6in.16xlarge	14	120
c6in.24xlarge	14	120
c6in.32xlarge	15	120
c6in.metal	15	120
c7a.medium	1	4
c7a.large	2	10
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120
c7a.24xlarge	14	120
c7a.32xlarge	14	120
c7a.48xlarge	14	120
c7a.metal-48xl	14	120
c7g.medium	1	4
c7g.large	2	10
c7g.xlarge	3	20

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c7g.2xlarge	3	40
c7g.4xlarge	7	60
c7g.8xlarge	7	60
c7g.12xlarge	7	60
c7g.16xlarge	14	120
c7g.metal	14	120
c7gd.medium	1	4
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60
c7gd.16xlarge	14	120
c7gd.metal	14	120
c7gn.medium	1	4
c7gn.large	2	10
c7gn.xlarge	3	20
c7gn.2xlarge	3	40
c7gn.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c7gn.8xlarge	7	60
c7gn.12xlarge	7	60
c7gn.16xlarge	14	120
c7gn.logam	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120
c7i-flex.large	2	4
c7i-flex.xlarge	3	10
c7i-flex.2xlarge	3	20
c7i-flex.4xlarge	7	40
c7i-flex.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c7i-flex.12xlarge	7	120
c7i-flex.16xlarge	14	120
c8g.sedang	1	4
c8g.large	2	10
c8g.xlarge	3	20
c8g.2xlarge	3	40
c8g.4xlarge	7	60
c8g.8xlarge	7	60
c8g.12xlarge	7	60
c8g.16xlarge	14	120
c8g.24xlarge	14	120
c8g.48xlarge	14	120
c8g.logam-24xl	14	120
c8g.logam-48xl	14	120
c8gd.sedang	1	4
c8gd.large	2	10
c8gd.xlarge	3	20
c8gd.2xlarge	3	40
c8gd.4xlarge	7	60
c8gd.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
c8gd.12xlarge	7	60
c8gd.16xlarge	14	120
c8gd.24xlarge	14	120
c8gd.48xlarge	14	120
c8gd.metal-24xl	14	120
c8gd.metal-48xl	14	120
c8gn.sedang	1	4
c8gn.large	2	10
c8gn.xlarge	3	20
c8gn.2xlarge	3	40
c8gn.4xlarge	7	60
c8gn.8xlarge	9	60
c8gn.12xlarge	11	60
c8gn.16xlarge	15	120
c8gn.24xlarge	23	120
c8gn.48xlarge	23	120
c8gn.metal-24xl	23	120
c8gn.metal-48xl	23	120

Memori yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r5.large	2	10
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.16xlarge	14	120
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40
r5a.4xlarge	7	60
r5a.8xlarge	7	60
r5a.12xlarge	7	60
r5a.16xlarge	14	120
r5a.24xlarge	14	120
r5ad.large	2	10
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r5ad.8xlarge	7	60
r5ad.12xlarge	7	60
r5ad.16xlarge	14	120
r5ad.24xlarge	14	120
r5b.16xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.8xlarge	7	60
r5d.12xlarge	7	60
r5d.16xlarge	14	120
r5d.24xlarge	14	120
r5dn.16xlarge	14	120
r6a.large	2	10
r6a.xlarge	3	20
r6a.2xlarge	3	40
r6a.4xlarge	7	60
r6a.8xlarge	7	90
r6a.12xlarge	7	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6a.16xlarge	14	120
r6a.24xlarge	14	120
r6a.32xlarge	14	120
r6a.48xlarge	14	120
r6a.metal	14	120
r6g.medium	1	4
r6g.large	2	10
r6g.xlarge	3	20
r6g.2xlarge	3	40
r6g.4xlarge	7	60
r6g.8xlarge	7	60
r6g.12xlarge	7	60
r6g.16xlarge	14	120
r6g.metal	14	120
r6gd.medium	1	4
r6gd.large	2	10
r6gd.xlarge	3	20
r6gd.2xlarge	3	40
r6gd.4xlarge	7	60
r6gd.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6gd.12xlarge	7	60
r6gd.16xlarge	14	120
r6gd.metal	14	120
r6i.large	2	10
r6i.xlarge	3	20
r6i.2xlarge	3	40
r6i.4xlarge	7	60
r6i.8xlarge	7	90
r6i.12xlarge	7	120
r6i.16xlarge	14	120
r6i.24xlarge	14	120
r6i.32xlarge	14	120
r6i.metal	14	120
r6idn.large	2	10
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120
r6idn.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6idn.24xlarge	14	120
r6idn.32xlarge	15	120
r6idn.metal	15	120
r6in.large	2	10
r6in.xlarge	3	20
r6in.2xlarge	3	40
r6in.4xlarge	7	60
r6in.8xlarge	7	90
r6in.12xlarge	7	120
r6in.16xlarge	14	120
r6in.24xlarge	14	120
r6in.32xlarge	15	120
r6in.metal	15	120
r6id.large	2	10
r6id.xlarge	3	20
r6id.2xlarge	3	40
r6id.4xlarge	7	60
r6id.8xlarge	7	90
r6id.12xlarge	7	120
r6id.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r6id.24xlarge	14	120
r6id.32xlarge	14	120
r6id.metal	14	120
r7a.medium	1	4
r7a.large	2	10
r7a.xlarge	3	20
r7a.2xlarge	3	40
r7a.4xlarge	7	60
r7a.8xlarge	7	90
r7a.12xlarge	7	120
r7a.16xlarge	14	120
r7a.24xlarge	14	120
r7a.32xlarge	14	120
r7a.48xlarge	14	120
r7a.metal-48xl	14	120
r7g.medium	1	4
r7g.large	2	10
r7g.xlarge	3	20
r7g.2xlarge	3	40
r7g.4xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r7g.8xlarge	7	60
r7g.12xlarge	7	60
r7g.16xlarge	14	120
r7g.metal	14	120
r7gd.medium	1	4
r7gd.large	2	10
r7gd.xlarge	3	20
r7gd.2xlarge	3	40
r7gd.4xlarge	7	60
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60
r7gd.16xlarge	14	120
r7gd.logam	14	120
r7i.large	2	10
r7i.xlarge	3	20
r7i.2xlarge	3	40
r7i.4xlarge	7	60
r7i.8xlarge	7	90
r7i.12xlarge	7	120
r7i.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r7i.24xlarge	14	120
r7i.48xlarge	14	120
r7i.metal-24xl	14	120
r7i.metal-48xl	14	120
r7iz.large	2	10
r7iz.xlarge	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120
r8g.sedang	1	4
r8g.besar	2	10
r8g.xlarge	3	20
r8g.2xlarge	3	40
r8g.4xlarge	7	60
r8g.8xlarge	7	60

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
r8g.12xlarge	7	60
r8g.16xlarge	14	120
r8g.24xlarge	14	120
r8g.48xlarge	14	120
r8g.logam-24xl	14	120
r8g.logam-48xl	14	120
r8gd.sedang	1	4
r8gd.large	2	10
r8gd.xlarge	3	20
r8gd.2xlarge	3	40
r8gd.4xlarge	7	60
r8gd.8xlarge	7	60
r8gd.12xlarge	7	60
r8gd.16xlarge	14	120
r8gd.24xlarge	14	120
r8gd.48xlarge	14	120
r8gd.metal-24xl	14	120
r8gd.metal-48xl	14	120
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
u7i-6tb.112xlarge	14	120
u7i-8tb.112xlarge	14	120
u7i-12tb.224xlarge	14	120
u7in-16tb.224xlarge	15	120
u7in-24tb.224xlarge	15	120
u7in-32tb.224xlarge	15	120
u7inh-32tb.480xlarge	15	120
x2gd.medium	1	10
x2gd.large	2	10
x2gd.xlarge	3	20
x2gd.2xlarge	3	40
x2gd.4xlarge	7	60
x2gd.8xlarge	7	60
x2gd.12xlarge	7	60
x2gd.16xlarge	14	120
x2gd.metal	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
x2idn.16xlarge	14	120
x2idn.24xlarge	14	120
x2idn.32xlarge	14	120
x2idn.metal	14	120
x2iedn.xlarge	3	13
x2iedn.2xlarge	3	29
x2iedn.4xlarge	7	60
x2iedn.8xlarge	7	120
x2iedn.16xlarge	14	120
x2iedn.24xlarge	14	120
x2iedn.32xlarge	14	120
x2iedn.metal	14	120
x2iezn.2xlarge	3	64
x2iezn.4xlarge	7	120
x2iezn.6xlarge	7	120
x2iezn.8xlarge	7	120
x2iezn.12xlarge	14	120
x2iezn.metal	14	120
x8g.sedang	1	4
x8g.large	2	10

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
x8g.xlarge	3	20
x8g.2xlarge	3	40
x8g.4xlarge	7	60
x8g.8xlarge	7	60
x8g.12xlarge	7	60
x8g.16xlarge	14	120
x8g.24xlarge	14	120
x8g.48xlarge	14	120
x8g.logam-24xl	14	120
x8g.logam-48xl	14	120

Penyimpanan yang dioptimalkan

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60
i4g.8xlarge	7	60
i4g.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i4i.xlarge	3	8
i4i.2xlarge	3	28
i4i.4xlarge	7	58
i4i.8xlarge	7	118
i4i.12xlarge	7	118
i4i.16xlarge	14	248
i4i.24xlarge	14	118
i4i.32xlarge	14	498
i4i.metal	14	498
i7i.large	2	10
i7i.xlarge	3	20
i7i.2xlarge	3	40
i7i.4xlarge	7	60
i7i.8xlarge	7	90
i7i.12xlarge	7	90
i7i.16xlarge	14	120
i7i.24xlarge	14	120
i7i.48xlarge	14	120
i7i.logam-24xl	14	120
i7i.logam-48xl	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i7ie.large	2	20
i7ie.xlarge	3	29
i7ie.2xlarge	3	29
i7ie.3xlarge	3	29
i7ie.6xlarge	7	60
i7ie.12xlarge	7	60
i7ie.18xlarge	14	120
i7ie.24xlarge	14	120
i7ie.48xlarge	14	120
i7ie.logam-24xl	14	120
i7ie.logam-48xl	14	120
i8g.large	2	10
i8g.xlarge	3	20
i8g.2xlarge	3	40
i8g.4xlarge	7	60
i8g.8xlarge	7	60
i8g.12xlarge	7	60
i8g.16xlarge	14	120
i8g.24xlarge	14	120
i8g.48xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
i8g.logam-24xl	14	120
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40
im4gn.4xlarge	7	60
im4gn.8xlarge	7	60
im4gn.16xlarge	14	120
is4gen.medium	1	4
is4gen.large	2	10
is4gen.xlarge	3	20
is4gen.2xlarge	3	40
is4gen.4xlarge	7	60
is4gen.8xlarge	7	60

Komputasi yang dipercepat

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
f2.6xbesar	7	90

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
f2.12xbesar	7	120
f2.48xbesar	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19
g5.4xlarge	7	40
g5.8xlarge	7	90
g5.12xlarge	14	120
g5.16xlarge	7	120
g5.24xlarge	14	120
g5.48xlarge	6	120
g5g.xlarge	3	20
g5g.2xlarge	3	40
g5g.4xlarge	7	60
g5g.8xlarge	7	60
g5g.16xlarge	14	120

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
g5g.metal	14	120
g6.xlarge	3	20
g6.2xlarge	3	40
g6.4xlarge	7	60
g6.8xlarge	7	90
g6.12xlarge	7	120
g6.16xlarge	14	120
g6.24xlarge	14	120
g6.48xlarge	14	120
g6e.xlarge	3	20
g6e.2xlarge	3	40
g6e.4xlarge	7	60
g6e.8xlarge	7	90
g6e.12xlarge	9	120
g6e.16xlarge	14	120
g6e.24xlarge	19	120
g6e.48xlarge	39	120
g6f.large	1	10
g6f.xlarge	3	20
g6f.2xlarge	3	40

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
g6f.4xlarge	7	60
gr6.4xbesar	7	60
gr6.8xbesar	7	90
gr6f.4xbesar	7	60
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.48xlarge	63	242
p5e.48xlarge	63	242
p5en.48xlarge	63	242
p6-b200.48xlarge	31	242
p6e-gb200.36xlarge	38	120
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242
trn2.48xlarge	31	242
trn2u.48xlarge	31	242

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

Komputasi performa tinggi

Jenis instans	Batas tugas tanpa ENI trunking	Batas tugas dengan ENI trunking
hpc6a.48xlarge	1	120
hpc6id.32xlarge	1	120
hpc7g.4xlarge	3	120
hpc7g.8xlarge	3	120
hpc7g.16xlarge	3	120

Memesan memori instans kontainer Amazon ECS Linux

Saat agen penampung Amazon ECS mendaftarkan instance kontainer ke klaster, agen harus menentukan berapa banyak memori yang tersedia untuk dicadangkan oleh instans penampung untuk tugas Anda. Karena overhead memori platform dan memori yang ditempati oleh kernel sistem, jumlah ini berbeda dari jumlah memori yang diinstal yang diiklankan untuk instans Amazon EC2 . Contohnya, instans `m4.large` memiliki memori terinstal sebesar 8 GiB. Namun, ini tidak selalu diterjemahkan ke tepat 8192 MiB memori yang tersedia untuk tugas saat instance container mendaftar.

Agen penampung Amazon ECS menyediakan variabel konfigurasi yang disebut`ECS_RESERVED_MEMORY`, yang dapat Anda gunakan untuk menghapus sejumlah memori MiB tertentu dari kumpulan yang dialokasikan untuk tugas Anda. Ini secara efektif menyimpan memori untuk proses sistem yang penting.

Jika Anda menempati semua memori pada instance kontainer dengan tugas Anda, maka ada kemungkinan bahwa tugas Anda akan bersaing dengan proses sistem penting untuk memori dan mungkin memulai kegagalan sistem.

Misalnya, jika Anda menentukan `ECS_RESERVED_MEMORY=256` dalam file konfigurasi agen kontainer Anda, agen akan mendaftarkan memori total dikurangi 256 MiB untuk instans tersebut, dan 256 MiB memori tidak dapat dialokasikan untuk tugas ECS. Untuk informasi lebih lanjut tentang variabel konfigurasi agen dan cara mengaturnya, lihat Konfigurasi agen kontainer Amazon ECS dan Bootstrapping instance container Amazon ECS Linux untuk meneruskan data.

Jika Anda menentukan 8192 MiB untuk tugas tersebut, dan tidak ada instance kontainer Anda yang memiliki 8192 MiB atau lebih besar memori yang tersedia untuk memenuhi persyaratan ini, maka tugas tersebut tidak dapat ditempatkan di cluster Anda. Jika Anda menggunakan lingkungan komputasi terkelola, maka AWS Batch harus meluncurkan jenis instans yang lebih besar untuk mengakomodasi permintaan.

Anda juga harus menyimpan beberapa memori untuk agen penampung Amazon ECS dan proses sistem penting lainnya pada instance penampung Anda, sehingga kontainer tugas Anda tidak bersaing untuk memori yang sama dan mungkin memulai kegagalan sistem.

Agen kontainer Amazon ECS menggunakan fungsi `ReadMemInfo()` Docker untuk kueri total memori yang tersedia untuk sistem operasi. Baik Linux dan Windows menyediakan utilitas baris perintah untuk menentukan total memori.

Example - Menentukan memori total Linux

Perintah free menampilkan memori total yang diakui oleh sistem operasi.

```
$ free -b
```

Contoh output untuk instans `m4.large` yang menjalankan AMI Amazon Linux yang dioptimalkan untuk Amazon ECS.

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

Instans ini memiliki 8373026816 byte memori total, yang diterjemahkan menjadi 7985 MiB yang tersedia untuk tugas.

Example - Menentukan memori total Windows

Perintah wmic menampilkan memori total yang diakui oleh sistem operasi.

```
C:\> wmic ComputerSystem get TotalPhysicalMemory
```

Contoh keluaran untuk `m4.large` instance yang menjalankan AMI Windows Server Amazon ECS yang dioptimalkan.

```
TotalPhysicalMemory
8589524992
```

Instans ini memiliki 8589524992 byte memori total, yang diterjemahkan menjadi 8191 MiB yang tersedia untuk tugas.

Melihat memori instance kontainer

Anda dapat melihat berapa banyak memori yang didaftarkan oleh instans kontainer di konsol Amazon ECS (atau dengan operasi DescribeContainerInstancesAPI). Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan tugas Anda sebanyak mungkin memori untuk jenis instance tertentu, Anda dapat mengamati memori yang tersedia untuk instance kontainer itu dan kemudian menetapkan tugas Anda sebanyak itu memori.

Untuk melihat memori instance kontainer

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Clusters, lalu pilih cluster yang menghosting instance container Anda.
3. Pilih Infrastruktur, dan kemudian di bawah Instance Container, pilih instance container.
4. Bagian Resources menunjukkan memori terdaftar dan tersedia untuk instance container.

   Nilai memori Terdaftar adalah contoh kontainer; terdaftar dengan Amazon ECS saat pertama kali diluncurkan, dan nilai memori yang tersedia adalah apa yang belum dialokasikan untuk tugas.

Mengelola instans penampung Amazon ECS dari jarak jauh menggunakan AWS Systems Manager

Anda dapat menggunakan kemampuan Run Command di AWS Systems Manager (Systems Manager) untuk mengelola konfigurasi instans container Amazon ECS secara aman dan jarak jauh. Run Command menyediakan cara sederhana untuk melakukan tugas-tugas administratif umum

tanpa harus masuk secara lokal pada instans. Anda dapat mengelola perubahan konfigurasi di klaster Anda dengan cara menjalankan perintah pada beberapa instans kontainer secara bersamaan. Run Command melaporkan status dan hasil dari setiap perintah.

Berikut adalah beberapa contoh tipe tugas yang dapat Anda lakukan dengan Run Command:

- Menginstal atau menghapus paket.
- Melakukan pembaruan keamanan.
- Membersihkan citra Docker.
- Menghentikan atau memulai layanan.
- Melihat sumber daya sistem.
- Melihat berkas log.
- Melakukan operasi file.

Untuk informasi selengkapnya tentang Jalankan Perintah, lihat AWS Systems Manager Menjalankan Perintah di Panduan AWS Systems Manager Pengguna.

Berikut ini adalah prasyarat untuk menggunakan Systems Manager dengan Amazon ECS.

1. Anda harus memberikan izin container instance role (ecsInstanceRole) untuk mengakses Systems Manager APIs. Anda dapat melakukan ini dengan menetapkan Amazon SSMManaged InstanceCore ke `ecsInstanceRole` peran tersebut. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memperbarui izin untuk peran di AWS Identity and Access Management Panduan Pengguna
2. Pastikan bahwa Agen SSM diinstal pada instans kontainer Anda. Untuk informasi selengkapnya, lihat Menginstal dan menghapus instalan Agen SSM secara manual pada EC2 instans untuk Linux.

Setelah melampirkan kebijakan terkelola Systems Manager ke `ecsInstanceRole` dan memverifikasi bahwa AWS Systems Manager Agen (Agen SSM) diinstal pada instance container, Anda dapat mulai menggunakan Run Command untuk mengirim perintah ke instance container Anda. Untuk informasi tentang menjalankan perintah dan skrip shell pada instance Anda dan melihat output yang dihasilkan, lihat Menjalankan Perintah Menggunakan Systems Manager Run Command and Run Command Walkthroughs di Panduan Pengguna.AWS Systems Manager

Kasus penggunaan yang umum adalah memperbarui perangkat lunak instance kontainer dengan Run Command. Anda dapat mengikuti prosedur di Panduan AWS Systems Manager Pengguna dengan parameter berikut.

Parameter	Nilai
Dokumen perintah	`AWS-RunShellScript`
Perintah	`$ yum update -y`
Contoh target	Instance kontainer Anda

Menggunakan proxy HTTP untuk instans penampung Amazon ECS Linux

Anda dapat mengonfigurasi instans penampung Amazon ECS untuk menggunakan proxy HTTP untuk agen penampung Amazon ECS dan daemon Docker. Ini berguna jika instance container Anda tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instance.

Untuk mengonfigurasi instans penampung Amazon ECS Linux Anda agar menggunakan proxy HTTP, setel variabel berikut dalam file yang relevan pada waktu peluncuran (dengan data EC2 pengguna Amazon). Anda juga dapat mengedit file konfigurasi secara manual, dan kemudian me-restart agen.

`/etc/ecs/ecs.config`(Amazon Linux 2dan AmazonLinux AMI)

`HTTP_PROXY=10.0.0.131:3128`

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan agen Amazon ECS untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata EC2 instance, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/systemd/system/ecs.service.d/http-proxy.conf`(Amazon Linux 2 saja)

`Environment="HTTP_PROXY=10.0.0.131:3128/"`

Atur nilai ini pada nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan `ecs-init` untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock"`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata EC2 instance, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/init/ecs.override`(Hanya AMI Amazon Linux)

`env HTTP_PROXY=10.0.0.131:3128`

Atur nilai ini pada nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan `ecs-init` untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

Tetapkan nilai ini `169.254.169.254,169.254.170.2,/var/run/docker.sock` untuk memfilter metadata EC2 instance, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

`/etc/systemd/system/docker.service.d/http-proxy.conf` (Amazon Linux 2 saja)

`Environment="HTTP_PROXY=http://10.0.0.131:3128"`

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan untuk daemon Docker untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`Environment="NO_PROXY=169.254.169.254,169.254.170.2"`

Tetapkan nilai ini `169.254.169.254,169.254.170.2` untuk memfilter metadata EC2 instance dari proxy.

`/etc/sysconfig/docker`(Amazon Linux AMI dan Amazon Linux 2 hanya)

`export HTTP_PROXY=http://10.0.0.131:3128`

Tetapkan nilai ini ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan untuk daemon Docker untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

`export NO_PROXY=169.254.169.254,169.254.170.2`

Tetapkan nilai ini `169.254.169.254,169.254.170.2` untuk memfilter metadata EC2 instance dari proxy.

Menyetel variabel lingkungan ini dalam file di atas hanya memengaruhi agen penampung Amazon ECS`ecs-init`, dan daemon Docker. Mereka tidak mengonfigurasi layanan lain (seperti yum) untuk menggunakan proxy.

Untuk informasi tentang cara mengonfigurasi proxy, lihat Bagaimana cara mengatur proxy HTTP untuk Docker dan agen penampung Amazon ECS di Amazon Linux 2 atau 023. AL2

Mengonfigurasi instans pra-inisialisasi untuk grup Auto Scaling Amazon ECS Anda

Amazon ECS mendukung kolam hangat Amazon EC2 Auto Scaling. Kolam hangat adalah sekelompok EC2 instance Amazon pra-inisialisasi yang siap digunakan. Kapan pun aplikasi Anda perlu diskalakan, Amazon EC2 Auto Scaling menggunakan instance pra-inisialisasi dari kolam hangat daripada meluncurkan instance dingin, memungkinkan proses inisialisasi akhir apa pun berjalan, dan kemudian menempatkan instance ke dalam layanan.

Untuk mempelajari lebih lanjut tentang kolam hangat dan cara menambahkan kolam hangat ke grup Auto Scaling Anda, lihat Kolam hangat untuk Amazon Auto EC2 Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.

Saat membuat atau memperbarui kolam hangat untuk grup Auto Scaling untuk Amazon ECS, Anda tidak dapat menyetel opsi yang mengembalikan instance ke kolam hangat pada skala di (). `ReuseOnScaleIn` Untuk informasi selengkapnya, lihat put-warm-pool dalam Referensi AWS Command Line Interface .

Untuk menggunakan kolam hangat dengan kluster Amazon ECS Anda, setel variabel konfigurasi `ECS_WARM_POOLS_CHECK` agen ke `true` bidang data Pengguna templat peluncuran grup Amazon EC2 Auto Scaling Anda.

Berikut ini menunjukkan contoh bagaimana variabel konfigurasi agen dapat ditentukan di bidang data Pengguna dari template EC2 peluncuran Amazon. Ganti *MyCluster* dengan nama cluster Anda.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_WARM_POOLS_CHECK=true
EOF
```

`ECS_WARM_POOLS_CHECK`Variabel hanya didukung pada versi agen `1.59.0` dan yang lebih baru. Untuk informasi lebih lanjut tentang variabel, lihatKonfigurasi agen kontainer Amazon ECS.

Memperbarui agen kontainer Amazon ECS

Terkadang, Anda mungkin perlu memperbarui agen penampung Amazon ECS untuk mengambil perbaikan bug dan fitur baru. Memperbarui agen penampung Amazon ECS tidak mengganggu tugas atau layanan yang sedang berjalan pada instance container. Proses untuk memperbarui agen berbeda-beda tergantung pada apakah instans container Anda diluncurkan dengan AMI yang dioptimalkan Amazon ECS atau sistem operasi lain.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Memeriksa versi agen penampung Amazon ECS

Anda dapat memeriksa versi agen kontainer yang berjalan pada instans kontainer Anda untuk melihat apakah pembaruan diperlukan. Tampilan instance container di konsol Amazon ECS menyediakan versi agen. Gunakan prosedur berikut untuk memeriksa versi agen Anda.

Amazon ECS console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster, kemudian pilih klaster yang meng-host instans eksternal.
4. Pada *name* halaman Cluster:, pilih tab Infrastruktur.

5. Di bawah Instance Container, perhatikan kolom versi Agen untuk instance kontainer Anda. Jika instans kontainer tidak memiliki versi instans kontainer terbaru, konsol akan memberi peringatan melalui pesan dan bendera bahwa versi agen kedaluwarsa.

   Jika versi agen Anda kedaluwarsa, Anda dapat memperbarui agen kontainer Anda dengan prosedur berikut:

   - Jika instance container Anda menjalankan AMI Amazon ECS yang dioptimalkan, lihat. Memperbarui agen kontainer Amazon ECS pada AMI Amazon ECS yang dioptimalkan
   - Jika instance container Anda tidak menjalankan AMI yang dioptimalkan Amazon ECS, lihat. Memperbarui agen penampung Amazon ECS secara manual (untuk non-Amazon ECS yang dioptimalkan) AMIs

   > **Important**
   >
   > Untuk memperbarui versi agen Amazon ECS dari versi sebelum v1.0.0 pada AMI yang dioptimalkan Amazon ECS, sebaiknya hentikan instans penampung saat ini dan meluncurkan instans baru dengan versi AMI terbaru. Setiap instans kontainer yang menggunakan versi pratinjau harus pensiun dan diganti dengan AMI terbaru. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Amazon ECS container agent introspection API

Anda juga dapat menggunakan untuk memeriksa versi API introspeksi agen penampung Amazon ECS dari instance container itu sendiri. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.

Untuk memeriksa apakah agen penampung Amazon ECS Anda menjalankan versi terbaru dengan API introspeksi

1. Masuk ke instans kontainer Anda melalui SSH.
2. Kueri API introspeksi.

```
[ec2-user ~]$ curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

> **Note**
>
> API introspeksi menambahkan `Version` informasi dalam versi v1.0.0 dari agen penampung Amazon ECS. Jika `Version` tidak ada ketika sedang melakukan kueri API introspeksi, atau API introspeksi sama sekali tidak ada dalam agen Anda, maka versi yang Anda jalankan adalah v0.0.3 atau sebelumnya. Anda harus memperbarui versi Anda.

Memperbarui agen kontainer Amazon ECS pada AMI Amazon ECS yang dioptimalkan

Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda memiliki beberapa opsi untuk mendapatkan versi terbaru dari agen penampung Amazon ECS (ditampilkan dalam urutan rekomendasi):

- Hentikan instans penampung dan luncurkan versi terbaru Amazon ECS Amazon Linux 2 AMI yang dioptimalkan Amazon (baik secara manual atau dengan memperbarui konfigurasi peluncuran Auto Scaling Anda dengan AMI terbaru). Ini menyediakan instance container baru dengan versi terbaru yang diuji dan divalidasi dariAmazon Linux, Docker`ecs-init`, dan agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.
- Hubungkan ke instans dengan SSH dan perbarui paket `ecs-init` (dan dependensinya) ke versi terbaru. Operasi ini menyediakan versi Docker yang diuji dan divalidasi terbaru dan `ecs-init` yang tersedia di Amazon Linux repositori dan versi terbaru dari agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Untuk memperbarui `ecs-init` paket pada AMI Amazon ECS yang dioptimalkan.
- Perbarui agen penampung dengan operasi `UpdateContainerAgent` API, baik melalui konsol atau dengan AWS CLI atau AWS SDKs. Untuk informasi selengkapnya, lihat Memperbarui agen penampung Amazon ECS dengan operasi `UpdateContainerAgent` API.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Untuk memperbarui **`ecs-init`** paket pada AMI Amazon ECS yang dioptimalkan

1. Masuk ke instans kontainer Anda melalui SSH.
2. Perbarui paket `ecs-init` dengan perintah berikut.

```
sudo yum update -y ecs-init
```

> **Note**
>
> `ecs-init`Paket dan agen kontainer Amazon ECS segera diperbarui. Namun, versi Docker yang lebih baru tidak dimuat sampai daemon Docker di mulai ulang. Pemulaian ulang dilakukan baik dengan booting ulang instans, atau dengan menjalankan perintah berikut pada instans Anda:
>
> - AMI Amazon Linux 2 yang dioptimalkan untuk Amazon ECS:
>
>   ```
>   sudo systemctl restart docker
>   ```
>
> - AMI Amazon Linux yang dioptimalkan untuk Amazon ECS:
>
>   ```
>   sudo service docker restart && sudo start ecs
>   ```

Memperbarui agen penampung Amazon ECS dengan operasi **`UpdateContainerAgent`** API

> **Important**
>
> `UpdateContainerAgent`API ini hanya didukung pada varian Linux dari AMI Amazon ECS yang dioptimalkan, dengan pengecualian AMI Amazon Linux 2 (arm64) Amazon ECS yang dioptimalkan Amazon. Untuk instance kontainer yang menggunakan AMI Amazon Linux 2 (arm64) Amazon ECS yang dioptimalkan Amazon, `ecs-init` perbarui paket untuk memperbarui agen. Untuk instans kontainer yang sedang menjalankan sistem operasi lain, lihat Memperbarui agen penampung Amazon ECS secara manual (untuk non-Amazon ECS yang dioptimalkan) AMIs. Jika Anda menggunakan instans kontainer Windows, Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Proses `UpdateContainerAgent` API dimulai saat Anda meminta pembaruan agen, baik melalui konsol atau dengan AWS CLI atau AWS SDKs. Amazon ECS memeriksa versi agen Anda saat ini terhadap versi agen terbaru yang tersedia, dan jika pembaruan dimungkinkan. Jika pembaruan tidak tersedia, misalnya, jika agen sudah menjalankan versi terbaru, maka `NoUpdateAvailableException` dikembalikan.

Tahapan dalam proses pembaruan yang ditunjukkan di atas adalah sebagai berikut:

`PENDING`

Pembaruan agen tersedia, dan proses pembaruan telah dimulai.

`STAGING`

Agen telah mulai mengunduh pembaruan agen. Jika agen tidak dapat mengunduh pembaruan, atau jika isi pembaruan salah atau rusak, maka agen akan mengirimkan notifikasi kegagalan dan transisi pembaruan ke status `FAILED`.

`STAGED`

Pengunduhan agen selesai dan isi agen telah diverifikasi.

`UPDATING`

Layanan `ecs-init` dimulai ulang dan versi agen baru diambil. Jika agen karena alasan tertentu tidak dapat memulai ulang, pembaruan beralih ke `FAILED` status; jika tidak, agen memberi sinyal kepada Amazon ECS bahwa pembaruan telah selesai.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Untuk memperbarui agen penampung Amazon ECS pada AMI Amazon ECS yang dioptimalkan di konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster dan pilih klaster.

4. Pada *name* halaman Cluster:, pilih tab Infrastruktur.
5. Di bawah Instance Container, pilih instance yang akan diperbarui, lalu pilih Actions, Update agent.

Memperbarui agen penampung Amazon ECS secara manual (untuk non-Amazon ECS yang dioptimalkan) AMIs

Terkadang, Anda mungkin perlu memperbarui agen penampung Amazon ECS untuk mengambil perbaikan bug dan fitur baru. Memperbarui agen penampung Amazon ECS tidak mengganggu tugas atau layanan yang sedang berjalan pada instance container.

> **Note**
>
> Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

1. Masuk ke instans kontainer Anda melalui SSH.
2. Lakukan pemeriksaan untuk melihat apakah agen Anda menggunakan variabel lingkungan `ECS_DATADIR` untuk menyimpan statusnya.

   ```
   ubuntu:~$ docker inspect ecs-agent | grep ECS_DATADIR
   ```

   Output:

   ```
   "ECS_DATADIR=/data",
   ```

   > **Important**
   >
   > Jika perintah sebelumnya tidak mengembalikan variabel lingkungan `ECS_DATADIR`, Anda harus menghentikan tugas apa pun yang berjalan pada instans kontainer ini sebelum memperbarui agen Anda. Agen yang lebih baru dengan variabel lingkungan `ECS_DATADIR` menyimpan statusnya dan Anda dapat memperbaruinya saat tugas sedang berjalan tanpa masalah.

3. Hentikan agen kontainer Amazon ECS.

```
ubuntu:~$ docker stop ecs-agent
```

4. Hapus kontainer agen.

```
ubuntu:~$ docker rm ecs-agent
```

5. Pastikan `/etc/ecs` direktori dan file konfigurasi agen penampung Amazon ECS ada di`/etc/ecs/ecs.config`.

```
ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
```

6. Edit file `/etc/ecs/ecs.config` dan pastikan bahwa file berisi setidaknya deklarasi variabel berikut. Jika Anda tidak ingin instans kontainer Anda untuk terdaftar dengan klaster default, tentukan nama klaster Anda sebagai nilai untuk `ECS_CLUSTER`.

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

> **Note**
>
> Anda dapat menyimpan variabel lingkungan agen secara opsional di Amazon S3 (yang dapat diunduh ke instans penampung Anda pada waktu peluncuran menggunakan data pengguna EC2 Amazon). Hal ini dianjurkan untuk informasi sensitif seperti kredensial autentikasi untuk repositori privat. Untuk informasi selengkapnya, lihat Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3 dan Menggunakan gambar AWS non-kontainer di Amazon ECS.

7. Tarik gambar agen kontainer Amazon ECS terbaru dari Amazon Elastic Container Registry Public.

```
ubuntu:~$ docker pull public.ecr.aws/ecs/amazon-ecs-agent:latest
```

Output:

```
Pulling repository amazon/amazon-ecs-agent
a5a56a5e13dc: Download complete
511136ea3c5a: Download complete
9950b5d678a1: Download complete
c48ddcf21b63: Download complete
Status: Image is up to date for amazon/amazon-ecs-agent:latest
```

8. Jalankan agen penampung Amazon ECS terbaru pada instance container Anda.

> **Note**
>
> Gunakan kebijakan mulai ulang Docker atau pengelola proses (seperti upstart atau systemd) untuk memperlakukan agen kontainer sebagai suatu layanan atau daemon dan pastikan mulai ulang dilakukan setelah keluar. AMI Amazon ECS yang dioptimalkan menggunakan `ecs-init` RPM untuk tujuan ini, dan Anda dapat melihat kode sumber untuk RPM ini aktif. GitHub

Contoh run command agen berikut dipecah menjadi baris terpisah agar setiap opsi dapat ditunjukkan. Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

> **Important**
>
> Sistem operasi dengan SELinux diaktifkan memerlukan `--privileged` opsi dalam docker run perintah Anda. Selain itu, untuk instance kontainer yang SELinux diaktifkan, kami menyarankan Anda menambahkan `:Z` opsi ke pemasangan `/log` dan `/data` volume. Namun, pemasangan host untuk volume ini harus ada sebelum Anda menjalankan perintah atau Anda menerima kesalahan `no such file or directory`. Lakukan tindakan berikut jika Anda mengalami kesulitan menjalankan agen Amazon ECS pada instance container SELinux yang diaktifkan:
>
> - Buat titik-titik pemasangan volume host pada instans kontainer Anda.

```
ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
```

- Tambahkan opsi `--privileged` pada perintah docker run di bawah ini.
- Tambahkan opsi `:Z` pada pemasangan volume kontainer `/log` dan `/data` (misalnya, `--volume=/var/log/ecs/:/log:Z`) pada perintah docker run di bawah ini.

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--volume=/etc/ecs:/etc/ecs/pki \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
```

> **Note**
>
> Jika Anda menerima pesan `Error response from daemon: Cannot start container`, Anda dapat menghapus kontainer yang gagal dengan perintah sudo docker rm ecs-agent dan coba jalankan agen kembali.

Windows Amazon ECS yang dioptimalkan AMIs

Amazon ECS yang dioptimalkan telah AMIs dikonfigurasi sebelumnya dengan komponen yang diperlukan yang Anda perlukan untuk menjalankan beban kerja Amazon ECS. Meskipun Anda dapat membuat instans kontainer sendiri AMI yang memenuhi spesifikasi dasar yang diperlukan untuk menjalankan beban kerja kontainer Anda di Amazon ECS, Amazon ECS yang dioptimalkan AMIs telah dikonfigurasi sebelumnya dan diuji di Amazon ECS oleh para insinyur. AWS Ini adalah cara termudah bagi Anda untuk memulai dan mendapatkan kontainer Anda yang berjalan pada AWS dengan cepat.

Metadata AMI Amazon ECS yang dioptimalkan, termasuk nama AMI, versi agen penampung Amazon ECS, dan versi runtime Amazon ECS yang menyertakan versi Docker, untuk setiap varian dapat

diambil secara terprogram. Untuk informasi selengkapnya, lihat the section called “Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS”.

> **Important**
>
> Semua varian AMI yang dioptimalkan ECS yang diproduksi setelah Agustus 2022 akan bermigrasi dari Docker EE (Mirantis) ke Docker CE (proyek Moby).
> Untuk memastikan bahwa pelanggan memiliki pembaruan keamanan terbaru secara default, Amazon ECS mempertahankan setidaknya tiga Windows Amazon AMIs ECS terakhir yang dioptimalkan. Setelah merilis Windows Amazon ECS baru yang dioptimalkan, AMIs Amazon ECS membuat Windows Amazon ECS AMIs dioptimalkan yang lebih lama menjadi pribadi. Jika ada AMI pribadi yang perlu Anda akses, beri tahu kami dengan mengajukan tiket dengan Cloud Support.

Varian AMI yang dioptimalkan Amazon ECS

Varian Windows Server berikut dari AMI Amazon ECS yang dioptimalkan tersedia untuk instans Amazon EC2 Anda.

> **Important**
>
> Semua varian AMI yang dioptimalkan ECS yang diproduksi setelah Agustus akan bermigrasi dari Docker EE (Mirantis) ke Docker CE (proyek Moby).

- Amazon ECS yang dioptimalkan Windows Server 2025 AMI Penuh
- Amazon ECS yang dioptimalkan Windows Server 2025 Core AMI
- Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh
- Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS
- Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh
- Amazon ECS yang dioptimalkan Windows Server 2019 Core AMI
- Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh

Important

Windows Server 2016 tidak mendukung versi Docker terbaru, misalnya 25.xx. Oleh karena itu Windows Server 2016 Full tidak AMIs akan menerima patch keamanan atau bug ke runtime Docker. Kami menyarankan Anda pindah ke salah satu platform Windows berikut:

- Windows Server 2022 Lengkap
- Windows Server 2022 Inti
- Windows Server 2019 Full
- Windows Server 2019 Core

Pada 9 Agustus 2022, Windows Server 20H2 Core AMI yang dioptimalkan Amazon ECS mencapai akhir tanggal dukungannya. Tidak ada versi baru dari AMI ini yang akan dirilis. Untuk informasi selengkapnya, lihat informasi rilis Windows Server.

Windows Server 2025, Windows Server 2022, Windows Server 2019, dan Windows Server 2016 adalah rilis Saluran Layanan Jangka Panjang (LTSC). Windows Server 20H2 adalah rilis Semi-Annual Channel (SAC). Untuk informasi selengkapnya, lihat informasi rilis Windows Server.

Pertimbangan

Berikut adalah beberapa hal yang harus Anda ketahui tentang wadah Amazon EC2 Windows dan Amazon ECS.

- Kontainer Windows tidak dapat berjalan pada instans kontainer Linux dan sebaliknya. Untuk penempatan tugas Windows dan Linux yang lebih baik, instans kontainer Windows dan Linux harus tetap berada di klaster terpisah dan tempatkan tugas Windows hanya pada klaster Windows. Anda dapat memastikan bahwa ketentuan tugas Windows hanya ditempatkan pada instans Windows dengan menetapkan batasan penempatan berikut: `memberOf(ecs.os-type=='windows')`.
- Kontainer Windows didukung untuk tugas-tugas yang menggunakan jenis peluncuran EC2 dan Fargate.
- Kontainer Windows dan instans kontainer tidak dapat mendukung semua parameter ketentuan tugas yang tersedia untuk kontainer Linux dan instans kontainer. Beberapa parameter tidak didukung sama sekali, dan perilaku parameter lainnya berbeda saat di Windows dan saat di Linux. Untuk informasi selengkapnya, lihat Perbedaan definisi tugas Amazon ECS untuk EC2 instance yang menjalankan Windows.

- Untuk peran IAM untuk fitur tugas, Anda perlu mengonfigurasi instance penampung Windows Anda untuk mengizinkan fitur saat diluncurkan. Kontainer Anda harus menjalankan beberapa PowerShell kode yang disediakan saat mereka menggunakan fitur tersebut. Untuk informasi selengkapnya, lihat Konfigurasi tambahan instans Amazon EC2 Windows.
- Fitur peran IAM untuk tugas menggunakan proxy kredenal untuk memberikan kredensil ke kontainer. Proxy kredenal ini menempati port 80 pada instance kontainer, jadi jika Anda menggunakan peran IAM untuk tugas, port 80 tidak tersedia untuk tugas. Untuk kontainer layanan web, Anda dapat menggunakan Application Load Balancer dan pemetaan port dinamis untuk menyediakan koneksi port HTTP 80 standar ke kontainer Anda. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.
- Gambar Windows Server Docker berukuran besar (9 GiB). Jadi, instance container Windows Anda membutuhkan lebih banyak ruang penyimpanan daripada instance container Linux.
- Untuk menjalankan wadah Windows di Windows Server, versi OS gambar dasar wadah harus cocok dengan host. Untuk informasi selengkapnya, lihat Kompatibilitas versi penampung Windows di situs web dokumentasi Microsoft. Jika klaster Anda menjalankan beberapa versi Windows, Anda dapat memastikan bahwa tugas ditempatkan pada EC2 instance yang berjalan pada versi yang sama dengan menggunakan batasan penempatan:. `memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)` Untuk informasi selengkapnya, lihat the section called “Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS”.

Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS

ID AMI, nama gambar, sistem operasi, versi agen kontainer, dan versi runtime untuk setiap varian Amazon ECS yang dioptimalkan AMIs dapat diambil secara terprogram dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya tentang Systems Manager Parameter Store API, lihat GetParametersdan GetParametersByPath.

> **Note**
>
> Pengguna administratif Anda harus memiliki izin IAM berikut untuk mengambil metadata AMI Amazon ECS yang dioptimalkan. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM.
>
> - ssm: GetParameters
> - ssm: GetParameter
> - ssm: GetParametersByPath

Systems Manager Parameter Menyimpan format parameter

> **Note**
>
> Parameter Systems Manager Parameter Store API berikut tidak digunakan lagi dan tidak boleh digunakan untuk mengambil Windows terbaru: AMIs
>
> - `/aws/service/ecs/optimized-ami/windows_server/2016/english/full/recommended/image_id`
> - `/aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id`

Berikut ini adalah format nama parameter untuk setiap varian AMI Amazon ECS yang dioptimalkan.

- Windows Server 2025 Metadata AMI Lengkap:

```
/aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized
```

- Metadata AMI Inti Windows Server 2025:

```
/aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized
```

- Windows Server 2022 Metadata AMI lengkap:

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

- Metadata AMI Inti Windows Server 2022:

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

- Metadata AMI Windows Server 2019 Full:

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

- Metadata AMI Windows Server 2019 Core:

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

- Metadata AMI Windows Server 2016 Full:

```
/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

Format nama parameter berikut mengambil metadata dari Windows Server 2019 Full AMI stabil terbaru

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-
English-Full-ECS_Optimized
```

Berikut ini adalah contoh objek JSON yang dikembalikan untuk nilai parameter.

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-
ECS_Optimized",
            "Type": "String",
            "Value": "{\"image_name\":\"Windows_Server-2019-English-Full-
ECS_Optimized-2023.06.13\",\"image_id\":\"ami-0debc1fb48e4aee16\",\"ecs_runtime_version
\":\"Docker (CE) version 20.10.21\",\"ecs_agent_version\":\"1.72.0\"}",
            "Version": 58,
            "LastModifiedDate": "2023-06-22T19:37:37.841000-04:00",
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ami-windows-latest/
Windows_Server-2019-English-Full-ECS_Optimized",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

Masing-masing kolom dalam output di atas tersedia untuk dilakukan kueri sebagai sub-parameter. Buat jalur parameter untuk sub-parameter dengan menambahkan nama sub-parameter ke jalur untuk AMI yang dipilih. Sub-parameter berikut tersedia:

- `schema_version`
- `image_id`
- `image_name`
- `os`
- `ecs_agent_version`

- `ecs_runtime_version`

Contoh

Contoh berikut menunjukkan cara Anda dapat mengambil metadata untuk setiap varian AMI Amazon ECS yang dioptimalkan.

Mengambil metadata AMI terbaru yang dioptimalkan Amazon ECS stabil

Anda dapat mengambil AMI stabil Amazon ECS terbaru yang dioptimalkan menggunakan AWS CLI perintah berikut. AWS CLI

- Untuk AMI Penuh Windows Server 2025 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized --region us-east-1
```

- Untuk AMI Inti Windows Server 2025 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized --region us-east-1
```

- Untuk AMI Lengkap Windows Server 2022 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized --region us-east-1
```

- Untuk AMI Inti Windows Server 2022 yang dioptimalkan oleh Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized --region us-east-1
```

- Untuk AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized --region us-east-1
```

- Untuk Windows Server 2019 Core AMI yang dioptimalkan Amazon ECS:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized --region us-east-1
```

- Untuk Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh:

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized --region us-east-1
```

Menggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation

Anda dapat mereferensikan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam AWS CloudFormation template dengan mereferensikan nama penyimpanan parameter Systems Manager.

```
Parameters:
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized/image_id
```

Versi Windows AMI yang dioptimalkan Amazon ECS

Lihat versi saat ini dan sebelumnya dari Amazon ECS yang dioptimalkan AMIs dan versi terkait dari agen kontainer Amazon ECS, Docker, dan paket. `ecs-init`

Metadata AMI Amazon ECS yang dioptimalkan, termasuk ID AMI, untuk setiap varian dapat diambil secara terprogram. Untuk informasi selengkapnya, lihat the section called “Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS”.

Tab berikut menampilkan daftar versi Windows Amazon ECS AMIs yang dioptimalkan. Untuk detail tentang referensi parameter Systems Manager Parameter Store dalam AWS CloudFormation template, lihatMenggunakan AMI terbaru yang dioptimalkan Amazon ECS yang dioptimalkan dalam template AWS CloudFormation.

> **Important**
>
> Untuk memastikan bahwa pelanggan memiliki pembaruan keamanan terbaru secara default, Amazon ECS mempertahankan setidaknya tiga Windows Amazon AMIs ECS terakhir yang dioptimalkan. Setelah merilis Windows Amazon ECS baru yang dioptimalkan, AMIs Amazon ECS membuat Windows Amazon ECS AMIs dioptimalkan yang lebih lama menjadi pribadi.

Jika ada AMI pribadi yang perlu Anda akses, beri tahu kami dengan mengajukan tiket dengan Cloud Support.
Windows Server 2016 tidak mendukung versi Docker terbaru, misalnya 25.xx. Oleh karena itu Windows Server 2016 Full tidak AMIs akan menerima patch keamanan atau bug ke runtime Docker. Kami menyarankan Anda pindah ke salah satu platform Windows berikut:

- Windows Server 2022 Lengkap
- Windows Server 2022 Inti
- Windows Server 2019 Full
- Windows Server 2019 Core

Windows Server 2025 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari AMI Penuh Windows Server 2025 yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2025 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2025-Inggris-penuh-ECS_dioptimal-2025.06.13	`1.94.0`	`25.0.6 (Docker CE)`	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2025 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized
```

Windows Server 2025 Core AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Windows Server 2025 Core AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2025 Core AMI	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2025-Inggris-core-ECS_Optimized-2025.06.13	`1.94.0`	`25.0.6 (Docker CE)`	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Inti Windows Server 2025 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized
```

Windows Server 2022 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2025.06.13	`1.94.0`	`25.0.6 (Docker CE)`	Publik

Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2025.05.17	1.93.1	25.0.6 (Docker CE)	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2025.4.14	1.91.2	25.0.6 (Docker CE)	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2025.3.17	1.91.0	25.0.6 (Docker CE)	Publik
Windows_server-2022-Inggris-penuh-ECS_dioptimal-2025.2.14	1.90.0	25.0.6 (Docker CE)	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

Windows Server 2022 Core AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Windows Server 2022 Core AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_Server-2022-Inggris-Core-ECS_Optimized-2025.06.13	1.94.0	25.0.6 (Docker CE)	Publik
Windows_server-2022-Inggris-core-ECS_Optimized-2025.05.17	1.93.1	25.0.6 (Docker CE)	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2025.4.14	1.91.2	25.0.6 (Docker CE)	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2025.3.17	1.91.0	25.0.6 (Docker CE)	Publik
Windows_Server-2022-Inggris-Core-ECS_Optimized-2025.2.14	1.90.0	25.0.6 (Docker CE)	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2022 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

Windows Server 2019 Full AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2025.06.13	`1.94.0`	`25.0.6 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2025.05.17	`1.93.1`	`25.0.6 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2025.4.14	`1.91.2`	`25.0.6 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2025.3.17	`1.91.0`	`25.0.6 (Docker CE)`	Publik
Windows_server-2019-Inggris-penuh-ECS_dioptimal-2025.2.14	`1.90.0`	`25.0.6 (Docker CE)`	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Windows Server 2019 Core AMI versions

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Windows Server 2019 Core AMI yang dioptimalkan Amazon ECS dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2019 Core AMI	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2019-Inggris-Core-ECS_Optimized-2025.06.13	1.94.0	25.0.6 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2025.05.17	1.93.1	25.0.6 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2025.4.14	1.91.2	25.0.6 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2025.3.17	1.91.0	25.0.6 (Docker CE)	Publik
Windows_server-2019-Inggris-Core-ECS_Optimized-2025.2.14	1.90.0	25.0.6 (Docker CE)	Publik

Gunakan AWS CLI perintah berikut untuk mengambil AMI Penuh Windows Server 2019 yang dioptimalkan Amazon ECS saat ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

Windows Server 2016 Full AMI versions

> **Important**
>
> Windows Server 2016 tidak mendukung versi Docker terbaru, misalnya 25.xx. Oleh karena itu Windows Server 2016 Full tidak AMIs akan menerima patch keamanan atau bug ke runtime Docker. Kami menyarankan Anda pindah ke salah satu platform Windows berikut:
>
> - Windows Server 2022 Lengkap
> - Windows Server 2022 Inti
> - Windows Server 2019 Full
> - Windows Server 2019 Core

Tabel di bawah ini mencantumkan versi saat ini dan sebelumnya dari Amazon ECS yang dioptimalkan Windows Server 2016 Full AMI dan versi terkait dari agen penampung Amazon ECS dan Docker.

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
Windows_server-2016-Inggris-penuh-ECS_dioptimal-2025.06.13	`1.94.0`	`20.10.23 (Docker CE)`	Publik
Windows_server-2016-Inggris-penuh-EC	`1.93.1`	`20.10.23 (Docker CE)`	Publik

Amazon ECS yang dioptimalkan Windows Server 2016 AMI Penuh	Versi agen kontainer Amazon ECS	Versi Docker	Visibilitas
S_Optimized-2025.05.17			
Windows_server-2016-Inggris-penuh-ECS_Optimized-2025.4.14	1.91.2	20.10.23 (Docker CE)	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2025.3.17	1.91.0	20.10.23 (Docker CE)	Publik
Windows_server-2016-Inggris-penuh-ECS_Optimized-2025.2.14	1.90.0	20.10.23 (Docker CE)	Publik

Gunakan AMI Penuh Windows Server 2016 yang dioptimalkan AWS CLI Amazon ECS berikut ini.

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

Membangun Windows AMI Amazon ECS Anda sendiri yang dioptimalkan

Gunakan EC2 Image Builder untuk membuat Windows AMI yang dioptimalkan Amazon ECS kustom Anda sendiri. Ini membuatnya mudah untuk menggunakan AMI Windows dengan lisensi Anda sendiri di Amazon ECS. Amazon ECS menyediakan komponen Image Builder terkelola yang menyediakan konfigurasi sistem yang diperlukan untuk menjalankan instance Windows untuk meng-host container Anda. Setiap komponen yang dikelola Amazon ECS menyertakan agen kontainer tertentu dan versi Docker. Anda dapat menyesuaikan gambar untuk menggunakan komponen terkelola Amazon

ECS terbaru, atau jika agen penampung atau versi Docker yang lebih lama diperlukan, Anda dapat menentukan komponen yang berbeda.

Untuk panduan lengkap tentang penggunaan EC2 Image Builder, lihat Memulai EC2 Image Builder di Panduan Pengguna EC2 Image Builder.

Saat membuat Windows AMI yang dioptimalkan Amazon ECS menggunakan EC2 Image Builder, Anda membuat resep gambar. Formula citra Anda harus memenuhi persyaratan berikut:

- Gambar Sumber harus didasarkan pada Windows Server 2019 Core, Windows Server 2019 Full, Windows Server 2022 Core, atau Windows Server 2022 Full. Sistem operasi Windows lain tidak didukung dan mungkin tidak kompatibel dengan komponen.
- Ketika menentukan komponen Pembangunan, maka diperlukan komponen `ecs-optimized-ami-windows`. Komponen `update-windows` disarankan, untuk menjamin bahwa citra memiliki pembaruan keamanan terbaru.

  Untuk menentukan versi komponen yang berbeda, perluas menu Opsi versioning dan tentukan versi komponen yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Membuat daftar versi komponen `ecs-optimized-ami-windows`.

Membuat daftar versi komponen **`ecs-optimized-ami-windows`**

Saat membuat resep EC2 Image Builder dan menentukan `ecs-optimized-ami-windows` komponen, Anda dapat menggunakan opsi default atau Anda dapat menentukan versi komponen tertentu. Untuk menentukan versi komponen apa yang tersedia, bersama dengan agen kontainer Amazon ECS dan versi Docker yang terdapat dalam komponen, Anda dapat menggunakan. AWS Management Console

Untuk mendaftarkan versi komponen **`ecs-optimized-ami-windows`** yang tersedia

1. Buka konsol EC2 Image Builder di https://console.aws.amazon.com/imagebuilder/.
2. Pada bilah navigasi, pilih Wilayah yang sedang membangun citra Anda.
3. Di panel navigasi, di bawah menu konfigurasi Tersimpan, pilih Komponen.
4. Pada halaman Komponen, di bilah pencarian, ketik `ecs-optimized-ami-windows` dan tarik menu kualifikasi ke bawah lalu pilih Quick start (Amazon-terkelola).
5. Gunakan kolom Deskripsi untuk menentukan versi komponen dengan agen penampung Amazon ECS dan versi Docker yang dibutuhkan gambar Anda.

Manajemen instans penampung Amazon ECS Windows

Saat Anda menggunakan EC2 instans untuk beban kerja Amazon ECS, Anda bertanggung jawab untuk mempertahankan instans.

Pembaruan agen tidak berlaku pada instans kontainer Windows. Kami menyarankan Anda agar meluncurkan instans kontainer baru untuk memperbarui versi agen di klaster Windows Anda.

Prosedur manajemen

- Meluncurkan instans penampung Amazon ECS Windows
- Bootstrapping instans penampung Amazon ECS Windows untuk meneruskan data
- Menggunakan proxy HTTP untuk instans penampung Amazon ECS Windows
- Mengonfigurasi instans penampung Amazon ECS Windows untuk menerima pemberitahuan Instans Spot

Meluncurkan instans penampung Amazon ECS Windows

Instans penampung Amazon ECS Anda dibuat menggunakan konsol Amazon EC2 . Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah tersebut di Siapkan untuk menggunakan Amazon ECS.

Untuk informasi selengkapnya tentang panduan peluncuran, lihat Meluncurkan instance menggunakan wizard instans peluncuran baru di Panduan EC2 Pengguna Amazon.

Anda dapat menggunakan EC2 wizard Amazon baru untuk meluncurkan instance. Anda dapat menggunakan daftar berikut untuk parameter dan membiarkan parameter tidak terdaftar sebagai default. Instruksi berikut membawa Anda melalui setiap kelompok parameter.

Prosedur

Sebelum memulai, selesaikan langkah-langkah di Siapkan untuk menggunakan Amazon ECS.

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Di bilah navigasi di bagian atas layar, AWS Wilayah saat ini ditampilkan (misalnya, AS Timur (Ohio)). Pilih Wilayah tempat instans akan diluncurkan. Pilihan ini penting karena beberapa EC2 sumber daya Amazon dapat dibagi antar Wilayah, sementara yang lain tidak.
3. Dari dasbor EC2 konsol Amazon, pilih Launch instance.

Nama dan tanda

Nama instans adalah tanda, di mana kuncinya adalah Name, dan nilainya adalah nama yang Anda tentukan. Anda dapat menandai instance, volume, dan grafik elastis. Untuk Instans Spot, Anda hanya dapat menandai permintaan Instans Spot.

Menentukan nama instans dan tanda tambahan bersifat opsional.

- Untuk Nama, masukkan nama deskriptif untuk instans tersebut. Jika Anda tidak menentukan nama, instans dapat diidentifikasi berdasarkan ID-nya, yang secara otomatis dihasilkan saat Anda meluncurkan instans tersebut.
- Untuk menambahkan tanda tambahan, pilih Tambahkan tanda tambahan. Pilih Tambahkan tanda, lalu masukkan kunci dan nilai, lalu pilih jenis sumber daya yang akan diberi tanda. Pilih Tambah tanda lagi untuk setiap tanda tambahan yang akan ditambahkan.

Aplikasi dan Gambar OS (Gambar Mesin Amazon)

Amazon Machine Image (AMI) berisi informasi yang diperlukan untuk membuat instans. Misalnya, AMI mungkin berisi perangkat lunak yang diperlukan untuk bertindak sebagai server web, seperti Apache, dan situs web Anda.

Untuk Amazon ECS terbaru yang dioptimalkan AMIs dan nilainya, lihat AMI yang dioptimalkan untuk Windows Amazon ECS.

Gunakan bilah Pencarian untuk menemukan AMI yang dioptimalkan Amazon ECS yang dioptimalkan yang diterbitkan oleh. AWS

1. Berdasarkan kebutuhan Anda, masukkan salah satu dari yang berikut ini AMIs di bilah Pencarian dan tekan Enter.
   - Windows_server-2022-Inggris-penuh-ECS_dioptimalkan
   - Windows_Server-2022-Inggris-Core-ECS_Optimized
   - Windows_server-2019-Inggris-penuh-ECS_dioptimalkan
   - Windows_Server-2019-Inggris-Core-ECS_Optimized
   - Windows_server-2016-Inggris-penuh-ECS_dioptimalkan
2. Pada halaman Pilih Gambar Mesin Amazon (AMI), pilih AMIs tab Komunitas.
3. Dari daftar yang muncul, pilih AMI yang diverifikasi Microsoft dengan tanggal publikasi terbaru dan klik Pilih.

Jenis instans

Tipe instans mendefinisikan konfigurasi perangkat keras dan ukuran instans. Tipe instans yang lebih besar memiliki lebih banyak CPU dan memori. Untuk informasi selengkapnya, lihat Tipe instans.

- Untuk Tipe instans, pilih tipe instans untuk instans tersebut.

  Tipe instans yang Anda pilih menentukan sumber daya yang tersedia untuk menjalankan tugas Anda.

Pasangan kunci (login)

Untuk Nama pasangan kunci, pilih pasangan kunci yang ada, atau pilih Buat pasangan kunci baru untuk membuat yang baru.

> **Important**
>
> Jika Anda memilih opsi Lanjutkan tanpa pasangan kunci (Tidak direkomendasikan), Anda tidak akan dapat terhubung ke instans tersebut, kecuali Anda memilih sebuah AMI yang dikonfigurasi agar pengguna dapat masuk dengan cara lain.

Pengaturan jaringan

Konfigurasikan pengaturan jaringan, sesuai keperluan.

- Platform jaringan: Pilih Virtual Private Cloud (VPC), lalu tentukan subnet di bagian Network interface.
- VPC: Pilih VPC yang ada untuk membuat grup keamanan.
- Subnet: Anda dapat meluncurkan sebuah instans di subnet yang terkait dengan Zona Ketersediaan, Local Zone, Wavelength Zone, atau Outpost.

  Untuk meluncurkan instans di Zona Ketersediaan, pilih subnet tempat Anda akan meluncurkan instans. Untuk membuat subnet baru, pilih Buat subnet baru untuk membuka konsol Amazon VPC. Setelah selesai, kembali ke wizard peluncuran instans dan pilih ikon Segarkan untuk memuat subnet Anda dalam daftar.

  Untuk meluncurkan instans di Local Zone, pilih subnet yang Anda buat di Local Zone.

Untuk meluncurkan sebuah instans di Outpost, pilih subnet di VPC yang Anda kaitkan dengan Outpost.

- Auto-assign IP Publik: Jika instans Anda harus dapat diakses dari internet, verifikasi bahwa bidang Auto-assign Public IP diatur ke Aktifkan. Jika tidak, atur bidang ini ke Nonaktifkan.

> **Note**
>
> Instans kontainer memerlukan akses untuk berkomunikasi dengan titik akhir layanan Amazon ECS. Ini bisa melalui titik akhir VPC antarmuka atau melalui instance container Anda yang memiliki alamat IP publik.
> Untuk informasi selengkapnya tentang titik akhir VPC antarmuka, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink
> Jika Anda tidak memiliki antarmuka VPC endpoint yang dikonfigurasi dan instance container Anda tidak memiliki alamat IP publik, maka mereka harus menggunakan terjemahan alamat jaringan (NAT) untuk menyediakan akses ini. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC dan dalam panduan ini. Menggunakan proxy HTTP untuk instans penampung Amazon ECS Linux

- Firewall (grup keamanan): Gunakan grup keamanan untuk menentukan aturan firewall untuk instance container Anda. Aturan ini menentukan lalu lintas jaringan masuk yang dikirim ke instans kontainer Anda. Semua lalu lintas lainnya diabaikan.
  - Untuk memilih grup keamanan yang ada, pilih Pilih grup keamanan yang ada, dan pilih grup keamanan yang Anda buat Siapkan untuk menggunakan Amazon ECS

Mengonfigurasi penyimpanan

AMI yang Anda pilih mencakup satu atau lebih volume penyimpanan, termasuk volume root. Anda dapat menentukan volume tambahan untuk dilampirkan ke instans.

Anda dapat menggunakan tampilan Sederhana.

- Jenis penyimpanan: Konfigurasikan penyimpanan untuk instance kontainer Anda.

  Jika Anda menggunakan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon, instans Anda memiliki dua volume yang dikonfigurasi. Volume Root adalah untuk penggunaan sistem operasi, dan volume Amazon EBS kedua (dilampirkan ke`/dev/xvdcz`) adalah untuk penggunaan Docker.

Anda dapat menambah atau mengurangi ukuran volume bagi instans Anda untuk memenuhi kebutuhan aplikasi Anda.

Detail lanjutan

Untuk Detail lanjutan, perluas bagian untuk melihat kolom dan menentukan parameter tambahan apa pun untuk instans.

- Opsi pembelian: Pilih Minta Instans Spot untuk meminta Instans Spot. Anda juga perlu mengatur bidang lain yang terkait dengan Instans Spot. Untuk informasi selengkapnya, lihat Permintaan Instans Spot.

  > **Note**
  >
  > Jika Anda menggunakan Instans Spot dan melihat `Not available` pesan, Anda mungkin perlu memilih jenis instans yang berbeda.

  .

- Profil instans IAM: Pilih peran IAM instance container Anda. Hal ini biasanya bernama `ecsInstanceRole`.

  > **Important**
  >
  > Jika Anda tidak meluncurkan instans penampung dengan izin IAM yang tepat, agen Amazon ECS Anda tidak dapat terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.

- (Opsional) Data pengguna: Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna, seperti variabel lingkungan agen dariKonfigurasi agen kontainer Amazon ECS. Skrip data EC2 pengguna Amazon dijalankan hanya satu kali, saat instance pertama kali diluncurkan. Berikut ini adalah contoh umum dari kegunaan data pengguna:

  - Secara default, instans kontainer Anda meluncurkan ke klaster default Anda. Untuk meluncurkan ke klaster non-default, pilih daftar Detail Lanjutan. Kemudian, tempelkan skrip berikut ke bidang Data pengguna, ganti *`your_cluster_name`* dengan nama cluster Anda.

    `EnableTaskIAMRole`Menghidupkan fitur peran IAM Tugas untuk tugas.

Selain itu, opsi berikut tersedia saat Anda menggunakan mode `awsvpc` jaringan.

- `EnableTaskENI`: Bendera ini mengaktifkan jaringan tugas dan diperlukan saat Anda menggunakan mode `awsvpc` jaringan.
- `AwsvpcBlockIMDS`: Bendera opsional ini memblokir akses IMDS untuk wadah tugas yang berjalan dalam mode `awsvpc` jaringan.
- `AwsvpcAdditionalLocalRoutes`: Bendera opsional ini memungkinkan Anda memiliki rute tambahan di namespace tugas.

  Ganti `ip-address` dengan Alamat IP untuk rute tambahan, misalnya 172.31.42.23/32.

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -EnableTaskENI -
AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

Bootstrapping instans penampung Amazon ECS Windows untuk meneruskan data

Saat meluncurkan EC2 instans Amazon, Anda dapat meneruskan data pengguna ke EC2 instans. Data dapat digunakan untuk melakukan tugas konfigurasi otomatis umum dan bahkan menjalankan skrip ketika boot instans. Untuk Amazon ECS, kasus penggunaan yang paling umum untuk data pengguna adalah meneruskan informasi konfigurasi ke daemon Docker dan agen penampung Amazon ECS.

Anda dapat meneruskan beberapa jenis data pengguna ke Amazon EC2, termasuk boothook cloud, skrip shell, dan arahan. `cloud-init` Untuk informasi selengkapnya tentang hal ini dan tipe format lainnya, lihat dokumentasi Cloud-init.

Anda dapat meneruskan data pengguna ini saat menggunakan wizard EC2 peluncuran Amazon. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Data pengguna Windows default

Contoh skrip data pengguna ini menunjukkan data pengguna default yang diterima instance container Windows Anda jika Anda menggunakan konsol. Script di bawah ini melakukan hal berikut:

- Menetapkan nama cluster ke nama yang Anda masukkan.

- Menetapkan peran IAM untuk tugas.
- Atur `json-file` dan `awslogs` sebagai driver pencatatan yang tersedia.

Selain itu, opsi berikut tersedia saat Anda menggunakan mode `awsvpc` jaringan.

- `EnableTaskENI`: Bendera ini mengaktifkan jaringan tugas dan diperlukan saat Anda menggunakan mode `awsvpc` jaringan.
- `AwsvpcBlockIMDS`: Bendera opsional ini memblokir akses IMDS untuk wadah tugas yang berjalan dalam mode `awsvpc` jaringan.
- `AwsvpcAdditionalLocalRoutes`: Bendera opsional ini memungkinkan Anda memiliki rute tambahan.

  Ganti `ip-address` dengan Alamat IP untuk rute tambahan, misalnya 172.31.42.23/32.

Anda dapat menggunakan skrip ini untuk instance kontainer Anda sendiri (asalkan diluncurkan dari AMI Server Windows Amazon ECS yang dioptimalkan).

Ganti `-Cluster` *`cluster-name`* baris untuk menentukan nama cluster Anda sendiri.

```
<powershell>
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]' -EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

Untuk tugas Windows yang dikonfigurasi untuk menggunakan driver pencatatan `awslogs`, Anda juga harus mengatur variabel lingkungan `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` pada instans kontainer Anda. Gunakan sintaksis berikut:

Ganti `-Cluster` *`cluster-name`* baris untuk menentukan nama cluster Anda sendiri.

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Data pengguna instalasi agen Windows

Contoh skrip data pengguna ini menginstal agen penampung Amazon ECS pada instance yang diluncurkan dengan AMI Windows_Server-2016-English-Full-Containers. Ini telah diadaptasi dari instruksi instalasi agen di halaman README GitHubrepositori Agen Kontainer Amazon ECS.

> **Note**
>
> Skrip ini dibagikan untuk keperluan contoh. Jauh lebih mudah untuk memulai dengan wadah Windows dengan menggunakan AMI Windows Server Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Membuat cluster Amazon ECS untuk jenis peluncuran Fargate.

Untuk informasi tentang cara menginstal agen Amazon ECS di Windows Server 2022 Lengkap, lihat Masalah 3753 di. GitHub

Anda dapat menggunakan skrip ini untuk instans kontainer Anda sendiri (asalkan mereka diluncurkan dengan versi AMI Windows_Server-2016-English-Full-Containers). Pastikan untuk mengganti baris *windows* untuk menentukan nama klaster Anda sendiri (jika Anda tidak menggunakan sebuah klaster bernama `windows`).

```
<powershell>
# Set up directories the agent uses
New-Item -Type directory -Path ${env:ProgramFiles}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS\data -Force
# Set up configuration
$ecsExeDir = "${env:ProgramFiles}\Amazon\ECS"
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", "windows", "Machine")
[Environment]::SetEnvironmentVariable("ECS_LOGFILE", "${env:ProgramData}\Amazon\ECS\log
\ecs-agent.log", "Machine")
[Environment]::SetEnvironmentVariable("ECS_DATADIR", "${env:ProgramData}\Amazon\ECS
\data", "Machine")
# Download the agent
$agentVersion = "latest"
$agentZipUri = "https://s3.amazonaws.com/amazon-ecs-agent/ecs-agent-windows-
$agentVersion.zip"
$zipFile = "${env:TEMP}\ecs-agent.zip"
Invoke-RestMethod -OutFile $zipFile -Uri $agentZipUri
# Put the executables in the executable directory.
```

```
Expand-Archive -Path $zipFile -DestinationPath $ecsExeDir -Force
Set-Location ${ecsExeDir}
# Set $EnableTaskIAMRoles to $true to enable task IAM roles
# Note that enabling IAM roles will make port 80 unavailable for tasks.
[bool]$EnableTaskIAMRoles = $false
if (${EnableTaskIAMRoles}) {
  $HostSetupScript = Invoke-WebRequest https://raw.githubusercontent.com/aws/amazon-
ecs-agent/master/misc/windows-deploy/hostsetup.ps1
  Invoke-Expression $($HostSetupScript.Content)
}
# Install the agent service
New-Service -Name "AmazonECS" `
        -BinaryPathName "$ecsExeDir\amazon-ecs-agent.exe -windows-service" `
        -DisplayName "Amazon ECS" `
        -Description "Amazon ECS service runs the Amazon ECS agent" `
        -DependsOn Docker `
        -StartupType Manual
sc.exe failure AmazonECS reset=300 actions=restart/5000/restart/30000/restart/60000
sc.exe failureflag AmazonECS 1
Start-Service AmazonECS
</powershell>
```

Menggunakan proxy HTTP untuk instans penampung Amazon ECS Windows

Anda dapat mengonfigurasi instans penampung Amazon ECS untuk menggunakan proxy HTTP untuk agen penampung Amazon ECS dan daemon Docker. Ini berguna jika instance container Anda tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instance.

Untuk mengonfigurasi instans penampung Amazon ECS Windows Anda agar menggunakan proxy HTTP, setel variabel berikut pada waktu peluncuran (dengan data EC2 pengguna Amazon).

```
[Environment]::SetEnvironmentVariable("HTTP_PROXY",
"http://proxy.mydomain:port", "Machine")
```

Setel HTTP_PROXY ke nama host (atau alamat IP) dan nomor port proxy HTTP yang akan digunakan agen Amazon ECS untuk terhubung ke internet. Misalnya, instance container Anda mungkin tidak memiliki akses jaringan eksternal melalui gateway internet VPC Amazon, gateway NAT, atau instans.

```
[Environment]::SetEnvironmentVariable("NO_PROXY",
"169.254.169.254,169.254.170.2,\\.\pipe\docker_engine", "Machine")
```

Setel `NO_PROXY` `169.254.169.254,169.254.170.2,\\.\pipe\docker_engine` untuk memfilter metadata EC2 instance, peran IAM untuk tugas, dan lalu lintas daemon Docker dari proxy.

Example Skrip data pengguna proxy HTTP Windows

Contoh PowerShell skrip data pengguna di bawah ini mengonfigurasi agen penampung Amazon ECS dan daemon Docker untuk menggunakan proxy HTTP yang Anda tentukan. Anda juga dapat menentukan klaster tempat instans kontainer akan terdaftar.

Untuk menggunakan skrip ini ketika Anda meluncurkan instans kontainer, ikuti langkah-langkah di the section called “Meluncurkan instans kontainer”. Cukup salin dan tempel PowerShell skrip di bawah ini ke bidang data Pengguna (pastikan untuk mengganti nilai contoh merah dengan informasi proxy dan cluster Anda sendiri).

> **Note**
>
> `-EnableTaskIAMRole`Opsi ini diperlukan untuk mengaktifkan peran IAM untuk tugas. Untuk informasi selengkapnya, lihat Konfigurasi tambahan instans Amazon EC2 Windows.

```
<powershell>
Import-Module ECSTools

$proxy = "http://proxy.mydomain:port"
[Environment]::SetEnvironmentVariable("HTTP_PROXY", $proxy, "Machine")
[Environment]::SetEnvironmentVariable("NO_PROXY", "169.254.169.254,169.254.170.2,\\.
\pipe\docker_engine", "Machine")

Restart-Service Docker
Initialize-ECSAgent -Cluster MyCluster -EnableTaskIAMRole
</powershell>
```

Mengonfigurasi instans penampung Amazon ECS Windows untuk menerima pemberitahuan Instans Spot

Amazon EC2 mengakhiri, menghentikan, atau hibernasi Instans Spot Anda ketika harga Spot melebihi harga maksimum untuk permintaan atau kapasitas Anda tidak lagi tersedia. Amazon EC2 menyediakan pemberitahuan interupsi Instans Spot, yang memberi instance peringatan dua menit sebelum terputus. Jika pengurasan Instans Spot Amazon ECS diaktifkan pada instans, ECS menerima pemberitahuan gangguan Instans Spot dan menempatkan instance dalam status. `DRAINING`

> **Important**
>
> Amazon ECS memantau pemberitahuan gangguan Instans Spot yang memiliki dan tindakan instans. `terminate` `stop` Jika Anda menentukan perilaku interupsi `hibernate` instans saat meminta Instans Spot atau Armada Spot, maka pengurasan Instans Spot Amazon ECS tidak didukung untuk instans tersebut.

Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container. Tugas layanan pada instans kontainer pengurasan yang ada di status `PENDING` segera dihentikan. Jika ada instans kontainer di klaster yang tersedia, tugas layanan pengganti dimulai.

Anda dapat mengaktifkan pengurasan Instance Spot saat meluncurkan instance. Anda harus mengatur `ECS_ENABLE_SPOT_INSTANCE_DRAINING` parameter sebelum memulai agen kontainer. Ganti *`my-cluster`* dengan nama klaster Anda.

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

Untuk informasi selengkapnya, lihat the section called “Meluncurkan instans kontainer”.

Cluster Amazon ECS untuk jenis peluncuran eksternal

Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM), ke kluster Amazon ECS Anda. Instans eksternal dioptimalkan guna

menjalankan aplikasi yang menghasilkan lalu lintas keluar atau data proses. Jika aplikasi Anda memerlukan lalu lintas masuk, kurangnya dukungan Elastic Load Balancing membuat menjalankan beban kerja ini menjadi kurang efisien. Amazon ECS menambahkan jenis `EXTERNAL` peluncuran baru yang dapat Anda gunakan untuk membuat layanan atau menjalankan tugas pada instans eksternal Anda.

Sistem pengoperasian dan arsitektur sistem yang didukung

Berikut ini merupakan daftar sistem pengoperasian dan arsitektur sistem yang didukung.

- Amazon Linux 2
- Amazon Linux 2023
- CentOS Aliran 9
- RHEL 7, RHEL 8, RHEL 9 - Baik repositori paket terbuka Docker atau RHEL tidak mendukung penginstalan Docker secara native di RHEL. Anda harus memastikan bahwa Docker telah diinstal sebelum menjalankan skrip penginstalan yang dijelaskan dalam dokumen ini.
- Fedora 32, Fedora 33, Fedora 40
- openSUSE Tumbleweed
- Ubuntu 18, Ubuntu 20, Ubuntu 22, Ubuntu 24
- Debian 10

> **Important**
>
> Debian 9 Long Term Support (dukungan LTS) berakhir pada 30 Juni 2022 dan tidak lagi didukung oleh Amazon ECS Anywhere.

- Debian 11
- Debian 12
- SUSE Enterprise Server 15
- Arsitektur CPU `x86_64` dan `ARM64` didukung.
- Versi sistem operasi Windows berikut didukung:
  - Windows Server 2022
  - Windows Server 2019
  - Windows Server 2016

- Windows Server 20H2

Pertimbangan

Sebelum Anda mulai menggunakan instance eksternal, perhatikan pertimbangan berikut.

- Anda dapat mendaftarkan instans eksternal ke dalam satu klaster dalam satu waktu. Untuk instruksi tentang cara mendaftarkan instans eksternal dengan klaster yang berbeda, lihat Membatalkan pendaftaran instans eksternal Amazon ECS.
- Instance eksternal Anda memerlukan peran IAM yang memungkinkan mereka untuk berkomunikasi. AWS APIs Untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere.
- Instance eksternal Anda seharusnya tidak memiliki rantai kredensi instans yang telah dikonfigurasi yang ditentukan secara lokal karena ini akan mengganggu skrip pendaftaran.
- Untuk mengirim log kontainer ke CloudWatch Log, pastikan Anda membuat dan menentukan peran IAM eksekusi tugas dalam definisi tugas Anda.
- Ketika instans eksternal didaftarkan ke sebuah klaster, atribut `ecs.capability.external` dikaitkan dengan instans tersebut. Atribut ini mengidentifikasikan instans tersebut sebagai instans eksternal. Anda dapat menambahkan atribut khusus ke instans eksternal untuk digunakan sebagai batasan penempatan tugas. Untuk informasi selengkapnya, lihat Atribut kustom.
- Anda dapat menambahkan tanda sumber daya ke instans eksternal Anda. Untuk informasi selengkapnya, lihat Contoh kontainer eksternal.
- ECS Exec didukung pada instans eksternal. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.
- Berikut ini merupakan pertimbangan tambahan yang khusus untuk jaringan dengan instans eksternal Anda. Untuk informasi selengkapnya, lihat Jaringan .
  - Penyeimbangan beban layanan tidak didukung.
  - Penemuan layanan tidak didukung.
  - Tugas yang berjalan pada instans eksternal harus menggunakan mode jaringan `bridge`, `host`, atau `none`. Mode jaringan `awsvpc` tidak didukung.
  - Ada domain layanan Amazon ECS di setiap AWS Wilayah. Domain layanan ini harus diizinkan untuk mengirim lalu lintas ke instans eksternal Anda.
  - Agen SSM yang diinstal pada instans eksternal Anda mempertahankan kredensional IAM yang diputar setiap 30 menit menggunakan sidik jari perangkat keras. Jika instans eksternal Anda kehilangan koneksi AWS, Agen SSM secara otomatis menyegarkan kredensialnya setelah

koneksi dibuat kembali. Untuk informasi selengkapnya, lihat Memvalidasi server lokal dan mesin virtual menggunakan sidik jari perangkat keras di AWS Systems Manager Panduan Pengguna.

- API `UpdateContainerAgent` tidak didukung. Untuk petunjuk tentang cara memperbarui Agen SSM atau agen Amazon ECS pada instans eksternal Anda, lihat. Memperbarui AWS Systems Manager agen dan agen penampung Amazon ECS pada instance eksternal
- Penyedia kapasitas Amazon ECS tidak didukung. Untuk membuat layanan atau menjalankan tugas mandiri pada instans eksternal Anda, gunakan tipe peluncuran `EXTERNAL`.
- SELinux tidak didukung.
- Menggunakan volume Amazon EFS atau menentukan `EFSVolumeConfiguration` tidak didukung.
- Integrasi dengan App Mesh tidak didukung.
- Jika Anda menggunakan konsol untuk membuat definisi tugas instance eksternal, Anda harus membuat definisi tugas dengan editor JSON konsol.
- Ketika Anda menjalankan ECS Anywhere di Windows, Anda harus menggunakan lisensi Windows Anda sendiri pada infrastruktur lokal.
- Bila Anda menggunakan AMI yang tidak dioptimalkan ECS Amazon, jalankan perintah berikut pada instance container eksternal untuk mengonfigurasi aturan guna menggunakan peran IAM untuk tugas. Untuk informasi selengkapnya, lihat Konfigurasi tambahan contoh eksternal.

```
$ sysctl -w net.ipv4.conf.all.route_localnet=1
$ iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
$ iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

Jaringan

Instans eksternal Amazon ECS dioptimalkan untuk menjalankan aplikasi yang menghasilkan lalu lintas keluar atau data proses. Jika aplikasi Anda memerlukan lalu lintas masuk, seperti layanan web, kurangnya dukungan Elastic Load Balancing membuat menjalankan beban kerja ini kurang efisien karena tidak ada dukungan untuk menempatkan beban kerja ini di belakang penyeimbang beban.

Berikut ini merupakan pertimbangan tambahan yang khusus untuk jaringan dengan instans eksternal Anda.

- Penyeimbangan beban layanan tidak didukung.

- Penemuan layanan tidak didukung.
- Tugas Linux yang berjalan pada instance eksternal harus menggunakan`bridge`,`host`, atau mode `none` jaringan. Mode jaringan `awsvpc` tidak didukung.

  Untuk informasi selengkapnya tentang setiap mode jaringan, lihat opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran.
- Tugas Windows yang berjalan pada instance eksternal harus menggunakan mode `default` jaringan.
- Ada domain layanan Amazon ECS di setiap Wilayah dan harus diizinkan untuk mengirim lalu lintas ke instans eksternal Anda.
- Agen SSM yang diinstal pada instans eksternal Anda mempertahankan kredensional IAM yang diputar setiap 30 menit menggunakan sidik jari perangkat keras. Jika instans eksternal Anda kehilangan koneksi AWS, Agen SSM secara otomatis menyegarkan kredensialnya setelah koneksi dibuat kembali. Untuk informasi selengkapnya, lihat Memvalidasi server lokal dan mesin virtual menggunakan sidik jari perangkat keras di AWS Systems Manager Panduan Pengguna.

Domain berikut digunakan untuk komunikasi antara layanan Amazon ECS dan agen Amazon ECS yang diinstal pada instans eksternal Anda. Pastikan lalu lintas diizinkan dan resolusi DNS berfungsi. Untuk setiap titik akhir, *`region`* mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh Amazon ECS, seperti `us-east-2` untuk Wilayah AS Timur (Ohio). Titik akhir untuk semua Wilayah yang Anda gunakan harus diizinkan. Untuk titik akhir `ecs-a` dan `ecs-t`, Anda harus menyertakan tanda bintang (misalnya, `ecs-a-*`).

- `ecs-a-*.`*`region`*`.amazonaws.com`Endpoint ini digunakan saat mengelola tugas.
- `ecs-t-*.`*`region`*`.amazonaws.com`— Titik akhir ini digunakan untuk mengelola metrik tugas dan kontainer.
- `ecs.`*`region`*`.amazonaws.com`- Ini adalah titik akhir layanan untuk Amazon ECS.
- `ssm.`*`region`*`.amazonaws.com` — Ini adalah titik akhir layanan untuk AWS Systems Manager.
- `ec2messages.`*`region`*`.amazonaws.com`— Ini adalah titik akhir layanan yang AWS Systems Manager digunakan untuk berkomunikasi antara agen Systems Manager dan layanan Systems Manager di cloud.
- `ssmmessages.`*`region`*`.amazonaws.com`— Ini adalah titik akhir layanan yang diperlukan untuk membuat dan menghapus saluran sesi dengan layanan Session Manager di cloud.
- Jika tugas Anda memerlukan komunikasi dengan AWS layanan lain, pastikan titik akhir layanan tersebut diizinkan. Contoh aplikasi termasuk menggunakan Amazon ECR untuk menarik gambar

kontainer atau menggunakan CloudWatch untuk CloudWatch Log. Untuk informasi selengkapnya, lihat Titik akhir layanan di Referensi Umum AWS .

Amazon FSx for Windows File Server dengan ECS Di Mana Saja

Untuk menggunakan instans eksternal Amazon FSx for Windows File Server dengan Amazon ECS, Anda harus membuat sambungan antara pusat data lokal dan. AWS Cloud Untuk informasi tentang opsi untuk menghubungkan jaringan ke VPC, lihat Opsi Konektivitas Amazon Virtual Private Cloud.

GMSA dengan ECS Anywhere

Kasus penggunaan berikut didukung untuk ECS Anywhere.

- Direktori Aktif ada di AWS Cloud - Untuk konfigurasi ini, Anda membuat sambungan antara jaringan lokal dan AWS Cloud menggunakan AWS Direct Connect koneksi. Untuk informasi tentang cara membuat sambungan, lihat Opsi Konektivitas Amazon Virtual Private Cloud. Anda membuat Direktori Aktif di AWS Cloud. Untuk informasi tentang cara memulai AWS Directory Service, lihat Menyiapkan AWS Directory Service di Panduan AWS Directory Service Administrasi. Anda kemudian dapat menggabungkan instance eksternal Anda ke domain menggunakan AWS Direct Connect koneksi. Untuk informasi tentang bekerja dengan GMSA dengan Amazon ECS, lihat. the section called “Pelajari cara menggunakan g MSAs untuk wadah EC2 Windows”
- Active Directory berada di pusat data lokal. - Untuk konfigurasi ini, Anda menggabungkan instance eksternal Anda ke Active Directory lokal. Anda kemudian menggunakan kredensil yang tersedia secara lokal saat menjalankan tugas Amazon ECS.

Membuat cluster Amazon ECS untuk tipe peluncuran Eksternal

Anda membuat klaster untuk menentukan infrastruktur tugas dan layanan yang dijalankan.

Sebelum memulai, pastikan Anda telah menyelesaikan langkah-langkah Siapkan untuk menggunakan Amazon ECS dan menetapkan izin IAM yang sesuai. Untuk informasi selengkapnya, lihat the section called “Contoh cluster Amazon ECS”. Konsol Amazon ECS menyediakan cara sederhana untuk membuat sumber daya yang dibutuhkan oleh cluster Amazon ECS dengan membuat AWS CloudFormation tumpukan.

Untuk membuat proses pembuatan cluster semudah mungkin, konsol memiliki pilihan default untuk banyak pilihan yang kami jelaskan di bawah ini. Ada juga panel bantuan yang tersedia untuk porsi besar bagian di konsol yang menyediakan konteks lebih lanjut.

Anda dapat mengubah opsi berikut:

- Tambahkan namespace ke cluster.

  Namespace memungkinkan layanan yang Anda buat di cluster dapat terhubung ke layanan lain di namespace tanpa konfigurasi tambahan. Untuk informasi selengkapnya, lihat Interkoneksi layanan Amazon ECS.
- Konfigurasikan cluster untuk instance eksternal
- Aktifkan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan, atau Wawasan Kontainer.

  CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan”.

  Pada tanggal 2 Desember 2024, AWS merilis Container Insights dengan peningkatan observabilitas untuk Amazon ECS. Versi ini mendukung peningkatan observabilitas untuk cluster Amazon ECS menggunakan jenis peluncuran Amazon dan EC2 Fargate. Setelah mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang disempurnakan di Amazon ECS, Container Insights secara otomatis mengumpulkan telemetri infrastruktur terperinci dari tingkat cluster hingga level container di lingkungan Anda dan menampilkan data Anda di dasbor yang menampilkan berbagai metrik dan dimensi. Anda kemudian dapat menggunakan out-of-the-box dasbor ini di konsol Container Insights untuk lebih memahami kesehatan dan kinerja container Anda, dan untuk mengurangi masalah lebih cepat dengan mengidentifikasi anomali.

  Kami menyarankan Anda menggunakan Wawasan Kontainer dengan observabilitas yang ditingkatkan, bukan Wawasan Kontainer karena ini memberikan visibilitas mendetail di lingkungan penampung Anda, sehingga mengurangi waktu rata-rata untuk resolusi.
- Tetapkan AWS KMS kunci untuk penyimpanan terkelola Anda. Untuk informasi tentang cara membuat kunci, lihat Membuat kunci KMS di Panduan AWS Key Management Service Pengguna.
- Tambahkan tag untuk membantu Anda mengidentifikasi klaster Anda.

Untuk membuat cluster baru (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih Buat klaster.
5. Di bawah konfigurasi Cluster, konfigurasikan berikut ini:
   - Untuk nama Cluster, masukkan nama unik.

     Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
   - (Opsional) Agar namespace yang digunakan untuk Service Connect berbeda dari nama cluster, untuk Namespace, masukkan nama yang unik.
6. (Opsional) Gunakan Wawasan Kontainer, perluas Pemantauan, lalu pilih salah satu opsi berikut:
   - Untuk menggunakan Wawasan Kontainer yang direkomendasikan dengan kemampuan observasi yang ditingkatkan, pilih Wawasan Kontainer dengan observabilitas yang ditingkatkan.
   - Untuk menggunakan Container Insights, pilih Container Insights.
7. (Opsional) Enkripsi data pada penyimpanan terkelola. Di bawah Enkripsi, untuk penyimpanan Terkelola, masukkan ARN AWS KMS kunci yang ingin Anda gunakan untuk mengenkripsi data penyimpanan terkelola.
8. (Opsional) Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:
   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.
9. Pilih Create (Buat).

Langkah selanjutnya

Anda harus mendaftarkan instance dengan cluster. Untuk informasi selengkapnya, lihat Mendaftarkan instans eksternal ke kluster Amazon ECS.

Buat definisi tugas untuk jenis peluncuran eksternal. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol

Jalankan aplikasi Anda sebagai tugas mandiri, atau sebagai bagian dari layanan. Untuk informasi selengkapnya, lihat berikut ini:

- Menjalankan aplikasi sebagai tugas Amazon ECS
- Membuat penyebaran pembaruan bergulir Amazon ECS

Mendaftarkan instans eksternal ke kluster Amazon ECS

Untuk setiap instans eksternal yang Anda daftarkan dengan kluster Amazon ECS, instans tersebut harus memiliki Agen SSM, agen penampung Amazon ECS, dan Docker yang diinstal. Untuk mendaftarkan instans eksternal ke cluster Amazon ECS, instans harus didaftarkan terlebih dahulu sebagai instans AWS Systems Manager terkelola. Anda dapat membuat skrip instalasi dalam beberapa klik pada konsol Amazon ECS. Skrip instalasi mencakup kunci aktivasi Systems Manager dan perintah untuk menginstal masing-masing agen dan Docker yang diperlukan. Skrip penginstalan harus dijalankan di server on-premise atau VM Anda untuk menyelesaikan langkah-langkah instalasi dan pendaftaran.

> **Note**
>
> Sebelum mendaftarkan instance eksternal Linux Anda dengan cluster, buat `/etc/ecs/ecs.config` file pada instance eksternal Anda dan tambahkan parameter konfigurasi agen kontainer apa pun yang Anda inginkan. Anda tidak dapat melakukan hal tersebut setelah mendaftarkan instans eksternal ke sebuah klaster. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman Klaster, pilih sebuah klaster tempat Anda mendaftarkan instans eksternal.
5. Pada *name* halaman Cluster:, pilih tab Infrastruktur.

6. Pada halaman Daftar instans eksternal, selesaikan langkah-langkah berikut.

   a. Untuk Durasi kunci aktivasi (dalam hari), masukkan jumlah hari untuk kunci aktivasi tetap dalam status aktif. Setelah jumlah hari yang Anda masukkan terlewati, kunci tidak lagi berfungsi saat mendaftarkan instans eksternal.

   b. Untuk Jumlah instans, masukkan jumlah instans eksternal yang ingin Anda daftarkan ke klaster Anda dengan kunci aktivasi.

   c. Untuk peran Instance, pilih peran IAM untuk dikaitkan dengan instance eksternal Anda. Jika peran belum dibuat, pilih Buat peran baru agar Amazon ECS membuat peran atas nama Anda. Untuk informasi selengkapnya tentang izin IAM yang diperlukan untuk instans eksternal Anda, lihat. Peran IAM Amazon ECS Anywhere

   d. Salin perintah pendaftaran. Perintah ini harus dijalankan pada setiap instans eksternal yang ingin Anda daftarkan ke klaster.

      > **Important**
      >
      > Bagian bash skrip harus dijalankan sebagai root. Jika perintah tidak dijalankan sebagai root, maka akan terjadi kesalahan.

   e. Pilih Tutup.

AWS CLI for Linux operating systems

1. Buat pasangan aktivasi Systems Manager. Ini digunakan untuk aktivasi instans terkelola Systems Manager. Output meliputi `ActivationId` dan `ActivationCode`. Kedua output tersebut akan digunakan di langkah berikutnya. Pastikan Anda menentukan peran IAM ECS Anywhere yang Anda buat. Untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere.

   ```
   aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
   ```

2. Di server on-premise atau mesin virtual (VM) Anda, unduh skrip instalasi.

   ```
   curl --proto "https" -o "/tmp/ecs-anywhere-install.sh" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"
   ```

3. (Opsional) Di server on-premise atau mesin virtual (VM) Anda, lakukan langkah-langkah berikut untuk memverifikasi penulisan instalasi menggunakan file standar penulisan.

a. Unduh dan instal GnuPG. Untuk informasi selengkapnya GNUpg, lihat situs web GnuPG. Untuk sistem Linux, instal gpg menggunakan pengelola paket pada tipe instans VM Linux Anda.

b. Ambil kunci publik Amazon ECS PGP.

```
gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
```

c. Unduh tanda tangan skrip instalasi. Tanda tangan tersebut merupakan tandatangan PGP ascii terpisah yang disimpan dalam sebuah file dengan ekstensi `.asc`.

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh.asc" "https://amazon-
ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh.asc"
```

d. Verifikasi file skrip instalasi menggunakan kunci.

```
gpg --verify /tmp/ecs-anywhere-install.sh.asc /tmp/ecs-anywhere-install.sh
```

Berikut adalah output yang diharapkan.

```
gpg: Signature made Tue 25 May 2021 07:16:29 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
 owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. Di server on-premise atau mesin virtual (VM) Anda, jalankan skrip instalasi. Tentukan nama cluster, Wilayah, dan ID aktivasi Systems Manager dan kode aktivasi dari langkah pertama.

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE
```

Untuk server lokal atau mesin virtual (VM) yang memiliki driver NVIDIA diinstal untuk beban kerja GPU, Anda harus menambahkan `--enable-gpu` tanda ke skrip instalasi. Ketika tanda

ini ditentukan, skrip penginstalan memverifikasi bahwa driver NVIDIA sedang berjalan dan kemudian menambahkan variabel konfigurasi yang diperlukan untuk menjalankan tugas Amazon ECS Anda. Untuk informasi selengkapnya tentang menjalankan beban kerja GPU dan menentukan persyaratan GPU dalam definisi tugas, lihat. Menentukan GPUs dalam definisi tugas Amazon ECS

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE \
    --enable-gpu
```

Gunakan langkah-langkah berikut untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda.

Untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda

1. Hentikan agen kontainer Amazon ECS.

```
sudo systemctl stop ecs.service
```

2. Edit file `/etc/ecs/ecs.config` dan pada baris `ECS_CLUSTER`, pastikan bahwa nama klaster cocok dengan nama klaster untuk mendaftarkan instans eksternal.
3. Hapus data agen Amazon ECS yang ada.

```
sudo rm /var/lib/ecs/data/agent.db
```

4. Mulai agen kontainer Amazon ECS.

```
sudo systemctl start ecs.service
```

AWS CLI for Windows operating systems

1. Buat pasangan aktivasi Systems Manager. Ini digunakan untuk aktivasi instans terkelola Systems Manager. Output meliputi `ActivationId` dan `ActivationCode`. Kedua output tersebut akan digunakan di langkah berikutnya. Pastikan Anda menentukan peran IAM ECS

Anywhere yang Anda buat. Untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere.

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. Di server on-premise atau mesin virtual (VM) Anda, unduh skrip instalasi.

```
Invoke-RestMethod -URI "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install.ps1" -OutFile “ecs-anywhere-install.ps1”
```

3. (Opsional) Skrip Powershell ditandatangani oleh Amazon dan oleh karena itu, Windows secara otomatis melakukan validasi sertifikat pada saat yang sama. Anda tidak perlu melakukan validasi manual apa pun.

   Untuk memverifikasi sertifikat secara manual, klik kanan pada file, navigasikan ke properti dan gunakan tab Tanda Tangan Digital untuk mendapatkan detail lebih lanjut.

   Opsi ini hanya tersedia ketika tuan rumah memiliki sertifikat di toko sertifikat.

   Verifikasi harus mengembalikan informasi yang serupa dengan yang berikut:

```
# Verification (PowerShell)
Get-AuthenticodeSignature -FilePath .\ecs-anywhere-install.ps1

SignerCertificate                         Status      Path
-----------------                         ------      ----
EXAMPLECERTIFICATE                        Valid       ecs-anywhere-install.ps1

...

Subject              : CN="Amazon Web Services, Inc.",...

----
```

4. Di server on-premise atau mesin virtual (VM) Anda, jalankan skrip instalasi. Tentukan nama cluster, Wilayah, dan ID aktivasi Systems Manager dan kode aktivasi dari langkah pertama.

```
.\ecs-anywhere-install.ps1 -Region $Region -Cluster $Cluster -
ActivationID $ActivationID -ActivationCode $ActivationCode
```

5. Verifikasi bahwa agen penampung Amazon ECS sedang berjalan.

```
Get-Service AmazonECS

Status   Name               DisplayName
------   ----               -----------
Running  AmazonECS          Amazon ECS
```

Gunakan langkah-langkah berikut untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda.

Untuk mendaftarkan instans eksternal yang sudah ada dengan klaster yang berbeda

1. Hentikan agen kontainer Amazon ECS.

   ```
   Stop-Service AmazonECS
   ```

2. Ubah `ECS_CLUSTER` parameter sehingga nama cluster cocok dengan nama cluster untuk mendaftarkan instance eksternal.

   ```
   [Environment]::SetEnvironmentVariable("ECS_CLUSTER", $ECSCluster,
    [System.EnvironmentVariableTarget]::Machine)
   ```

3. Hapus data agen Amazon ECS yang ada.

   ```
   Remove-Item -Recurse -Force $env:ProgramData\Amazon\ECS\data\*
   ```

4. Mulai agen kontainer Amazon ECS.

   ```
   Start-Service AmazonECS
   ```

Ini AWS CLI dapat digunakan untuk membuat aktivasi Systems Manager sebelum menjalankan skrip instalasi untuk menyelesaikan proses pendaftaran instance eksternal.

Membatalkan pendaftaran instans eksternal Amazon ECS

Kami menyarankan Anda membatalkan pendaftaran instans dari Amazon ECS dan AWS Systems Manager setelah Anda selesai dengan instans. Setelah pembatalan pendaftaran, instans eksternal tidak lagi dapat menerima tugas baru.

Jika Anda memiliki tugas yang berjalan di instans kontainer saat membatalkan pendaftarannya, maka tugas tersebut akan tetap berjalan hingga berhenti melalui beberapa cara lain. Namun, tugas-tugas ini tidak lagi dipantau atau diperhitungkan oleh Amazon ECS. Jika tugas-tugas ini pada instans eksternal Anda merupakan bagian dari layanan Amazon ECS, maka penjadwal layanan memulai salinan lain dari tugas itu, pada instance yang berbeda, jika memungkinkan.

Setelah Anda membatalkan pendaftaran instance, bersihkan AWS sumber daya yang tersisa pada instance. Anda kemudian dapat mendaftarkannya ke cluster baru.

Prosedur

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Klaster, kemudian pilih klaster yang meng-host instans eksternal.
4. Pada *name* halaman Cluster:, pilih tab Infrastruktur.
5. Di bawah Instance Container, pilih ID instans eksternal untuk membatalkan pendaftaran. Anda dialihkan ke halaman detail instans kontainer.
6. Pada *id* halaman Container Instance:, pilih Deregister.
7. Tinjau pesan pembatalan pendaftaran. Pilih Deregister from AWS Systems Manager untuk juga membatalkan pendaftaran instance eksternal sebagai instans terkelola Systems Manager. Pilih Batalkan pendaftaran.

   > **Note**
   >
   > Anda dapat membatalkan pendaftaran instans eksternal sebagai instans terkelola Systems Manager di konsol Systems Manager. Untuk petunjuknya, lihat Membatalkan pendaftaran node terkelola di lingkungan hybrid dan multicloud di Panduan Pengguna.AWS Systems Manager

8. Setelah membatalkan pendaftaran instans, bersihkan AWS sumber daya di server lokal atau VM Anda.

Sistem operasi	Langkah-langkah	

Linux	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.
	`sudo systemctl stop ecs amazon-ssm-agent`
	b. Hapus paket Amazon ECS dan Systems Manager.
	Untuk CentOS 7, CentOS 8, dan RHEL 7
	`sudo yum remove -y amazon-ecs-init amazon-ssm-agent`
	Untuk SUSE Enterprise Server 15
	`sudo zypper remove -y amazon-ecs-init amazon-ssm-agent`
	Untuk Debian dan Ubuntu
	`sudo apt remove -y amazon-ecs-init amazon-ssm-agent`
	c. Hapus direktori yang tersisa.	

Sistem operasi	Langkah-langkah
	`sudo rm -rf /var/lib/ecs /etc/ecs /var/lib/amazon/ssm /var/log/ecs /var/log/amazon/ssm`

Windows	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. Hapus paket Amazon ECS.
	`.\ecs-anywhere-install.ps1 -Uninstall`

AWS CLI

1. Anda memerlukan ID instance dan ARN instance kontainer untuk membatalkan pendaftaran instance container. Jika Anda tidak memiliki nilai tesis, jalankan comand berikut

   Jalankan perintah berikutuntuk mendapatkan ID instance.

   Anda menggunakan instance ID (`instanceID`) untuk mendapatkan instance kontainer ARN ()`containerInstanceARN`.

```
instanceId=$(aws ssm describe-instance-information --region "{{ region }}" |
 jq ".InstanceInformationList[] |select(.IPAddress==\"{{ IPv4 Address }}\")
 | .InstanceId" | tr -d'"'
```

Jalankan perintah berikut.

Anda menggunakan parameter `containerInstanceArn` as a dalam perintah untuk membatalkan pendaftaran instance (). `deregister-container-instance`

```
instances=$(aws ecs list-container-instances --cluster "{{ cluster }}" --region
 "{{ region }}" | jq -c '.containerInstanceArns')
containerInstanceArn=$(aws ecs describe-container-instances --cluster
 "{{ cluster }}" --region "{{ region }}" --container-instances $instances
 | jq ".containerInstances[] | select(.ec2InstanceId==\"{{ instanceId }}\")
 | .containerInstanceArn" | tr -d '"')
```

2. Jalankan perintah berikut untuk menguras instance.

```
aws ecs update-container-instances-state --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instances "{{ containerInstanceArn }}" --status
 DRAINING
```

3. Setelah instance container selesai menguras, jalankan perintah berikut untuk membatalkan pendaftaran instance.

```
aws ecs deregister-container-instance --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instance "{{ containerInstanceArn }}"
```

4. Jalankan perintah berikut untuk menghapus instance container dari SSM.

```
aws ssm deregister-managed-instance --region "{{ region }}" --instance-id
 "{{ instanceId }}"
```

5. Setelah membatalkan pendaftaran instans, bersihkan AWS sumber daya di server lokal atau VM Anda.

Sistem operasi	Langkah-langkah	
Linux	a. Hentikan agen kontainer Amazon ECS dan	

Sistem operasi	Langkah-langkah	

	layanan Agen SSM pada instans.
	`sudo systemctl stop ecs amazon-ssm-agent`
	b. Hapus paket Amazon ECS dan Systems Manager.
	`sudo (yum/apt/zypper) remove amazon-ecs-init amazon-ssm-agent`
	c. Hapus direktori yang tersisa.
	`sudo rm -rf /var/lib/ecs /etc/ecs /var/lib/amazon/ssm /var/log/ecs /var/log/amazon/ssm`	

Sistem operasi	Langkah-langkah	

Windows	a. Hentikan agen kontainer Amazon ECS dan layanan Agen SSM pada instans.
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. Hapus paket Amazon ECS.
	`.\ecs-anywhere-install.ps1 -Uninstall`	

Memperbarui AWS Systems Manager agen dan agen penampung Amazon ECS pada instance eksternal

Server lokal atau VM Anda harus menjalankan AWS Systems Manager Agen (Agen SSM) dan agen penampung Amazon ECS saat menjalankan beban kerja Amazon ECS. AWS merilis versi baru dari agen ini ketika kemampuan apa pun ditambahkan atau diperbarui. Jika instans eksternal Anda menggunakan versi yang lebih lama dari salah satu agen, maka Anda dapat memperbaruinya menggunakan prosedur berikut.

Memperbarui SSM Agent pada instans eksternal

AWS Systems Manager merekomendasikan agar Anda mengotomatiskan proses memperbarui Agen SSM pada instans Anda. Beberapa metode disediakan untuk mengotomatisasi pembaruan. Untuk informasi selengkapnya, lihat Mengotomatiskan pembaruan ke Agen SSM di AWS Systems Manager Panduan Pengguna.

Memperbarui agen Amazon ECS pada instans eksternal

Pada instans eksternal Anda, agen penampung Amazon ECS diperbarui dengan memutakhirkan paket. `ecs-init` Memperbarui agen Amazon ECS tidak mengganggu tugas atau layanan yang sedang berjalan. Amazon ECS menyediakan `ecs-init` paket dan file tanda tangan dalam bucket Amazon S3 di setiap Wilayah. Dimulai dengan `ecs-init` versi`1.52.1-1`, Amazon ECS menyediakan `ecs-init` paket terpisah untuk digunakan tergantung pada sistem operasi dan arsitektur sistem yang digunakan instans eksternal Anda.

Gunakan tabel berikut untuk menentukan paket `ecs-init` yang harus Anda unduh berdasarkan sistem pengoperasian dan arsitektur sistem yang digunakan instans eksternal Anda.

> **Note**
>
> Anda dapat menentukan sistem pengoperasian dan sistem arsitektur yang menggunakan instans eksternal Anda dengan menggunakan perintah berikut.
>
> ```
> cat /etc/os-release
> uname -m
> ```

Sistem pengoperasian (arsitektur)	Paket ecs-init
CentOS 7 (x86_64) CentOS 8 (x86_64) CentOS Aliran 9 (x86_64) SUSE Enterprise Server 15 (x86_64) RHEL 7 (x86_64) RHEL 8 (x86_64)	`amazon-ecs-init-latest.x86_64.rpm`
CentOS 7 (aarch64) CentOS 8 (aarch64) CentOS Stream 9 (aarch64)	`amazon-ecs-init-latest.aarch64.rpm`

Sistem pengoperasian (arsitektur)	Paket ecs-init
RHEL 7 (aarch64)	
Debian 9 (x86_64) Debian 10 (x86_64) Debian 11 (x86_64) Debian 12 (x86_64) Ubuntu 18 (x86_64) Ubuntu 20 (x86_64) Ubuntu 22 (x86_64) Ubuntu 24 (x86_64)	`amazon-ecs-init-latest.amd64.deb`
Debian 9 (aarch64) Debian 10 (aarch64) Debian 11 (aarch64) Debian 12 (aarch64) Ubuntu 18 (aarch64) Ubuntu 20 (aarch64) Ubuntu 22 (aarch64) Ubuntu 24 (aarch64)	`amazon-ecs-init-latest.arm64.deb`

Ikuti langkah-langkah ini untuk memperbarui agen Amazon ECS.

Untuk memperbarui agen Amazon ECS

1. Konfirmasikan versi agen Amazon ECS yang sedang Anda jalankan.

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

2. Unduh `ecs-init` paket untuk sistem operasi dan arsitektur sistem Anda. Amazon ECS menyediakan file `ecs-init` paket dalam bucket Amazon S3 di setiap Wilayah. Pastikan Anda mengganti *<region>* pengenal dalam perintah dengan nama Region (misalnya,`us-west-2`) yang secara geografis paling dekat dengan Anda.

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm
   ```

   amazon-ecs-init-latest.aarch64.rpm

   ```
   curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm
   ```

   amazon-ecs-init-latest.amd64.deb

   ```
   curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb
   ```

   amazon-ecs-init-latest.arm64.deb

   ```
   curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb
   ```

3. (Opsional) Verifikasi validitas file paket `ecs-init` menggunakan tanda tangan PGP.

   a. Unduh dan instal GnuPG. Untuk informasi selengkapnya GNUpg, lihat situs web GnuPG. Untuk sistem Linux, instal gpg menggunakan pengelola paket pada tipe instans VM Linux Anda.

   b. Ambil kunci publik Amazon ECS PGP.

      ```
      gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
      ```

   c. Unduh tanda tangan paket `ecs-init`. Tanda tangan adalah tanda tangan PGP terpisah ASCII yang disimpan dalam file dengan ekstensi. `.asc` Amazon ECS menyediakan file

tanda tangan dalam bucket Amazon S3 di setiap Wilayah. Pastikan Anda mengganti *<region>* pengenal dalam perintah dengan nama Region (misalnya,`us-west-2`) yang secara geografis paling dekat dengan Anda.

amazon-ecs-init-latest.x86_64.rpm

```
curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm.asc
```

amazon-ecs-init-latest.aarch64.rpm

```
curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm.asc
```

amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb.asc
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb.asc
```

d. Verifikasi file paket `ecs-init` menggunakan kunci.

Untuk **`rpm`** paket-paketnya

```
gpg --verify amazon-ecs-init.rpm.asc ./amazon-ecs-init.rpm
```

Untuk **`deb`** paket-paketnya

```
gpg --verify amazon-ecs-init.deb.asc ./amazon-ecs-init.deb
```

Berikut adalah output yang diharapkan.

```
gpg: Signature made Fri 14 May 2021 09:31:36 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
```

```
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. Instal paket `ecs-init`.

   Untuk **`rpm`** paket di CentOS 7, CentOS 8, dan RHEL 7

   ```
   sudo yum install -y ./amazon-ecs-init.rpm
   ```

   Untuk **`rpm`** paket di SUSE Enterprise Server 15

   ```
   sudo zypper install -y --allow-unsigned-rpm ./amazon-ecs-init.rpm
   ```

   Untuk **`deb`** paketnya

   ```
   sudo dpkg -i ./amazon-ecs-init.deb
   ```

5. Mulai ulang layanan `ecs`.

   ```
   sudo systemctl restart ecs
   ```

6. Verifikasi bahwa versi agen Amazon ECS telah diperbarui.

   ```
   curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
   ```

Memperbarui kluster Amazon ECS

Anda dapat memodifikasi properti cluster berikut:

- Tetapkan penyedia kapasitas default

  Setiap cluster dapat memiliki satu atau lebih penyedia kapasitas dan strategi penyedia kapasitas opsional. Strategi penyedia kapasitas menentukan bagaimana tugas-tugas tersebar di penyedia kapasitas klaster ini. Saat menjalankan tugas mandiri atau membuat layanan, Anda dapat menggunakan strategi penyedia kapasitas default kluster atau strategi penyedia kapasitas yang mengganti tugas default.
- Aktifkan Wawasan Kontainer.

CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Container Insights juga menyediakan informasi diagnostik, seperti kegagalan restart kontainer, yang Anda gunakan untuk mengisolasi masalah dan menyelesaikannya dengan cepat. Untuk informasi selengkapnya, lihat the section called “Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan”.

- Tambahkan tag untuk membantu Anda mengidentifikasi cluster Anda.

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada *name* halaman Cluster:, pilih Update cluster.
5. Untuk mengatur penyedia kapasitas default, di bawah Strategi penyedia kapasitas default, pilih Tambahkan lebih banyak.

   a. Untuk penyedia Kapasitas, pilih penyedia kapasitas.

   b. (Opsional) Untuk Base, masukkan jumlah minimum tugas yang dijalankan pada penyedia kapasitas.

      Anda hanya dapat menetapkan nilai Dasar untuk satu penyedia kapasitas.

   c. (Opsional) Untuk Berat, masukkan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan.

   d. (Opsional) Ulangi langkah-langkah untuk penyedia kapasitas tambahan.
6. Untuk mengaktifkan atau menonaktifkan Container Insights, perluas Monitoring, lalu aktifkan Use Container Insights.
7. Untuk membantu mengidentifikasi klaster Anda, perluas Tag, lalu konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag dan lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.

   [Hapus tanda] Pilih Hapus di sebelah kanan Kunci dan Nilai tanda.

8. Pilih Perbarui.

Menghapus kluster Amazon ECS

Jika Anda selesai menggunakan klaster, Anda dapat menghapusnya. Setelah Anda menghapus cluster, itu transisi ke `INACTIVE` status. Klaster dengan status `INACTIVE` mungkin tetap dapat ditemukan di akun Anda selama jangka waktu tertentu. Namun, perilaku ini dapat berubah di masa depan, jadi sebaiknya Anda tidak mengandalkan klaster `INACTIVE` yang tersimpan.

Sebelum Anda menghapus cluster, Anda harus melakukan operasi berikut:

- Hapus semua layanan di cluster. Untuk informasi selengkapnya, lihat the section called “Menghapus layanan”.
- Hentikan semua tugas yang sedang berjalan. Untuk informasi selengkapnya, lihat the section called “Menghentikan tugas”.
- Deregister semua instance kontainer terdaftar di cluster. Untuk informasi selengkapnya, lihat the section called “Membatalkan pendaftaran instance kontainer”.
- Hapus namespace. Untuk informasi selengkapnya, lihat Menghapus namespace di Panduan Developer AWS Cloud Map .

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Klaster.
4. Pada halaman klaster, pilih klaster untuk menghapus.
5. Di bagian kanan atas halaman, pilih Hapus Klaster.

   Pesan ditampilkan ketika Anda tidak menghapus semua sumber daya yang terkait dengan cluster.
6. Di kotak konfirmasi, masukkan hapus ***`cluster name`***.

Membatalkan pendaftaran instans kontainer Amazon ECS

> **Important**
>
> Topik ini EC2 hanya untuk instance container yang dibuat di Amazon. Untuk informasi lebih lanjut tentang membatalkan pendaftaran instans eksternal, lihat Membatalkan pendaftaran instans eksternal Amazon ECS.

Setelah selesai dengan instance penampung yang EC2 didukung Amazon, Anda harus membatalkan pendaftarannya dari klaster Anda. Mengikuti pembatalan pendaftaran, instans kontainer tidak lagi mampu menerima tugas baru.

Jika Anda memiliki tugas yang berjalan pada instans kontainer saat membatalkan pendaftaran tersebut, maka tugas ini tetap berjalan sampai Anda mengakhiri instans atau tugas berhenti melalui beberapa cara lain. Namun, tugas-tugas ini menjadi yatim piatu yang berarti mereka tidak lagi dipantau atau diperhitungkan oleh Amazon ECS. Jika tugas yatim piatu pada instance container Anda adalah bagian dari layanan Amazon ECS, maka penjadwal layanan akan memulai salinan lain dari tugas tersebut, pada instance container yang berbeda, jika memungkinkan. Setiap kontainer dalam tugas layanan yatim piatu yang terdaftar pada kelompok sasaran Application Load Balancer dideregistrasi. Kontainer tersebut memulai menguras koneksi sesuai dengan pengaturan pada penyeimbang beban atau grup target. Jika tugas tanpa sumber menggunakan mode jaringan `awsvpc`, maka antarmuka jaringan elastis tugas tersebut dihapus.

Jika Anda ingin menggunakan instans kontainer untuk beberapa tujuan lain setelah pembatalan pendaftaran, Anda harus menghentikan semua tugas yang berjalan pada instans kontainer sebelum pembatalan pendaftaran. Penghentian ini menghentikan tugas tanpa sumber mengonsumsi sumber daya.

Saat membatalkan pendaftaran sebuah instans kontainer, pertimbangan beberapa hal berikut.

- Karena setiap instans kontainer memiliki status informasi yang unik, maka instans tersebut tidak boleh dibatalkan pendaftarannya dari suatu klaster kemudian di didaftarkan kembali di klaster lain. Untuk memindahkan sumber daya instans kontainer, kami menyarankan Anda untuk mengakhiri instans kontainer dari suatu klaster kemudian meluncurkan instans kontainer baru di klaster baru. Untuk informasi selengkapnya, lihat Mengakhiri instans Anda di Panduan EC2 Pengguna Amazon danMeluncurkan instans penampung Amazon ECS Linux.

- Jika instance container dikelola oleh grup Auto Scaling atau AWS CloudFormation tumpukan, hentikan instance dengan memperbarui grup atau tumpukan Auto Scaling. AWS CloudFormation Jika tidak, grup Auto Scaling atau AWS CloudFormation akan membuat instance baru setelah Anda menghentikannya.
- Jika Anda menghentikan instance container yang sedang berjalan dengan agen container Amazon ECS yang terhubung, agen akan secara otomatis membatalkan pendaftaran instance dari klaster Anda. Instans kontainer yang dihentikan atau instans dengan agen yang tidak terkoneksi tidak secara otomatis dibatalkan pendaftarannya saat diakhiri.
- Membatalkan pendaftaran instance container akan menghapus instance dari klaster, tetapi instans Amazon tidak dihentikan. EC2 Jika Anda selesai menggunakan instans, pastikan Anda mengakhirinya untuk menghentikan penagihan. Untuk informasi selengkapnya, lihat Menghentikan instans Anda di Panduan EC2 Pengguna Amazon.

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih Wilayah tempat instans eksternal Anda didaftarkan.
3. Di panel navigasi, pilih Clusters dan pilih cluster yang menghosting instance.
4. Pada *name* halaman Cluster:, pilih tab Infrastruktur.
5. Di bawah Instance Container, pilih ID instance untuk deregister. Anda dialihkan ke halaman detail instans kontainer.
6. Pada *id* halaman Container Instance:, pilih Deregister.
7. Pada layar konfirmasi, pilih Deregister.
8. Jika Anda sudah selesai dengan instance container, hentikan EC2 instance Amazon yang mendasarinya. Untuk informasi selengkapnya, lihat Menghentikan Instans Anda di Panduan EC2 Pengguna Amazon.

Menguras instans kontainer Amazon ECS

Mungkin ada saat-saat ketika Anda perlu menghapus instance container dari cluster Anda, misalnya, untuk melakukan pembaruan sistem atau untuk mengurangi kapasitas cluster. Amazon ECS menyediakan kemampuan untuk mentransisikan instans kontainer ke `DRAINING` status. Ini disebut sebagai pengurasan instans kontainer. Saat instance container disetel ke`DRAINING`, Amazon ECS mencegah tugas baru dijadwalkan untuk penempatan pada instance container.

Menguras perilaku untuk layanan

Tugas yang merupakan bagian dari layanan yang ada di status `PENDING` dihentikan dengan segera. Jika terdapat kapasitas instans kontainer dalam klaster, penjadwal layanan akan memulai tugas penggantian. Jika kapasitas instans kontainer tidak cukup, pesan peristiwa layanan yang menunjukkan masalah tersebut akan dikirim.

Tugas yang merupakan bagian dari layanan pada instans kontainer yang berada di status `RUNNING` dialihkan ke status `STOPPED`. Penjadwal layanan mencoba mengganti tugas sesuai dengan jenis penyebaran layanan dan parameter konfigurasi penerapan, dan. `minimumHealthyPercent` `maximumPercent` Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS dan Parameter definisi layanan Amazon ECS.

- Jika `minimumHealthyPercent` di bawah 100%, penjadwal dapat mengabaikan `desiredCount` untuk sementara selama penggantian tugas. Misalnya, `desiredCount` adalah empat tugas, setidaknya 50% mengizinkan penjadwal menghentikan dua tugas yang ada sebelum memulai dua tugas baru. Jika minimumnya adalah 100%, penjadwal layanan tidak dapat menghapus tugas yang ada hingga tugas penggantian dianggap sehat. Jika tugas untuk layanan yang tidak menggunakan penyeimbang beban berada di status `RUNNING`, maka mereka dianggap sehat. Tugas untuk layanan yang menggunakan penyeimbang beban dianggap sehat jika berada di status `RUNNING` dan instans kontainer tempat mereka di-host dilaporkan dalam kondisi baik oleh penyeimbang beban.

  > **Important**
  >
  > Jika Anda menggunakan Instans Spot dan `minimumHealthyPercent` lebih besar dari atau sama dengan 100%, maka layanan tidak akan memiliki cukup waktu untuk mengganti tugas sebelum Instans Spot berakhir.

- `maximumPercent`Parameter mewakili batas atas jumlah tugas yang berjalan selama penggantian tugas, yang memungkinkan Anda menentukan ukuran batch pengganti. Contohnya, jika `desiredCount` dari empat tugas, maksimal 200% memulai empat tugas baru sebelum menghentikan empat tugas yang akan dikuras (asalkan sumber daya klaster yang diperlukan untuk melakukan ini tersedia). Jika maksimumnya adalah 100%, maka tugas penggantian tidak dapat dimulai hingga tugas pengeringan berhenti.

> **Important**
>
> Jika `maximumPercent` keduanya `minimumHealthyPercent` dan 100%, maka layanan tidak dapat menghapus tugas yang ada, dan juga tidak dapat memulai tugas penggantian. Ini mencegah pengeringan instance kontainer yang berhasil dan mencegah pembuatan penerapan baru.

Menguras perilaku untuk tugas mandiri

Setiap tugas mandiri di status `PENDING` atau `RUNNING` tidak terpengaruh; Anda harus menunggu tugas tersebut berhenti sendiri atau menghentikannya secara manual. Instance kontainer akan tetap dalam `DRAINING` status.

Sebuah instans kontainer telah selesai menguras ketika semua tugas yang berjalan di ditransisi ke status `STOPPED`. Instans kontainer tetap dalam status `DRAINING` hingga diaktifkan kembali atau dihapus. Anda dapat memverifikasi status tugas pada instance container dengan menggunakan ListTasksoperasi dengan `containerInstance` parameter untuk mendapatkan daftar tugas pada instance diikuti dengan DescribeTasksoperasi dengan Amazon Resource Name (ARN) atau ID dari setiap tugas untuk memverifikasi status tugas.

Ketika Anda merasa instans kontainer siap untuk memulai tugas menghosting kembali, maka Anda mengubah status instans kontainer dari `DRAINING` ke `ACTIVE`. Penjadwal layanan Amazon ECS kemudian akan mempertimbangkan instance kontainer untuk penempatan tugas lagi.

Prosedur

Langkah-langkah berikut dapat digunakan untuk mengatur instance kontainer ke pengeringan menggunakan yang baru AWS Management Console.

Anda juga dapat menggunakan tindakan UpdateContainerInstancesStateAPI atau update-container-instances-stateperintah untuk mengubah status instance container menjadi`DRAINING`.

AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih klaster yang meng-host instance Anda.

4. Pada *name* halaman Cluster:, pilih tab Infrastruktur. Kemudian, di bawah Contoh kontainer pilih kotak centang untuk setiap instance kontainer yang ingin Anda tiriskan.
5. Pilih Tindakan, Tiriskan.

Instans EC2 Kontainer Amazon ECS

Agen Amazon ECS adalah proses yang berjalan pada setiap instance container yang terdaftar di klaster Anda. Ini memfasilitasi komunikasi antara instans kontainer Anda dan Amazon ECS.

> **Note**
>
> Pada instance kontainer Linux, container agen memasang direktori tingkat atas seperti`/lib`,, dan. `/lib64` `/proc` Ini diperlukan untuk fitur dan fungsi ECS seperti volume Amazon EBS, mode `awsvpc` jaringan, Amazon ECS Service Connect, dan FireLens untuk Amazon ECS.

Setiap versi agen penampung Amazon ECS mendukung set fitur yang berbeda dan menyediakan perbaikan bug dari versi sebelumnya. Jika memungkinkan, kami selalu merekomendasikan menggunakan versi terbaru dari agen kontainer Amazon ECS. Untuk memperbarui agen kontainer ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Agen kontainer Amazon ECS berisi `amazon-ecs-pause` gambar.Amazon ECS menggunakan gambar ini untuk tugas-tugas yang menggunakan mode jaringan. `awsvpc`

Untuk melihat fitur dan penyempurnaan mana yang disertakan dengan setiap rilis agen, lihat https://github.com/aws/ amazon-ecs-agent /releases.

> **Important**
>
> Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
> Versi agen Amazon ECS `1.20.0` dan yang lebih baru telah menghentikan dukungan untuk versi Docker yang lebih lama dari. `18.01.0`

Siklus hidup

Saat agen penampung Amazon ECS mendaftarkan EC2 instans Amazon ke klaster Anda, EC2 instans Amazon melaporkan statusnya sebagai `ACTIVE` dan status koneksi agennya sebagai. `TRUE` Instance container ini dapat menerima permintaan tugas run.

Jika Anda menghentikan (tidak menghentikan) instance kontainer, statusnya tetap ada`ACTIVE`, tetapi status koneksi agen bertransisi ke `FALSE` dalam beberapa menit. Tugas yang berjalan pada instans kontainer berhenti. Jika Anda memulai instance kontainer lagi, agen kontainer akan terhubung kembali dengan layanan Amazon ECS, dan Anda dapat menjalankan tugas pada instance lagi.

> **Important**
>
> Jika Anda menghentikan dan memulai instance container, atau me-reboot instance tersebut, beberapa versi lama dari agen penampung Amazon ECS akan mendaftarkan instance tersebut lagi tanpa membatalkan pendaftaran ID instance container asli. Dalam hal ini, Amazon ECS mencantumkan lebih banyak instance container di cluster Anda daripada yang sebenarnya Anda miliki. (Jika Anda memiliki instance penampung duplikat IDs untuk ID EC2 instans Amazon yang sama, Anda dapat dengan aman membatalkan pendaftaran duplikat yang terdaftar sebagai `ACTIVE` dengan status koneksi agen.) `FALSE` Masalah ini telah diperbaiki dalam versi agen kontainer Amazon ECS saat ini. Untuk informasi selengkapnya tentang memperbarui versi CNI Anda, lihat Memperbarui agen kontainer Amazon ECS.

Jika Anda mengubah status instans kontainer ke `DRAINING`, maka tugas baru tidak ditempatkan pada instans kontainer. Tugas layanan yang berjalan pada instans kontainer akan dihapus jika memungkinkan, agar Anda dapat melakukan pembaruan sistem. Untuk informasi selengkapnya, lihat Menguras instans kontainer Amazon ECS.

Jika Anda membatalkan pendaftaran atau mengakhiri instans kontainer, maka status instans kontainer seketika berubah menjadi `INACTIVE`, dan instans kontainer tidak lagi dilaporkan ketika Anda mencantumkan instans kontainer Anda. Namun, Anda tetap dapat menjelaskan instans kontainer selama satu jam setelah penghentian. Setelah satu jam, deskripsi instans tidak lagi tersedia.

> **Important**
>
> Anda dapat menguras instance secara manual, atau membuat hook siklus hidup grup Auto Scaling untuk menyetel status instans. `DRAINING` Lihat kait siklus hidup EC2 Auto Scaling Amazon untuk informasi selengkapnya tentang kait siklus hidup Auto Scaling.

Support Docker

Amazon ECS mendukung dua versi utama terakhir dari Docker yang diterbitkan di Amazon Linux. Saat ini, ini termasuk Docker 20.10.x dan Docker 25.x.

Versi Docker minimum yang diperlukan untuk Amazon ECS dapat ditemukan di file spesifikasi Amazon ECS Agent. GitHub

Saat menggunakan AMI Amazon ECS yang dioptimalkan, Docker sudah diinstal sebelumnya dan dikonfigurasi untuk bekerja dengan agen penampung Amazon ECS. AMI menyertakan versi Docker yang diuji dan didukung oleh Amazon ECS.

> **Note**
>
> Meskipun Amazon ECS mendukung beberapa versi Docker, sebaiknya gunakan versi Docker yang disertakan dengan AMI Amazon ECS yang dioptimalkan untuk kompatibilitas dan dukungan terbaik.

AMI Amazon ECS yang dioptimalkan

Varian Linux dari AMI Amazon ECS yang dioptimalkan menggunakan Amazon Linux 2 AMI sebagai basis mereka. Nama AMI sumber Amazon Linux 2 untuk setiap varian dapat diambil dengan menanyakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan. Saat Anda meluncurkan instans kontainer kami dari Amazon Linux 2 AMI Amazon ECS terbaru yang dioptimalkan Amazon ECS, Anda menerima versi agen kontainer saat ini. Untuk meluncurkan instans kontainer dengan Amazon Linux 2 AMI Amazon ECS terbaru yang dioptimalkan oleh Amazon ECS, lihat. Meluncurkan instans penampung Amazon ECS Linux

Informasi tambahan

Halaman-halaman berikut memberikan informasi tambahan tentang perubahan:

- Amazon ECS Agent changelog aktif GitHub
- Kode sumber untuk `ecs-init` aplikasi dan skrip dan konfigurasi untuk pengemasan agen sekarang menjadi bagian dari repositori agen. Untuk versi `ecs-init` dan kemasan yang lebih lama, lihat Amazon ecs-init changelog di GitHub
- Catatan rilis Amazon Linux 2.
- Catatan rilis Docker Engine dalam dokumentasi Docker
- Dokumentasi Driver NVIDIA dalam dokumentasi NVIDIA

Konfigurasi agen kontainer Amazon ECS

Berlaku untuk: EC2 instance

Agen penampung Amazon ECS mendukung sejumlah opsi konfigurasi, yang sebagian besar Anda tetapkan melalui variabel lingkungan.

Jika instance container Anda diluncurkan dengan varian Linux dari AMI Amazon ECS yang dioptimalkan, Anda dapat mengatur variabel lingkungan ini dalam `/etc/ecs/ecs.config` file dan kemudian memulai ulang agen. Anda juga dapat menulis variabel konfigurasi ini ke instance container Anda dengan data EC2 pengguna Amazon pada waktu peluncuran. Untuk informasi selengkapnya, lihat Bootstrapping instance container Amazon ECS Linux untuk meneruskan data.

Jika instance container Anda diluncurkan dengan varian Windows dari AMI Amazon ECS yang dioptimalkan, Anda dapat mengatur variabel lingkungan ini dengan PowerShell SetEnvironmentVariable perintah dan kemudian memulai ulang agen. Untuk informasi selengkapnya, lihat Menjalankan perintah saat Anda meluncurkan EC2 instance dengan input data pengguna di Panduan EC2 Pengguna Amazon danthe section called “Contoh wadah bootstrapping”.

Jika Anda secara manual memulai agen penampung Amazon ECS (untuk non-Amazon ECS yang dioptimalkan AMIs), Anda dapat menggunakan variabel lingkungan ini dalam docker run perintah yang Anda gunakan untuk memulai agen. Gunakan variabel ini dengan `--env=`*`VARIABLE_NAME`*`=`*`VARIABLE_VALUE`* sintaksis. Untuk informasi sensitif, seperti kredensial autentikasi untuk repositori privat, Anda harus menyimpan variabel lingkungan agen Anda dalam file dan melakukan semuanya pada satu waktu dengan opsi `--env-file` *`path_to_env_file`*. Anda dapat menggunakan perintah berikut untuk menambahkan variabel.

```
sudo systemctl stop ecs
sudo vi /etc/ecs/ecs.config
# And add the environment variables with VARIABLE_NAME=VARIABLE_VALUE format.
sudo systemctl start ecs
```

Jalankan agen Amazon ECS dengan namespace PID host

Secara default, agen Amazon ECS berjalan dengan namespace PID-nya sendiri. Dalam konfigurasi berikut, Anda dapat mengonfigurasi agen Amazon ECS agar berjalan dengan namespace PID host:

- SELinux mode penegakan diaktifkan.
- Kebijakan SELinux keamanan Docker disetel ke true.

Anda dapat mengonfigurasi perilaku ini dengan menyetel variabel `ECS_AGENT_PID_NAMESPACE_HOST` lingkungan ke `true` dalam `/etc/ecs/ecs.config` file Anda. Ketika variabel ini diaktifkan, `ecs-init` akan memulai wadah agen Amazon ECS dengan namespace PID host (`--pid=host`), memungkinkan agen untuk mem-bootstrap sendiri dengan benar di SELinux lingkungan -enforcing. Fitur ini tersedia dalam versi agen Amazon ECS `1.94.0` dan yang lebih baru.

Untuk mengaktifkan fitur ini, tambahkan baris berikut ke `/etc/ecs/ecs.config` file Anda:

```
ECS_AGENT_PID_NAMESPACE_HOST=true
```

Setelah melakukan perubahan ini, restart agen Amazon ECS agar perubahan diterapkan:

```
sudo systemctl restart ecs
```

Fitur berikut tidak akan berfungsi, mode SELinux penegakan diaktifkan dan kebijakan keamanan Docker disetel ke true, bahkan saat `ECS_AGENT_PID_NAMESPACE_HOST=true` disetel.

- Amazon ECS Exec
- Lampirkan tugas Amazon EBS
- Layanan Connect
- FireLensuntuk Amazon ECS

Parameter yang tersedia

Untuk informasi tentang parameter konfigurasi agen kontainer Amazon ECS yang tersedia, lihat Agen Kontainer Amazon ECS di. GitHub

Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3

Konfigurasi agen kontainer Amazon ECS dikendalikan dengan variabel lingkungan. Varian Linux dari AMI yang dioptimalkan Amazon ECS mencari variabel-variabel ini `/etc/ecs/ecs.config` saat agen penampung memulai dan mengonfigurasi agen yang sesuai. Variabel lingkungan non-sensatif, sepertiECS_CLUSTER, dapat diteruskan ke instance container saat diluncurkan melalui data EC2 pengguna Amazon dan ditulis ke file ini tanpa konsekuensi. Namun, informasi sensitif lainnya, seperti AWS kredensi Anda atau ECS_ENGINE_AUTH_DATA variabel, tidak boleh diteruskan ke instance dalam data pengguna atau ditulis dengan cara yang memungkinkan mereka muncul dalam file. `/etc/ecs/ecs.config` `.bash_history`

Menyimpan informasi konfigurasi dalam bucket pribadi di Amazon S3 dan memberikan akses hanya-baca ke peran IAM instance container Anda adalah cara yang aman dan nyaman untuk mengizinkan konfigurasi instance container saat diluncurkan. Anda dapat menyimpan salinan file `ecs.config` dalam bucket privat. Anda kemudian dapat menggunakan data EC2 pengguna Amazon untuk menginstal AWS CLI dan menyalin informasi konfigurasi Anda `/etc/ecs/ecs.config` saat instance diluncurkan.

Untuk menyimpan **`ecs.config`** file di Amazon S3

1. Anda harus memberikan izin container instance role (ecsInstanceRole) agar hanya dapat membaca akses ke Amazon S3. Anda dapat melakukan ini dengan menetapkan AmazonS3 ReadOnlyAccess ke peran tersebut. `ecsInstanceRole` Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memperbarui izin untuk peran di AWS Identity and Access Management Panduan Pengguna
2. Buat `ecs.config` file dengan variabel konfigurasi agen Amazon ECS yang valid menggunakan format berikut. Contoh ini mengonfigurasi autentikasi registri privat. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

> **Note**
>
> Untuk daftar lengkap variabel konfigurasi agen Amazon ECS yang tersedia, lihat Agen Kontainer Amazon ECS di. GitHub

3. Untuk menyimpan file konfigurasi Anda, buat bucket pribadi di Amazon S3. Untuk informasi selengkapnya, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
4. Unggah file `ecs.config` ke bucket S3 Anda. Untuk informasi selengkapnya, lihat Mengunggah objek di Panduan Pengguna Amazon Simple Storage Service.

Untuk memuat **`ecs.config`** file dari Amazon S3 saat peluncuran

1. Selesaikan prosedur sebelumnya di bagian ini untuk mengizinkan akses Amazon S3 hanya-baca ke instans penampung Anda dan simpan `ecs.config` file di bucket S3 pribadi.
2. Luncurkan instance container baru dan gunakan contoh script berikut dalam data EC2 User. Skrip menginstal AWS CLI dan menyalin file konfigurasi Anda ke`/etc/ecs/ecs.config`. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

Memasang agen kontainer Amazon ECS

Jika Anda ingin mendaftarkan EC2 instans Amazon dengan cluster Amazon ECS Anda dan instans tersebut tidak menggunakan AMI berdasarkan AMI yang dioptimalkan Amazon ECS, Anda dapat menginstal agen penampung Amazon ECS secara manual menggunakan prosedur berikut. Untuk melakukan ini, Anda dapat mengunduh agen dari salah satu bucket Amazon S3 regional atau dari Amazon Elastic Container Registry Public. Jika Anda mengunduh dari salah satu bucket Amazon S3 regional, Anda dapat memverifikasi validitas file agen kontainer secara opsional menggunakan tanda tangan PGP.

> **Note**
>
> `systemd`Unit untuk layanan Amazon ECS dan Docker memiliki arahan untuk menunggu `cloud-init` hingga selesai sebelum memulai kedua layanan. `cloud-init`Proses ini tidak dianggap selesai sampai data EC2 pengguna Amazon Anda selesai berjalan. Oleh karena itu, memulai Amazon ECS atau Docker melalui data EC2 pengguna Amazon dapat menyebabkan kebuntuan. Untuk memulai agen penampung menggunakan data EC2 pengguna Amazon yang dapat Anda gunakan`systemctl enable --now --no-block ecs.service`.

Menginstal agen kontainer Amazon ECS pada non-instance Amazon Linux EC2

Untuk menginstal agen penampung Amazon ECS pada EC2 instans Amazon, Anda dapat mengunduh agen dari salah satu bucket Amazon S3 regional dan menginstalnya.

> **Note**
>
> Saat menggunakan AMI Linux non-Amazon, EC2 instans Amazon Anda memerlukan `cgroupfs` dukungan untuk `cgroup` driver agar agen Amazon ECS mendukung batas sumber daya tingkat tugas. Untuk informasi selengkapnya, lihat agen Amazon ECS di GitHub.

File agen kontainer Amazon ECS terbaru, menurut Wilayah, untuk setiap arsitektur sistem tercantum di bawah ini untuk referensi.

Wilayah	Nama wilayah	Amazon ECS memanaskan file deb	Amazon ECS memanaskan file rpm
us-east-2	AS Timur (Ohio)	Amazon ECS panas amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
us-east-1	AS Timur (Virginia Utara)	Amazon ECS panas amd64 (amd64)	Amazon ECS init x86_64 (x86_64)

Wilayah	Nama wilayah	Amazon ECS memanaskan file deb	Amazon ECS memanaskan file rpm
		Amazon ECS panas arm64 (arm64)	Amazon ECS panas aarch64 (aarch64)
us-west-1	AS Barat (California Utara)	Amazon ECS panas amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
us-west-2	AS Barat (Oregon)	Amazon ECS panas amd64 (amd64) Amazon ECS panas arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-east-1	Asia Pasifik (Hong Kong)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-northeast-1	Asia Pasifik (Tokyo)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-northeast-2	Asia Pasifik (Seoul)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS memanaskan file deb	Amazon ECS memanaskan file rpm
ap-south-1	Asia Pasifik (Mumbai)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-southeast-1	Asia Pasifik (Singapura)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ap-southeast-2	Asia Pasifik (Sydney)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
ca-central-1	Kanada (Pusat)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-central-1	Eropa (Frankfurt)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS memanaskan file deb	Amazon ECS memanaskan file rpm
eu-west-1	Eropa (Irlandia)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-west-2	Eropa (London)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
eu-west-3	Eropa (Paris)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)
sa-east-1	Amerika Selatan (Sao Paulo)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 Amazon ECS panas aarch64 (aarch64)
us-gov-east-1	AWS GovCloud (AS-Timur)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Wilayah	Nama wilayah	Amazon ECS memanaskan file deb	Amazon ECS memanaskan file rpm
us-gov-west-1	AWS GovCloud (AS-Barat)	Amazon ECS memanas amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS panas x86_64 (x86_64) Amazon ECS panas aarch64 (aarch64)

Untuk menginstal agen penampung Amazon ECS pada EC2 instans Amazon menggunakan Amazon Linux non-AMI

1. Luncurkan EC2 instans Amazon dengan peran IAM yang memungkinkan akses ke Amazon ECS. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.
2. Hubungkan ke instans Anda.
3. Instal versi terbaru Docker di instans Anda.
4. Periksa versi Docker Anda untuk memastikan bahwa sistem Anda memenuhi persyaratan versi minimum.

   > **Note**
   >
   > Versi Docker minimum untuk metrik yang andal adalah versi Docker `v20.10.13` dan yang lebih baru, yang disertakan dalam AMI yang dioptimalkan Amazon ECS dan yang lebih baru. `20220607`
   > Versi agen Amazon ECS `1.20.0` dan yang lebih baru telah menghentikan dukungan untuk versi Docker yang lebih lama dari. `18.01.0`

   ```
   docker --version
   ```

5. Unduh file agen Amazon ECS yang sesuai untuk sistem operasi dan arsitektur sistem Anda dan instal.

   Untuk deb arsitektur:

```
ubuntu:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.amd64.deb
ubuntu:~$ sudo dpkg -i amazon-ecs-init-latest.amd64.deb
```

Untuk `rpm` arsitektur:

```
fedora:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.x86_64.rpm
fedora:~$ sudo yum localinstall -y amazon-ecs-init-latest.x86_64.rpm
```

6. Edit `/lib/systemd/system/ecs.service` file dan tambahkan baris berikut di akhir `[Unit]` bagian.

```
After=cloud-final.service
```

7. (Opsional) Untuk mendaftarkan instance dengan cluster selain `default` cluster, edit `/etc/ecs/ecs.config` file dan tambahkan konten berikut. Contoh berikut menentukan `MyCluster` cluster.

```
ECS_CLUSTER=MyCluster
```

Untuk informasi selengkapnya tentang opsi waktu aktif agen ini, lihat Konfigurasi agen kontainer Amazon ECS.

> **Note**
>
> Anda dapat menyimpan variabel lingkungan agen secara opsional di Amazon S3 (yang dapat diunduh ke instans penampung Anda pada waktu peluncuran menggunakan data pengguna EC2 Amazon). Hal ini dianjurkan untuk informasi sensitif seperti kredensial autentikasi untuk repositori privat. Untuk informasi selengkapnya, lihat Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3 dan Menggunakan gambar AWS non-kontainer di Amazon ECS.

8. Mulai layanan `ecs`.

```
ubuntu:~$ sudo systemctl start ecs
```

Menjalankan agen Amazon ECS dengan mode jaringan host

Saat menjalankan agen penampung Amazon ECS, `ecs-init` akan membuat wadah agen kontainer dengan mode `host` jaringan. Ini adalah satu-satunya mode jaringan untuk kontainer agen kontainer.

Hal ini memungkinkan Anda memblokir akses ke titik akhir layanan metadata EC2 instans Amazon (`http://169.254.169.254`) untuk container yang dimulai oleh agen penampung. Hal ini memastikan bahwa kontainer tidak dapat mengakses kredensil peran IAM dari profil instance container dan memberlakukan bahwa tugas hanya menggunakan kredenal peran tugas IAM. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Hal ini juga mengakibatkan agen kontainer tidak berebut lalu lintas koneksi dan jaringan di jembatan `docker0`.

Parameter konfigurasi log agen kontainer Amazon ECS

Agen kontainer Amazon ECS menyimpan log pada instans kontainer Anda.

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, secara default log terletak di `/var/log/ecs/ecs-agent.log` pada instans Linux dan di `C:\ProgramData\Amazon\ECS\log\ecs-agent.log` pada instans Windows.

Untuk agen kontainer versi 1.35.0 dan sebelumnya, secara default log terletak di `/var/log/ecs/ecs-agent.log.`*`timestamp`* pada instans Linux dan di `C:\ProgramData\Amazon\ECS\log\ecs-agent.log.`*`timestamp`* pada instans Windows.

Secara default, log agen diputar per jam dengan maksimum 24 log yang disimpan.

Berikut ini adalah variabel konfigurasi agen kontainer yang dapat digunakan untuk mengubah perilaku pencatatan agen default. Untuk informasi terperinci tentang semua parameter konfigurasi yang tersedia, lihat Konfigurasi agen kontainer Amazon ECS atau README Agen Amazon ECS aktif. GitHub

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, berikut ini adalah contoh berkas log ketika format `logfmt` digunakan.

```
level=info time=2019-12-12T23:43:29Z msg="Loading configuration" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-agent:latest" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-pause:0.1.0" module=parse.go
```

```
level=info time=2019-12-12T23:43:29Z msg="Amazon ECS agent Version: 1.36.0, Commit:
 ca640387" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Creating root ecs cgroup: /ecs"
 module=init_linux.go
level=info time=2019-12-12T23:43:29Z msg="Creating cgroup /ecs"
 module=cgroup_controller_linux.go
level=info time=2019-12-12T23:43:29Z msg="Loading state!" module=statemanager.go
level=info time=2019-12-12T23:43:29Z msg="Event stream ContainerChange start
 listening..." module=eventstream.go
level=info time=2019-12-12T23:43:29Z msg="Restored cluster 'auto-robc'" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Restored from checkpoint file. I
 am running as 'arn:aws:ecs:us-west-2:0123456789:container-instance/auto-
robc/3330a8a91d15464ea30662d5840164cd' in cluster 'auto-robc'" module=agent.go
```

Berikut ini contoh berkas log ketika format JSON digunakan.

```
{"time": "2019-11-07T22:52:02Z", "level": "info", "msg": "Starting Amazon Elastic
 Container Service Agent", "module": "engine.go"}
```

Untuk agen kontainer versi 1.35.0 dan sebelumnya, berikut adalah format berkas log.

```
2016-08-15T15:54:41Z [INFO] Starting Agent: Amazon ECS Agent - v1.12.0 (895f3c1)
2016-08-15T15:54:41Z [INFO] Loading configuration
2016-08-15T15:54:41Z [WARN] Invalid value for task cleanup duration, will be overridden
 to 3h0m0s, parsed value 0, minimum threshold 1m0s
2016-08-15T15:54:41Z [INFO] Checkpointing is enabled. Attempting to load state
2016-08-15T15:54:41Z [INFO] Loading state! module="statemanager"
2016-08-15T15:54:41Z [INFO] Detected Docker versions [1.17 1.18 1.19 1.20 1.21 1.22]
2016-08-15T15:54:41Z [INFO] Registering Instance with ECS
2016-08-15T15:54:41Z [INFO] Registered! module="api client"
```

Mengonfigurasi instans penampung Amazon ECS untuk gambar Docker pribadi

Agen kontainer Amazon ECS dapat mengautentikasi dengan pendaftar pribadi, menggunakan otentikasi dasar. Bila Anda mengaktifkan autentikasi registri privat, Anda dapat menggunakan citra Docker privat dalam ketentuan tugas Anda. Fitur ini hanya didukung oleh tugas yang menggunakan tipe peluncuran EC2 .

Metode lain untuk mengaktifkan otentikasi registri pribadi digunakan AWS Secrets Manager untuk menyimpan kredensil registri pribadi Anda dengan aman dan kemudian mereferensikannya dalam

definisi kontainer Anda. Hal ini mengizinkan tugas Anda untuk menggunakan citra dari repositori privat. Metode ini mendukung tugas menggunakan jenis peluncuran EC2 atau Fargate. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

Agen kontainer Amazon ECS mencari dua variabel lingkungan saat diluncurkan:

- `ECS_ENGINE_AUTH_TYPE`, yang menentukan tipe data autentikasi yang sedang dikirim.
- `ECS_ENGINE_AUTH_DATA`, yang berisi kredensial autentikasi aktual.

Varian Linux dari AMI Amazon ECS yang dioptimalkan memindai `/etc/ecs/ecs.config` file untuk variabel-variabel ini ketika instance container diluncurkan, dan setiap kali layanan dimulai (dengan perintah). sudo start ecs AMIs yang tidak dioptimalkan Amazon ECS harus menyimpan variabel lingkungan ini dalam file dan meneruskannya dengan `--env-file` *`path_to_env_file`* opsi ke docker run perintah yang memulai agen penampung.

> **Important**
>
> Kami tidak menyarankan Anda menyuntikkan variabel lingkungan otentikasi ini saat peluncuran instance dengan data EC2 pengguna Amazon atau meneruskannya dengan `--env` opsi ke perintah. docker run Metode ini tidak sesuai untuk data sensitif, seperti kredensial autentikasi. Untuk informasi tentang menambahkan kredensial autentikasi secara aman ke instans kontainer Anda, lihat Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3.

Format autentikasi

Ada dua format yang tersedia untuk autentikasi registri privat, `dockercfg` dan `docker`.

Format autentikasi dockercfg

Format `dockercfg` menggunakan informasi autentikasi yang disimpan dalam file konfigurasi yang dibuat ketika Anda menjalankan perintah docker login. Anda dapat membuat file ini dengan menjalankan docker login di sistem lokal dan memasukkan nama pengguna registri, kata sandi, dan alamat email. Anda juga dapat masuk ke instans kontainer dan menjalankan perintah di sana. Tergantung pada versi Docker Anda, file ini disimpan sebagai `~/.dockercfg` atau `~/.docker/config.json`.

```
cat ~/.docker/config.json
```

Output:

```
{
  "auths": {
    "https://index.docker.io/v1/": {
      "auth": "zq212MzEXAMPLE7o6T25Dk0i"
    }
  }
}
```

> **Important**
>
> Versi Docker yang lebih baru membuat file konfigurasi seperti yang ditunjukkan di atas dengan objek `auths`. Agen Amazon ECS hanya mendukung data `dockercfg` otentikasi yang dalam format di bawah ini, tanpa objek. `auths` Jika Anda memiliki utilitas jq yang terinstal, Anda dapat mengekstrak data ini dengan perintah berikut: cat ~/.docker/config.json | jq .auths

```
cat ~/.docker/config.json | jq .auths
```

Output:

```
{
  "https://index.docker.io/v1/": {
    "auth": "zq212MzEXAMPLE7o6T25Dk0i",
    "email": "email@example.com"
  }
}
```

Dalam contoh di atas, variabel lingkungan berikut harus ditambahkan ke file variabel lingkungan (`/etc/ecs/ecs.config`untuk AMI yang dioptimalkan Amazon ECS) yang dimuat agen kontainer Amazon ECS saat runtime. Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan dan Anda memulai agen secara manualdocker run, tentukan file variabel lingkungan dengan `--env-file` *`path_to_env_file`* opsi saat Anda memulai agen.

```
ECS_ENGINE_AUTH_TYPE=dockercfg
```

```
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

Anda dapat mengonfigurasi beberapa registri privat dengan sintaksis berikut:

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example-01.com"},"repo.example-02.com":
{"auth":"fQ172MzEXAMPLEoF7225DU0j","email":"email@example-02.com"}}
```

Format autentikasi dockercfg

Format `docker` menggunakan representasi server registri JSON yang harus diautentikasi dengan agen. Hal ini juga mencakup parameter autentikasi yang diperlukan oleh registri tersebut (seperti nama pengguna, kata sandi, dan alamat email untuk akun tersebut). Untuk akun Docker Hub, representasi JSON terlihat seperti berikut ini:

```
{
  "https://index.docker.io/v1/": {
    "username": "my_name",
    "password": "my_password",
    "email": "email@example.com"
  }
}
```

Dalam contoh ini, variabel lingkungan berikut harus ditambahkan ke file variabel lingkungan (`/etc/ecs/ecs.config`untuk AMI yang dioptimalkan Amazon ECS) yang dimuat agen kontainer Amazon ECS saat runtime. Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan, dan Anda memulai agen secara manualdocker run, tentukan file variabel lingkungan dengan `--env-file` *`path_to_env_file`* opsi saat Anda memulai agen.

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

Anda dapat mengonfigurasi beberapa registri privat dengan sintaksis berikut:

```
ECS_ENGINE_AUTH_TYPE=docker
```

```
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"username":"my_name","password":"my_password","email":"email@example-01.com"},"repo.example-02
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

Prosedur

Gunakan prosedur berikut untuk mengaktifkan pendaftar pribadi untuk instance kontainer Anda.

Untuk mengaktifkan pendaftar pribadi di AMI Amazon ECS yang dioptimalkan

1. Masuk ke instans kontainer Anda menggunakan SSH.
2. Buka file `/etc/ecs/ecs.config` dan tambahkan nilai `ECS_ENGINE_AUTH_TYPE` dan `ECS_ENGINE_AUTH_DATA` untuk registri dan akun Anda:

   ```
   sudo vi /etc/ecs/ecs.config
   ```

   Contoh ini mengautentikasi akun pengguna Docker Hub:

   ```
   ECS_ENGINE_AUTH_TYPE=docker
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"username":"my_name","password":"my_password","email":"email@example.com"}}
   ```

3. Lakukan pemeriksaan untuk melihat apakah agen Anda menggunakan variabel lingkungan `ECS_DATADIR` untuk menyimpan statusnya:

   ```
   docker inspect ecs-agent | grep ECS_DATADIR
   ```

   Output:

   ```
   "ECS_DATADIR=/data",
   ```

   > **Important**
   >
   > Jika perintah sebelumnya tidak mengembalikan variabel lingkungan `ECS_DATADIR`, Anda harus menghentikan setiap tugas yang berjalan pada instans kontainer ini sebelum menghentikan agen. Agen yang lebih baru dengan variabel lingkungan `ECS_DATADIR` menyimpan statusnya dan Anda dapat menghentikannya saat tugas sedang berjalan

tanpa masalah. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

4. Hentikan layanan `ecs`:

```
sudo stop ecs
```

Output:

```
ecs stop/waiting
```

5. Mulai ulang layanan `ecs`.

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

     ```
     sudo systemctl restart ecs
     ```

   - Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

     ```
     sudo stop ecs && sudo start ecs
     ```

6. (Opsional) Anda dapat memastikan bahwa agen sedang berjalan dan melihat beberapa informasi tentang instans kontainer baru dengan melakukan kueri pada operasi API introspeksi agen. Untuk informasi selengkapnya, lihat the section called “Introspeksi wadah”.

```
curl http://localhost:51678/v1/metadata
```

Tugas Amazon ECS otomatis dan pembersihan gambar

Setiap kali tugas ditempatkan pada instance kontainer, agen penampung Amazon ECS memeriksa untuk melihat apakah gambar yang direferensikan dalam tugas adalah yang terbaru dari tag yang ditentukan dalam repositori. Jika tidak, perilaku default mengizinkan agen untuk menarik citra dari repositori masing-masing. Jika Anda sering memperbarui citra dalam tugas dan layanan Anda, penyimpanan instans kontainer dapat dengan cepat terisi citra Docker yang tidak lagi Anda gunakan dan mungkin tidak akan pernah digunakan kembali. Misalnya, Anda dapat menggunakan integrasi dan alur (CI/CD) deployment berkelanjutan.

> **Note**
>
> Perilaku tarik gambar agen Amazon ECS dapat disesuaikan menggunakan `ECS_IMAGE_PULL_BEHAVIOR` parameter. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Demikian juga, kontainer dengan tugas berhenti juga dapat memakan penyimpanan instans kontainer dengan informasi log, volume data, dan artefak lainnya. Artefak ini berguna untuk debugging kontainer yang telah berhenti tiba-tiba, tetapi sebagian besar penyimpanan ini dapat dengan aman dibersihkan setelah jangka waktu tertentu.

Secara default, agen penampung Amazon ECS secara otomatis membersihkan tugas yang dihentikan dan gambar Docker yang tidak digunakan oleh tugas apa pun pada instance penampung Anda.

> **Note**
>
> Fitur pembersihan gambar otomatis memerlukan setidaknya versi 1.13.0 dari agen penampung Amazon ECS. Untuk memperbarui agen ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.

Variabel konfigurasi agen berikut tersedia untuk menyetel tugas otomatis dan pengalaman pembersihan citra Anda. Untuk informasi lebih lanjut tentang cara mengatur variabel ini pada instans kontainer Anda, lihat Konfigurasi agen kontainer Amazon ECS.

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

Waktu default untuk menunggu untuk menghapus kontainer untuk tugas yang dihentikan. Jika nilai diatur ke kurang dari 1 detik, nilai diabaikan. Secara default, parameter ini diatur ke 3 jam, tetapi Anda dapat mengurangi periode ini serendah 1 detik jika Anda perlu untuk aplikasi Anda.

> **Note**
>
> Proses pembersihan citra tidak dapat dilakukan selama ada kontainer yang mereferensikannya. Setelah wadah dihapus, gambar yang tidak direferensikan menjadi kandidat untuk pembersihan berdasarkan parameter konfigurasi pembersihan gambar.

`ECS_DISABLE_IMAGE_CLEANUP`

Jika Anda menyetel variabel ini`true`, maka pembersihan gambar otomatis dimatikan pada instance penampung Anda dan tidak ada gambar yang dihapus secara otomatis.

`ECS_IMAGE_CLEANUP_INTERVAL`

Variabel ini menentukan seberapa sering proses pembersihan citra otomatis harus diperiksa untuk menghapus citra. Pengaturan defaultnya adalah setiap 30 menit tetapi Anda dapat mengurangi periode ini hingga 10 menit untuk lebih sering menghapus citra.

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

Variabel ini menentukan jumlah minimum waktu antara ketika citra ditarik dan kapan citra tersebut akan dihapus. Hal ini digunakan agar citra yang baru saja ditarik tidak dihapus. Pengaturan default-nya adalah 1 jam.

`ECS_NUM_IMAGES_DELETE_PER_CYCLE`

Variabel ini menentukan berapa banyak gambar yang dapat dihapus selama siklus pembersihan tunggal. Pengaturan default-nya adalah 5 dan nilai minimumnya adalah 1.

Saat agen penampung Amazon ECS berjalan dan pembersihan gambar otomatis tidak dimatikan, agen memeriksa gambar Docker yang tidak direferensikan dengan menjalankan atau menghentikan kontainer pada frekuensi yang ditentukan oleh variabel. `ECS_IMAGE_CLEANUP_INTERVAL` Jika citra yang tidak digunakan ditemukan dan citra tersebut lebih tua dari waktu pembersihan minimum yang ditentukan oleh variabel `ECS_IMAGE_MINIMUM_CLEANUP_AGE`, agen tersebut menghapus citra hingga jumlah maksimum yang ditentukan oleh variabel `ECS_NUM_IMAGES_DELETE_PER_CYCLE`. Citra yang paling sedikit direferensikan baru-baru ini akan dihapus terlebih dahulu. Setelah citra dihapus, agen menunggu sampai interval berikutnya dan mengulangi proses lagi.

Jadwalkan wadah Anda di Amazon ECS

Amazon Elastic Container Service (Amazon ECS) adalah sistem konkurensi optimis status bersama yang menyediakan kemampuan penjadwalan fleksibel untuk beban kerja kontainer Anda. Penjadwal Amazon ECS menggunakan informasi status cluster yang sama dengan Amazon ECS API untuk membuat keputusan penempatan yang tepat.

Amazon ECS menyediakan penjadwal layanan untuk tugas dan aplikasi yang berjalan lama. Ini juga menyediakan kemampuan untuk menjalankan tugas mandiri atau tugas terjadwal untuk pekerjaan batch atau tugas tunggal. Anda dapat menentukan strategi dan kendala penempatan tugas untuk menjalankan tugas yang paling sesuai dengan kebutuhan Anda. Misalnya, Anda dapat menentukan apakah tugas berjalan di beberapa Availability Zone atau dalam Availability Zone tunggal. Dan, secara opsional, Anda dapat mengintegrasikan tugas dengan penjadwal kustom atau pihak ketiga Anda sendiri.

Opsi	Kapan harus digunakan	Informasi selengkapnya
Layanan	Penjadwal layanan cocok untuk layanan dan aplikasi stateless yang berjalan lama. Penjadwal layanan secara opsional juga memastikan bahwa tugas terdaftar terhadap penyeimbang beban Elastic Load Balancing. Anda dapat memperbarui layanan Anda yang dikelola oleh penjadwal layanan. Ini mungkin termasuk men-deploy ketentuan tugas baru atau mengubah jumlah tugas yang diinginkan yang sedang berjalan. Secara default, penjadwal layanan menyebar tugas di beberapa Availability Zone. Namun,	Layanan-layanan Amazon ECS

Opsi	Kapan harus digunakan	Informasi selengkapnya
	Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuai kan keputusan penempatan tugas.	
Tugas mandiri	Tugas mandiri cocok untuk proses seperti pekerjaan batch yang melakukan pekerjaan dan kemudian berhenti. Misalnya, Anda dapat memiliki panggilan proses `RunTask` ketika pekerjaan masuk ke antrian. Tugas menarik pekerjaan dari antrean, melaksanakan pekerjaan , dan kemudian keluar. Menggunakan `RunTask`, Anda dapat mengizinkan strategi penempatan tugas default untuk mendistribusikan tugas secara acak di klaster Anda. Hal ini meminimalkan kemungkinan instans tunggal mendapatkan jumlah tugas yang tidak proporsional.	Tugas mandiri Amazon ECS

Opsi	Kapan harus digunakan	Informasi selengkapnya
Tugas terjadwal	Tugas terjadwal cocok ketika Anda memiliki tugas untuk dijalankan pada interval yang ditetapkan di cluster Anda, Anda dapat menggunak an EventBridge Scheduler untuk membuat jadwal. Anda dapat menjalankan tugas untuk operasi pencadangan atau pemindaian log. Jadwal EventBridge Scheduler yang Anda buat dapat menjalankan satu atau beberapa tugas di klaster Anda pada waktu yang ditentukan. Acara terjadwal Anda dapat diatur ke interval tertentu (dijalankan setiap *N* menit, jam, atau hari). Jika tidak, untuk penjadwalan yang lebih rumit, Anda dapat menggunakan `cron` ekspresi.	Menggunakan Amazon EventBridge Scheduler untuk menjadwalkan tugas Amazon ECS

Opsi komputasi

Dengan Amazon ECS, Anda dapat menentukan infrastruktur tugas atau layanan yang dijalankan. Anda dapat menggunakan strategi penyedia kapasitas, atau jenis peluncuran.

Untuk Fargate, penyedia kapasitas adalah Fargate dan Fargate Spot. Untuk EC2, penyedia kapasitas adalah grup Auto Scaling dengan instance kontainer terdaftar.

Strategi penyedia kapasitas mendistribusikan tugas Anda ke seluruh penyedia kapasitas yang terkait dengan klaster Anda.

Hanya penyedia kapasitas yang keduanya sudah terkait dengan cluster dan memiliki `UPDATING` status `ACTIVE` atau yang dapat digunakan dalam strategi penyedia kapasitas. Anda dapat mengaitkan penyedia kapasitas dengan klaster saat membuat klaster.

Dalam strategi penyedia kapasitas, nilai dasar opsional menunjukkan berapa banyak tugas, minimal, yang dijalankan pada penyedia kapasitas tertentu. Hanya satu penyedia kapasitas di strategi penyedia kapasitas yang dapat menentukan nilai dasar.

Nilai bobot menentukan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan. Pertimbangkan contoh berikut. Anda memiliki strategi yang berisi dua penyedia kapasitas, dan keduanya memiliki bobot1. Ketika persentase dasar tercapai, tugas dibagi secara merata di dua penyedia kapasitas. Menggunakan logika yang sama, misalkan Anda menentukan bobot untuk CapacityProvidera dan bobot 1 untuk CapacityProviderB. *4* Kemudian, untuk setiap tugas yang dijalankan menggunakan CapacityProvidera, ada empat tugas yang menggunakan CapacityProviderB.

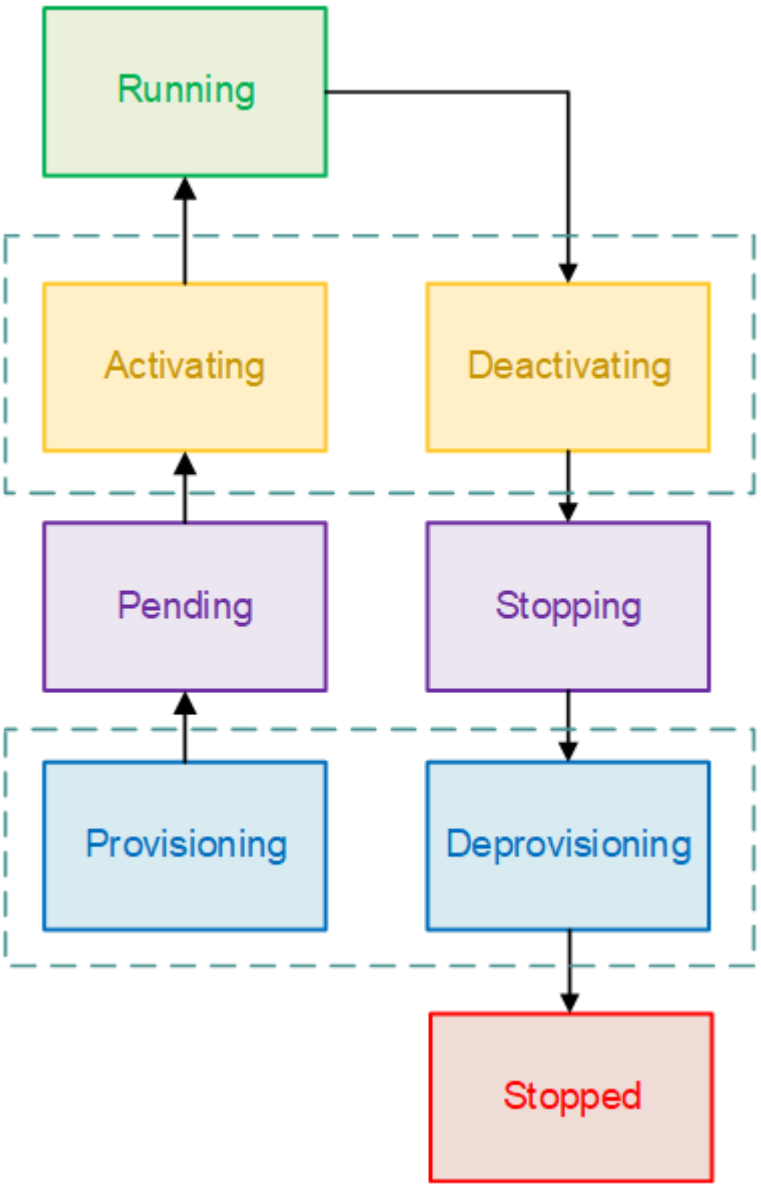
Jenis peluncuran meluncurkan tugas Anda secara langsung di Fargate atau di instans EC2 Amazon yang telah Anda daftarkan secara manual ke cluster Anda.

Siklus hidup tugas Amazon ECS

Ketika tugas dimulai, baik secara manual atau sebagai bagian dari layanan, itu dapat melalui beberapa status sebelum selesai sendiri atau dihentikan secara manual. Beberapa tugas dimaksudkan untuk dijalankan sebagai tugas batch yang secara alami terus berjalan dari `PENDING` ke `RUNNING` ke `STOPPED`. Tugas-tugas lain, yang dapat menjadi bagian dari layanan, dimaksudkan untuk terus berjalan tanpa batas waktu, atau untuk dinaik-turunkan skalanya sesuai kebutuhan.

Ketika perubahan status tugas diminta, seperti menghentikan tugas atau memperbarui jumlah layanan yang diinginkan untuk menskalakannya naik atau turun, agen penampung Amazon ECS melacak perubahan ini sebagai status (`lastStatus`) tugas terakhir yang diketahui dan status (`desiredStatus`) tugas yang diinginkan. Kedua status yang diketahui terakhir dan status yang diinginkan dari tugas dapat dilihat baik di konsol atau dengan menggambarkan tugas dengan API atau AWS CLI.

Diagram alur di bawah ini menunjukkan alur siklus hidup tugas.



Status siklus hidup

Berikut ini adalah deskripsi dari masing-masing status siklus hidup tugas.

PENYEDIAAN

Amazon ECS harus melakukan langkah-langkah tambahan sebelum tugas diluncurkan. Misalnya, untuk tugas yang menggunakan mode jaringan `awsvpc`, antarmuka jaringan elastis perlu ditetapkan.

MENUNGGU

Ini adalah keadaan transisi di mana Amazon ECS sedang menunggu agen kontainer untuk mengambil tindakan lebih lanjut. Tugas tetap dalam status tertunda sampai ada sumber daya yang tersedia untuk tugas tersebut.

MENGAKTIFKAN

Ini adalah keadaan transisi di mana Amazon ECS harus melakukan langkah-langkah tambahan setelah tugas diluncurkan tetapi sebelum tugas dapat bertransisi ke `RUNNING` status. Ini adalah keadaan di mana Amazon ECS menarik gambar kontainer, membuat kontainer, mengonfigurasi jaringan tugas, mendaftarkan grup target penyeimbang beban, dan mengonfigurasi penemuan layanan.

BERJALAN

Tugas ini berhasil berjalan.

MENONAKTIFKAN

Ini adalah keadaan transisi di mana Amazon ECS harus melakukan langkah-langkah tambahan sebelum tugas dihentikan. Misalnya, untuk tugas yang merupakan bagian dari layanan yang dikonfigurasi untuk menggunakan grup target Elastic Load Balancings, deregistrasi grup target terjadi selama status ini.

BERHENTI

Ini adalah keadaan transisi di mana Amazon ECS sedang menunggu agen kontainer untuk mengambil tindakan lebih lanjut.

Untuk wadah Linux, agen kontainer akan mengirim `SIGTERM` sinyal untuk memberi tahu aplikasi harus selesai dan dimatikan, dan kemudian mengirim `SIGKILL` setelah menunggu `StopTimeout` durasi yang ditetapkan dalam definisi tugas.

PEMBATALAN PENYEDIAAN

Amazon ECS harus melakukan langkah-langkah tambahan setelah tugas berhenti tetapi sebelum tugas beralih ke negara bagian`STOPPED`. Misalnya, untuk tugas yang menggunakan mode jaringan`awsvpc`, antarmuka jaringan elastis perlu dilepas dan dihapus.

DIHENTIKAN

Tugas telah berhasil dihentikan.

Jika tugas Anda berhenti karena kesalahan, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

DELETED

Ini adalah keadaan transisi ketika tugas berhenti. Status ini tidak ditampilkan di konsol, tetapi ditampilkan di`describe-tasks`.

Bagaimana Amazon ECS menempatkan tugas pada instans kontainer

Anda dapat menggunakan penempatan tugas untuk mengonfigurasi Amazon ECS untuk menempatkan tugas Anda pada instance container yang memenuhi kriteria tertentu, misalnya Availability Zone atau tipe instans.

Berikut ini adalah komponen penempatan tugas:

- Strategi penempatan tugas - Algoritma untuk memilih instance kontainer untuk penempatan tugas atau tugas untuk penghentian. Misalnya, Amazon ECS dapat memilih instans kontainer secara acak, atau dapat memilih instance kontainer sedemikian rupa sehingga tugas didistribusikan secara merata di seluruh grup instance.
- Kelompok tugas - Sekelompok tugas terkait, misalnya tugas database.
- Kendala penempatan tugas - Ini adalah aturan yang harus dipenuhi untuk menempatkan tugas pada instance kontainer. Jika kendala tidak terpenuhi, tugas tidak ditempatkan dan tetap di negara bagian. `PENDING` Misalnya, Anda dapat menggunakan kendala untuk menempatkan tugas hanya pada jenis instance tertentu.

Amazon ECS memiliki algoritma yang berbeda untuk jenis peluncuran.

EC2 jenis peluncuran

Untuk tugas yang menggunakan tipe EC2 peluncuran, Amazon ECS harus menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang ditentukan dalam definisi tugas, seperti CPU dan memori. Demikian pula, saat Anda menurunkan jumlah tugas, Amazon ECS harus menentukan tugas mana yang akan dihentikan. Anda dapat menerapkan strategi dan batasan penempatan tugas untuk menyesuaikan cara Amazon ECS menempatkan dan mengakhiri tugas.

Strategi penempatan tugas default bergantung pada apakah Anda menjalankan tugas secara manual (tugas mandiri) atau dalam layanan. Untuk tugas yang berjalan sebagai bagian dari layanan Amazon ECS, strategi penempatan tugas `spread` menggunakan. `attribute:ecs.availability-zone` Tidak ada batasan penempatan tugas default untuk tugas yang tidak ada di layanan. Untuk informasi selengkapnya, lihat Jadwalkan wadah Anda di Amazon ECS.

> **Note**
>
> Strategi penempatan tugas adalah upaya terbaik. Amazon ECS masih mencoba untuk menempatkan tugas bahkan ketika opsi penempatan yang paling optimal tidak tersedia. Namun, kendala penempatan tugas bersifat mengikat, dan mereka dapat mencegah penempatan tugas.

Anda dapat menggunakan strategi dan kendala penempatan tugas bersama-sama. Misalnya, Anda dapat menggunakan strategi penempatan tugas dan kendala penempatan tugas untuk mendistribusikan tugas di tugas Availability Zone dan paket bin berdasarkan memori dalam setiap Availability Zone, tetapi hanya untuk instans G2.

Saat Amazon ECS menempatkan tugas, Amazon menggunakan proses berikut untuk memilih instance container:

1. Identifikasi instance kontainer yang memenuhi persyaratan CPU, GPU, memori, dan port dalam definisi tugas.
2. Identifikasi instance kontainer yang memenuhi kendala penempatan tugas.
3. Identifikasi instance kontainer yang memenuhi strategi penempatan tugas.
4. Pilih instance kontainer untuk penempatan tugas.

Jenis peluncuran Fargate

Strategi penempatan tugas dan kendala tidak didukung untuk tugas yang menggunakan tipe peluncuran Fargate. Fargate akan mencoba yang terbaik untuk menyebarkan tugas di seluruh Availability Zone yang dapat diakses. Jika penyedia kapasitas mencakup Fargate dan Fargate Spot, perilaku spread bersifat independen untuk setiap penyedia kapasitas.

Gunakan strategi untuk menentukan penempatan tugas Amazon ECS

Untuk tugas yang menggunakan tipe EC2 peluncuran, Amazon ECS harus menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang ditentukan dalam definisi tugas, seperti CPU dan memori. Demikian pula, saat Anda menurunkan jumlah tugas, Amazon ECS harus menentukan tugas mana yang akan dihentikan. Anda dapat menerapkan strategi dan batasan penempatan tugas untuk menyesuaikan cara Amazon ECS menempatkan dan mengakhiri tugas.

Strategi penempatan tugas default bergantung pada apakah Anda menjalankan tugas secara manual (tugas mandiri) atau dalam layanan. Untuk tugas yang berjalan sebagai bagian dari layanan Amazon ECS, strategi penempatan tugas `spread` menggunakan. `attribute:ecs.availability-zone` Tidak ada batasan penempatan tugas default untuk tugas yang tidak ada di layanan. Untuk informasi selengkapnya, lihat Jadwalkan wadah Anda di Amazon ECS.

> **Note**
>
> Strategi penempatan tugas adalah upaya terbaik. Amazon ECS masih mencoba untuk menempatkan tugas bahkan ketika opsi penempatan yang paling optimal tidak tersedia. Namun, kendala penempatan tugas bersifat mengikat, dan mereka dapat mencegah penempatan tugas.

Anda dapat menggunakan strategi dan kendala penempatan tugas bersama-sama. Misalnya, Anda dapat menggunakan strategi penempatan tugas dan kendala penempatan tugas untuk mendistribusikan tugas di tugas Availability Zone dan paket bin berdasarkan memori dalam setiap Availability Zone, tetapi hanya untuk instans G2.

Saat Amazon ECS menempatkan tugas, Amazon menggunakan proses berikut untuk memilih instance container:

1. Identifikasi instance kontainer yang memenuhi persyaratan CPU, GPU, memori, dan port dalam definisi tugas.
2. Identifikasi instance kontainer yang memenuhi kendala penempatan tugas.
3. Identifikasi instance kontainer yang memenuhi strategi penempatan tugas.
4. Pilih instance kontainer untuk penempatan tugas.

Anda menentukan strategi penempatan tugas dalam definisi layanan, atau definisi tugas menggunakan `placementStrategy` parameter.

```
"placementStrategy": [
    {
        "field": "The field to apply the placement strategy against",
        "type": "The placement strategy to use"
    }
]
```

Anda dapat menentukan strategi saat menjalankan task (RunTask), membuat layanan baru (CreateService), atau memperbarui layanan yang ada (UpdateService).

Tabel berikut menjelaskan jenis dan bidang yang tersedia.

jenis	Nilai bidang yang valid	
`binpack` Tugas ditempatkan pada instans kontainer sehingga meninggalkan jumlah paling sedikit untuk CPU atau memori yang tidak terpakai. Strategi ini meminimalkan jumlah instans kontainer yang digunakan. Ketika strategi ini digunakan dan tindakan scale-in diambil, Amazon ECS menghenti kan tugas. Hal ini dilakukan berdasarkan jumlah sumber daya yang tersisa pada instance container setelah tugas dihentikan. Instance kontainer yang memiliki sumber daya paling banyak yang tersisa setelah penghenti an tugas telah menghentikan tugas tersebut.	• cpu • memori	

jenis	Nilai bidang yang valid
`random` Tugas ditempatkan secara acak.	Tidak digunakan
`spread` Tugas ditempatkan secara merata berdasarkan nilai yang ditentukan. Tugas layanan tersebar berdasarkan tugas dari layanan tersebut. Tugas mandiri tersebar berdasarkan tugas dari grup tugas yang sama. Untuk informasi selengkapnya tentang grup tugas, lihat Tugas Amazon ECS terkait grup . Saat `spread` strategi digunakan dan tindakan penskalaan diambil, Amazon ECS memilih tugas untuk dihentikan yang menjaga keseimbangan di seluruh Availability Zone. Dalam Availability Zone, tugas dipilih secara acak.	• `instanceId` (atau`host`, yang memiliki efek yang sama) • platform atau atribut khusus apa pun yang diterapkan ke instance container, seperti `attribute:ecs.availability-zone`

Strategi penempatan tugas dapat diperbarui untuk layanan yang ada juga. Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Anda dapat membuat strategi penempatan tugas yang menggunakan beberapa strategi dengan membuat array strategi sesuai urutan yang Anda inginkan. Misalnya, jika Anda ingin menyebarkan

tugas di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone, tentukan strategi Availability Zone yang diikuti oleh strategi memori. Misalnya strategi, lihatContoh strategi penempatan tugas Amazon ECS.

Contoh strategi penempatan tugas Amazon ECS

Anda dapat menentukan strategi penempatan tugas dengan tindakan berikut: CreateService, UpdateService, dan RunTask.

Contoh

- Mendistribusikan tugas secara merata di seluruh Availability Zone
- Mendistribusikan tugas secara merata di semua contoh
- Tugas bin pack berdasarkan memori
- Tempatkan tugas secara acak
- Mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instance dalam setiap Availability Zone
- Mendistribusikan tugas secara merata di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone
- Mendistribusikan tugas secara merata di seluruh instance dan kemudian tugas bin pack berdasarkan memori

Mendistribusikan tugas secara merata di seluruh Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    }
]
```

Mendistribusikan tugas secara merata di semua contoh

Strategi berikut mendistribusikan tugas secara merata di semua instans.

```
"placementStrategy": [
    {
```

```
        "field": "instanceId",
        "type": "spread"
    }
]
```

Tugas bin pack berdasarkan memori

Berikut tugas bin pack strategi berdasarkan memori.

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Tempatkan tugas secara acak

Strategi berikut menempatkan tugas secara acak.

```
"placementStrategy": [
    {
        "type": "random"
    }
]
```

Mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instance dalam setiap Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian mendistribusikan tugas secara merata di seluruh instans dalam setiap Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

Mendistribusikan tugas secara merata di seluruh Availability Zones dan kemudian bin pack task berdasarkan memori dalam setiap Availability Zone

Strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zone dan kemudian tugas bin pack berdasarkan memori dalam setiap Availability Zone.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Mendistribusikan tugas secara merata di seluruh instance dan kemudian tugas bin pack berdasarkan memori

Strategi berikut mendistribusikan tugas secara merata di semua instance dan kemudian mengemas tugas berdasarkan memori dalam setiap instance.

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Tugas Amazon ECS terkait grup

Anda dapat mengidentifikasi serangkaian tugas terkait dan menempatkannya dalam kelompok tugas. Semua tugas dengan nama grup tugas yang sama dianggap sebagai satu rangkaian ketika menggunakan strategi penempatan tugas `spread`. Misalnya, Anda menjalankan aplikasi yang berbeda dalam satu cluster, seperti database dan server web. Untuk memastikan bahwa

basis data Anda seimbang di seluruh Availability Zone, tambahkan basis data ke grup tugas bernama `databases` dan kemudian gunakan strategi penempatan tugas `spread`. Untuk informasi selengkapnya, lihat Gunakan strategi untuk menentukan penempatan tugas Amazon ECS.

Kelompok tugas juga dapat digunakan sebagai kendala penempatan tugas. Saat Anda menentukan grup tugas dalam `memberOf` batasan, tugas hanya dikirim ke instance kontainer yang menjalankan tugas dalam grup tugas yang ditentukan. Sebagai contoh, lihat Contoh kendala penempatan tugas Amazon ECS.

Secara default, tugas mandiri menggunakan nama keluarga definisi tugas (misalnya,`family:my-task-definition`) sebagai nama grup tugas jika nama grup tugas kustom tidak ditentukan. Tugas yang diluncurkan sebagai bagian dari layanan menggunakan nama layanan sebagai nama grup tugas dan tidak dapat diubah.

Persyaratan berikut untuk kelompok tugas berlaku.

- Nama grup tugas harus 255 karakter atau kurang.
- Setiap tugas bisa berada dalam satu grup.
- Setelah meluncurkan tugas, Anda tidak dapat memodifikasi grup tugasnya.

Tentukan instance kontainer mana yang digunakan Amazon ECS untuk tugas

Batasan penempatan tugas adalah aturan tentang instance container yang digunakan Amazon ECS untuk menentukan apakah tugas diizinkan untuk dijalankan pada instance. Setidaknya satu instance kontainer harus cocok dengan kendala. Jika tidak ada instance yang cocok dengan kendala, tugas tetap dalam keadaan. `PENDING` Saat membuat layanan baru atau memperbarui layanan yang sudah ada, Anda dapat menentukan batasan penempatan tugas untuk tugas layanan.

Anda dapat menentukan batasan penempatan tugas dalam definisi layanan, definisi tugas, atau tugas menggunakan parameter. `placementConstraint`

```
"placementConstraints": [
    {
        "expression": "The expression that defines the task placement constraints",
        "type": "The placement constraint type to use"
    }
]
```

Tabel berikut menjelaskan cara menggunakan parameter.

Jenis kendala	Dapat ditentukan kapan	
`distinctInstance` Tempatkan setiap tugas aktif pada instance kontainer yang berbeda. Amazon ECS melihat status tugas yang diinginkan untuk penempatan tugas. Misalnya, jika status yang diinginkan dari tugas yang ada adalah`STOPPED`, (tetapi status terakhir tidak), tugas masuk baru dapat ditempatkan pada instance yang sama meskipun ada kendala `distinctInstance` penempatan. Oleh karena itu, Anda mungkin melihat 2 tugas dengan status terakhir `RUNNING` pada instance yang sama. **Important** Kami menyarankan agar pelanggan yang mencari isolasi yang kuat untuk tugas mereka menggunakan Fargate. Fargate menjalankan setiap tugas dalam lingkungan virtualisasi perangkat keras.	• Menjalankan tugas RunTask • Membuat layanan baru CreateService,	

Jenis kendala	Dapat ditentukan kapan
Ini memastikan bahwa beban kerja kontainer ini tidak berbagi antarmuka jaringan, penyimpan an sementara Fargate, CPU, atau memori dengan tugas lain. Untuk informasi selengkapnya, lihat Ikhtisar Keamanan AWS Fargate.	
`memberOf` Tempatkan tugas pada instans kontainer yang memenuhi ekspresi.	• Menjalankan tugas RunTask • Membuat layanan baru CreateService, • Membuat definisi tugas baru RegisterTaskDefinition • Membuat revisi baru dari definisi tugas RegisterT askDefinition • Memperbarui layanan UpdateService

Bila menggunakan tipe `memberOf` constraint, Anda dapat membuat ekspresi menggunakan bahasa kueri klaster yang menentukan instance container tempat Amazon ECS dapat menempatkan tugas. Ekspresi adalah cara bagi Anda untuk mengelompokkan instance container Anda berdasarkan atribut. Ekspresi masuk dalam `expression` parameter`placementConstraint`.

Atribut instans wadah Amazon ECS

Anda dapat menambahkan metadata kustom ke instans kontainer Anda, yang dikenal sebagai atribut. Tiap atribut memiliki nama dan nilai string opsional. Anda dapat menggunakan atribut bawaan yang disediakan oleh Amazon ECS atau menentukan atribut khusus.

Bagian berikut berisi contoh built-in, opsional, dan atribut kustom.

Atribut bawaan

Amazon ECS secara otomatis menerapkan atribut berikut ke instance container Anda.

`ecs.ami-id`

ID AMI yang digunakan untuk meluncurkan instans. Nilai contoh untuk atribut ini adalah `ami-1234abcd`.

`ecs.availability-zone`

Availability Zone untuk instans. Nilai contoh untuk atribut ini adalah `us-east-1a`.

`ecs.instance-type`

Jenis instans untuk instans tersebut. Nilai contoh untuk atribut ini adalah `g2.2xlarge`.

`ecs.os-type`

Sistem operasi untuk instans. Nilai yang mungkin untuk atribut ini adalah `linux` dan `windows`.

`ecs.os-family`

Versi sistem operasi untuk contoh.

Untuk contoh Linux, nilai yang valid adalah`LINUX`. Untuk instance Windows, ECS menetapkan nilai dalam format. `WINDOWS_SERVER_<OS_Release>_<FULL or CORE>` Nilai yang valid adalah`WINDOWS_SERVER_2022_FULL`,`WINDOWS_SERVER_2022_CORE`,`WINDOWS_SERVER_20H2_CORE` dan`WINDOWS_SERVER_2016_FULL`.

Ini penting untuk wadah Windows dan Windows containers on AWS Fargate karena versi OS dari setiap wadah Windows harus cocok dengan host. Jika versi Windows dari gambar kontainer berbeda dari host, penampung tidak dimulai. Untuk informasi selengkapnya, lihat Kompatibilitas versi penampung Windows di situs web dokumentasi Microsoft.

Jika klaster Anda menjalankan beberapa versi Windows, Anda dapat memastikan bahwa tugas ditempatkan pada EC2 instance yang berjalan pada versi yang sama dengan menggunakan batasan penempatan:. `memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)` Untuk informasi selengkapnya, lihat the section called “Mengambil metadata Windows AMI yang dioptimalkan Amazon ECS”.

`ecs.cpu-architecture`

Arsitektur CPU untuk instans. Nilai contoh untuk atribut ini adalah `x86_64` dan `arm64`.

`ecs.vpc-id`

VPC tempat instans diluncurkan. Nilai contoh untuk atribut ini adalah `vpc-1234abcd`.

`ecs.subnet-id`

Subnet yang digunakan instans. Nilai contoh untuk atribut ini adalah `subnet-1234abcd`.

Atribut opsional

Amazon ECS dapat menambahkan atribut berikut ke instance container Anda.

`ecs.awsvpc-trunk-id`

Jika atribut ini ada, instans memiliki antarmuka jaringan trunk. Untuk informasi selengkapnya, lihat Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux.

`ecs.outpost-arn`

Jika atribut ini ada, itu berisi Amazon Resource Name (ARN) dari Outpost. Untuk informasi selengkapnya, lihat the section called “Layanan Kontainer Elastis Amazon aktif AWS Outposts”.

`ecs.capability.external`

Jika atribut ini ada, instans diidentifikasi sebagai instans eksternal. Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk jenis peluncuran eksternal.

Atribut kustom

Anda dapat menerapkan atribut kustom ke instans kontainer Anda. Misalnya, Anda dapat menentukan atribut dengan nama “stack” dan nilai “prod”.

Saat menentukan atribut khusus, Anda harus mempertimbangkan hal berikut.

- `name` harus berisi antara 1 hingga 128 karakter dan nama boleh berisi huruf (huruf besar dan huruf kecil), angka, tanda hubung, garis bawah, garis miring ke depan, garis miring belakang, atau titik.
- `value` harus berisi antara 1 hingga 128 karakter dan boleh berisi huruf (huruf besar dan huruf kecil), angka, tanda hubung, garis bawah, titik, tanda (@), garis miring ke depan, garis miring belakang, titik dua, atau spasi. Nilai tidak dapat berisi spasi putih utama atau belakang.

Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS

Kueri klaster adalah ekspresi yang memungkinkan Anda mengelompokkan objek. Misalnya, Anda dapat mengelompokkan instans kontainer dengan atribut seperti Availability Zone, tipe instans, atau metadata kustom. Untuk informasi selengkapnya, lihat Atribut instans wadah Amazon ECS.

Setelah menetapkan grup instans penampung, Anda dapat menyesuaikan Amazon ECS untuk menempatkan tugas pada instans penampung berdasarkan grup. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS, dan Membuat penyebaran pembaruan bergulir Amazon ECS. Anda juga dapat menerapkan filter grup ketika mencantumkan instans kontainer.

Sintaksis ekspresi

Ekspresi memiliki sintaksis berikut:

```
subject operator [argument]
```

Subjek

Atribut atau bidang yang perlu dievaluasi.

`agentConnected`

Pilih instans kontainer berdasarkan status koneksi agen penampung Amazon ECS mereka. Anda dapat menggunakan filter ini untuk mencari instans dengan agen kontainer yang terputus.

Operasi yang valid: equals (==), not_equals (!=), in, not_in (!in), matches (=~), not_matches (!~)

`agentVersion`

Pilih instans kontainer berdasarkan versi agen penampung Amazon ECS mereka. Anda dapat menggunakan filter ini untuk menemukan instance yang menjalankan versi lama dari agen penampung Amazon ECS.

Operator yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

`attribute:`*`attribute-name`*

Pilih instans kontainer berdasarkan atribut. Untuk informasi selengkapnya, lihat Atribut instans wadah Amazon ECS.

`ec2InstanceId`

Pilih instans kontainer berdasarkan ID EC2 instans Amazon mereka.

Operasi yang valid: equals (==), not_equals (!=), in, not_in (!in), matches (=~), not_matches (!~)

`registeredAt`

Pilih instans kontainer berdasarkan tanggal pendaftaran instans kontainer mereka. Anda dapat menggunakan filter ini untuk menemukan instans yang baru terdaftar atau instans yang sangat usang.

Operasi yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

Format tanggal yang valid: 2018-06-18T22:28:28+00:00, 2018-06-18T22:28:28Z, 2018-06-18T22:28:28, 2018-06-18

`runningTasksCount`

Pilih instans kontainer berdasarkan jumlah tugas yang sedang berjalan. Anda dapat menggunakan filter ini untuk menemukan instans yang kosong atau hampir kosong (hanya beberapa tugas yang berjalan pada instans tersebut).

Operasi yang valid: equals (==), not_equals (!=), greater_than (>), greater_than_equal (>=), less_than (<), less_than_equal (<=)

`task:group`

Pilih instans kontainer oleh grup tugas. Untuk informasi selengkapnya, lihat Tugas Amazon ECS terkait grup .

Operator

Operasi perbandingan. Mendukung operasi berikut ini.

Operasi	Deskripsi
==, equals	Kesamaan string
!=, not_equals	Ketidaksamaan string

Operasi	Deskripsi
>, greater_than	Lebih besar dari
>=, greater_than_equal	Lebih besar dari atau sama dengan
<, less_than	Kurang dari
<=, less_than_equal	Kurang dari atau sama dengan
exists	Subjek ada
!exists, not_exists	Subjek tidak ada
in	Nilai dalam daftar argumen
!in, not_in	Nilai tidak dalam daftar argumen
=~, matches	Pola cocok
!~, not_matches	Pola tidak cocok

> **Note**
>
> Sebuah ekspresi tunggal tidak dapat berisi tanda kurung. Namun, tanda kurung dapat digunakan untuk menunjukkan prioritas dalam ekspresi majemuk.

Pendapat

Di sebagian besar operasi, argumen adalah nilai literal.

Operasi `in` dan `not_in` menganggap daftar argumen sebagai argumen. Anda menentukan daftar argumen sebagai berikut:

```
[argument1, argument2, ..., argumentN]
```

Operasi matches dan not_matches menerima argumen yang sesuai dengan sintaksis ekspresi reguler Java. Untuk informasi selengkapnya, lihat java.util.regex.Pattern.

Ekspresi majemuk

Anda dapat menggabungkan ekspresi menggunakan operasi Boolean berikut:

- &&, dan
- ||, atau
- !, tidak

Anda dapat menentukan prioritas menggunakan tanda kurung:

```
(expression1 or expression2) and expression3
```

Contoh ekspresi

Berikut ini adalah contoh-contoh ekspresi.

Contoh: String Sama Dengan

Ekspresi berikut memilih instans dengan tipe instans yang ditentukan.

```
attribute:ecs.instance-type == t2.small
```

Contoh: Daftar Argumen

Ekspresi berikut memilih instans di Availability Zone us-east-1a atau us-east-1b.

```
attribute:ecs.availability-zone in [us-east-1a, us-east-1b]
```

Contoh: Ekspresi Majemuk

Ekspresi berikut memilih instance G2 yang tidak berada di Availability Zone us-east-1d.

```
attribute:ecs.instance-type =~ g2.* and attribute:ecs.availability-zone != us-east-1d
```

Contoh: Afinitas Tugas

Ekspresi berikut memilih instans yang meng-host tugas di grup`service:production`.

```
task:group == service:production
```

Contoh: Afinitas Terbalik Tugas

Ekspresi berikut memilih instance yang tidak menghosting tugas dalam grup database.

```
not(task:group == database)
```

Contoh: Menjalankan penghitungan tugas

Ekspresi berikut memilih instans yang hanya menjalankan satu tugas.

```
runningTasksCount == 1
```

Contoh: Amazon ECS versi agen kontainer

Ekspresi berikut memilih instans yang menjalankan versi agen kontainer di bawah 1.14.5.

```
agentVersion < 1.14.5
```

Contoh: Waktu pendaftaran instans

Ekspresi berikut memilih instans yang terdaftar sebelum 13 Februari 2018.

```
registeredAt < 2018-02-13
```

Contoh: ID EC2 contoh Amazon

Ekspresi berikut memilih instance dengan instance Amazon EC2 berikut. IDs

```
ec2InstanceId in ['i-abcd1234', 'i-wxyx7890']
```

Contoh kendala penempatan tugas Amazon ECS

Berikut ini adalah contoh kendala penempatan tugas.

Contoh ini menggunakan `memberOf` kendala untuk menempatkan tugas pada instance t2. Ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, RegisterTaskDefinition, dan RunTask.

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

Contoh menggunakan `memberOf` kendala untuk menempatkan tugas replika pada instance dengan tugas di grup tugas layanan daemon, dengan menghormati strategi penempatan `daemon-service` tugas apa pun yang juga ditentukan. Batasan ini memastikan bahwa tugas layanan daemon ditempatkan pada EC2 instance sebelum tugas layanan replika.

Ganti `daemon-service` dengan nama layanan daemon.

```
"placementConstraints": [
    {
        "expression": "task:group == service:daemon-service",
        "type": "memberOf"
    }
]
```

Contoh tersebut menggunakan kendala `memberOf` untuk menempatkan tugas pada instans dengan tugas-tugas lain di kelompok tugas `databases`, perihal setiap strategi penempatan tugas yang juga ditentukan. Untuk informasi selengkapnya tentang grup tugas, lihat Tugas Amazon ECS terkait grup . Ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, RegisterTaskDefinition, dan RunTask.

```
"placementConstraints": [
    {
        "expression": "task:group == databases",
        "type": "memberOf"
    }
]
```

Kendala `distinctInstance` menempatkan setiap tugas dalam grup pada instans yang berbeda. Hal ini dapat ditentukan dengan tindakan berikut: CreateService, UpdateService, dan RunTask

Amazon ECS melihat status tugas yang diinginkan untuk penempatan tugas. Misalnya, jika status yang diinginkan dari tugas yang ada adalah`STOPPED`, (tetapi status terakhir tidak), tugas masuk

baru dapat ditempatkan pada instance yang sama meskipun ada kendala `distinctInstance` penempatan. Oleh karena itu, Anda mungkin melihat 2 tugas dengan status terakhir `RUNNING` pada instance yang sama.

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

Tugas mandiri Amazon ECS

Anda dapat menjalankan aplikasi Anda sebagai tugas ketika Anda memiliki aplikasi yang melakukan beberapa pekerjaan, dan kemudian berhenti, misalnya proses batch. Jika Anda ingin menjalankan tugas satu kali, Anda dapat menggunakan konsol, AWS CLI, APIs, atau SDKs.

Jika Anda perlu menjalankan aplikasi berdasarkan tarif, berbasis cron, atau jadwal satu kali, Anda dapat membuat jadwal menggunakan Scheduler. EventBridge

Alur kerja tugas

Saat Anda meluncurkan tugas Amazon ECS (tugas mandiri atau oleh layanan Amazon ECS), tugas dibuat dan awalnya dipindahkan ke status. `PROVISIONING` Saat tugas dalam `PROVISIONING` status, baik tugas maupun kontainer tidak ada karena Amazon ECS perlu menemukan kapasitas komputasi untuk menempatkan tugas.

Amazon ECS memilih kapasitas komputasi yang sesuai untuk tugas Anda berdasarkan jenis peluncuran atau konfigurasi penyedia kapasitas. Anda dapat menggunakan penyedia kapasitas dan strategi penyedia kapasitas dengan jenis EC2 peluncuran Fargate dan Amazon. Dengan Fargate, Anda tidak perlu memikirkan penyediaan, konfigurasi, dan penskalaan kapasitas cluster Anda. Fargate menangani semua manajemen infrastruktur untuk tugas-tugas Anda. Untuk jenis EC2 peluncuran, Anda dapat mengelola kapasitas klaster dengan mendaftarkan EC2 instans Amazon ke klaster, atau menggunakan penskalaan otomatis cluster untuk menyederhanakan manajemen kapasitas komputasi. Penskalaan otomatis cluster menangani penskalaan kapasitas cluster secara dinamis, sehingga Anda dapat fokus menjalankan tugas. Amazon ECS menentukan tempat untuk menempatkan tugas berdasarkan persyaratan yang Anda tentukan dalam definisi tugas, seperti CPU dan memori, serta batasan dan strategi penempatan Anda. Untuk informasi selengkapnya, lihat, Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

Jika Anda menggunakan penyedia kapasitas dengan penskalaan terkelola diaktifkan, tugas yang tidak dapat dimulai karena kurangnya kapasitas komputasi akan dipindahkan ke `PROVISIONING` status daripada gagal segera. Setelah menemukan kapasitas untuk menempatkan tugas Anda, Amazon ECS menyediakan lampiran yang diperlukan (misalnya, Antarmuka Jaringan Elastis (ENIs) untuk tugas dalam `awsvpc` mode). Ini menggunakan agen kontainer Amazon ECS untuk menarik gambar kontainer Anda, dan kemudian memulai wadah Anda. Setelah penyediaan selesai dan kontainer yang relevan diluncurkan, Amazon ECS memindahkan tugas ke status. `RUNNING` Untuk informasi tentang status tugas, lihatSiklus hidup tugas Amazon ECS.

Menjalankan aplikasi sebagai tugas Amazon ECS

Anda dapat membuat tugas untuk proses satu kali menggunakan. AWS Management Console

Untuk membuat tugas mandiri ()AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Konsol Amazon ECS memungkinkan Anda membuat tugas mandiri dari halaman detail klaster atau dari daftar revisi definisi tugas. Gunakan langkah-langkah berikut untuk membuat tugas mandiri Anda tergantung pada halaman sumber daya yang Anda pilih.

Untuk memulai layanan dari	Langkah-langkah
Halaman detail cluster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Tugas, pilih Jalankan tugas.
Halaman revisi definisi tugas	a. Pada halaman Definisi tugas, pilih keluarga definisi tugas untuk menampilkan revisi untuk keluarga tersebut. b. Pilih revisi yang ingin Anda gunakan. c. Dari menu Deploy, pilih Jalankan tugas.

3. Untuk klaster yang ada, pilih klaster.

   Pilih Buat klaster untuk menjalankan tugas di klaster baru

4. Pilih bagaimana tugas Anda didistribusikan di seluruh infrastruktur klaster Anda. Di bawah Konfigurasi komputasi, pilih opsi Anda.Untuk menggunakan strategi penyedia kapasitas, Anda harus mengonfigurasi penyedia kapasitas di tingkat klaster.

   Jika Anda belum mengonfigurasi klaster untuk menggunakan penyedia kapasitas, gunakan jenis peluncuran sebagai gantinya.

Metode distribusi	Langkah-langkah	
Strategi penyedia kapasitas	a. Di bagian Opsi komputasi , pilih Strategi penyedia kapasitas. b. Pilih strategi: • Untuk menggunak an strategi penyedia kapasitas default cluster, pilih Use cluster default. • Jika klaster Anda tidak memiliki strategi penyedia kapasitas default, atau menggunakan strategi khusus, pilih Gunakan strategi penyedia kustom, Tambahkan kapasitas, dan tentukan strategi penyedia kapasitas khusus Anda dengan menentuka n Basis, Penyedia Kapasitas, dan Berat.	

Metode distribusi	Langkah-langkah
	Note Untuk menggunakan penyedia kapasitas dalam suatu strategi, penyedia kapasitas harus dikaitkan dengan cluster.
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. (Opsional) Ketika tipe peluncuran Fargate diwajibkan, untuk Versi platform, tentukan versi platform yang akan digunakan. Jika versi platform tidak ditentuka n, versi LATEST platform akan digunakan.

5. Di bawah konfigurasi Deployment, lakukan hal berikut:

   a. Untuk definisi Tugas, masukkan definisi tugas.

   Important
   Konsol memvalidasi pilihan untuk memastikan bahwa keluarga dan revisi definisi tugas yang dipilih kompatibel dengan konfigurasi komputasi yang ditentukan.

   b. Untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan.

   c. Untuk grup Tugas, masukkan nama grup tugas.

6. Jika definisi tugas Anda menggunakan mode `awsvpc` jaringan, perluas Jaringan. Gunakan langkah-langkah berikut untuk menentukan konfigurasi kustom.

   a. Untuk VPC, pilih VPC yang akan digunakan.

   b. Untuk Subnet, pilih satu atau beberapa subnet di VPC yang dipertimbangkan oleh penjadwal tugas saat menempatkan tugas Anda.

   c. Untuk grup Keamanan, Anda dapat memilih grup keamanan yang ada atau membuat yang baru. Untuk menggunakan grup keamanan yang ada, pilih grup keamanan dan lanjutkan ke langkah berikutnya. Untuk membuat grup keamanan baru, pilih Buat grup keamanan baru. Anda harus menentukan nama grup keamanan, deskripsi, dan kemudian tambahkan satu atau beberapa aturan masuk untuk grup keamanan.

   d. Untuk IP Publik, pilih apakah akan menetapkan alamat IP publik secara otomatis ke antarmuka jaringan elastis (ENI) tugas.

      AWS Fargate tugas dapat diberikan alamat IP publik ketika dijalankan di subnet publik sehingga mereka memiliki rute ke internet. EC2 tugas tidak dapat diberikan IP publik menggunakan bidang ini. Untuk informasi selengkapnya, lihat opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate dan Alokasikan antarmuka jaringan untuk tugas Amazon ECS. .

7. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

   Nama volume dan jenis volume dikonfigurasi saat membuat revisi definisi tugas dan tidak dapat diubah saat Anda menjalankan tugas mandiri. Untuk memperbarui nama dan jenis volume, Anda harus membuat revisi definisi tugas baru dan menjalankan tugas dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda. b. Untuk Ukuran (GiB), masukkan nilai yang valid	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentuka n minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah input/output operasi maksimum (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk`io1`,`io2`, dan jenis `gp3` volume. d. Untuk Throughput (MIB/ s), masukkan throughpu t yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/ s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis `gp3` volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat volume baru yang kosong	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	dengan tidak memilih atau memasukkan ID snapshot. f. Jika Anda menentukan ID Snapshot, Anda dapat menentukan tingkat inisialisasi Volume (MIB/s). Masukkan nilai antara 100 dan 300, dalam MIB/s, yang akan menentukan seberapa cepat data dimuat dari snapshot yang ditentukan menggunakan ID Snapshot untuk pembuatan volume. g. Untuk kebijakan Penghentian, batalkan centang kotak jika Anda ingin volume yang dikonfigurasi untuk lampiran ke tugas dipertahankan setelah tugas dihentikan. Secara default, volume EBS yang dilampirkan ke tugas dihapus saat tugas dihentikan. h. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. i. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas. Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola ke peran, atau Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	diperlukan, lihatPeran IAM infrastruktur Amazon ECS. j. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigurasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan enkripsi default Amazon EBS tidak diaktifkan, volume tidak akan dienkripsi. Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume akan tidak dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	volume dari snapshot terenkripsi. k. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentuka n AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam kebijakan Anda. AWS KMS key Untuk informasi selengkapnya, lihat Enkripsi data untuk volume Amazon EBS. l. (Opsional) Di bawah Tag, Anda dapat menambahk an tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau dengan memberikan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	Definisi tugas untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk setiap tag yang Anda tambahkan. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

8. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas Penempatan Tugas, lalu pilih dari opsi berikut.

   Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

   - AZ Balanced Spread — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instance container di Availability Zone.
   - AZ Balanced BinPack — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instans kontainer dengan memori yang paling sedikit tersedia.
   - BinPack— Mendistribusikan tugas berdasarkan jumlah CPU atau memori yang paling sedikit tersedia.
   - Satu Tugas Per Host - Tempatkan, paling banyak, satu tugas dari layanan pada setiap instance kontainer.

- Kustom - Tentukan strategi penempatan tugas Anda sendiri.

Jika Anda memilih Kustom, tentukan algoritme untuk menempatkan tugas dan aturan yang dipertimbangkan selama penempatan tugas.

- Di bawah Strategi, untuk Jenis dan Bidang, pilih algoritma dan entitas yang akan digunakan untuk algoritme.

  Anda dapat memasukkan maksimal 5 strategi.

- Di bawah Constraint, untuk Type dan Expression, pilih aturan dan atribut untuk kendala.

  Misalnya, untuk menyetel batasan untuk menempatkan tugas pada instance T2, untuk Ekspresi, masukkan atribut:ecs.instance-type =~ t2. *.

  Anda dapat memasukkan maksimal 10 kendala.

9. (Opsional) Untuk mengganti peran IAM tugas, atau peran eksekusi tugas yang ditentukan dalam definisi tugas Anda, perluas penggantian Tugas, lalu selesaikan langkah-langkah berikut:

   a. Untuk peran Tugas, pilih peran IAM untuk tugas ini. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

      Hanya peran dengan hubungan kepercayaan `ecs-tasks.amazonaws.com` yang ditampilkan. Untuk petunjuk tentang cara membuat peran IAM untuk tugas Anda, lihatMembuat peran tugas IAM.

   b. Untuk peran eksekusi tugas, pilih peran eksekusi tugas. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

10. (Opsional) Untuk mengganti perintah container dan variabel lingkungan, perluas Container Overrides, lalu perluas container.

    - Untuk mengirim perintah ke wadah selain perintah definisi tugas, untuk penggantian Command, masukkan perintah Docker.
    - Untuk menambahkan variabel lingkungan, pilih Tambahkan Variabel Lingkungan. Untuk Kunci, masukkan nama variabel lingkungan Anda. Untuk Nilai, masukkan nilai string untuk nilai lingkungan Anda (tanpa tanda kutip ganda (" ")).

      AWS mengelilingi string dengan tanda kutip ganda (” “) dan meneruskan string ke wadah dalam format berikut:

```
MY_ENV_VAR="This variable contains a string."
```

11. (Opsional) Untuk membantu mengidentifikasi tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Definisi tugas.

    Menambah atau menghapus tanda.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.
12. Pilih Buat.

Menggunakan Amazon EventBridge Scheduler untuk menjadwalkan tugas Amazon ECS

EventBridge Scheduler adalah penjadwal tanpa server yang memungkinkan Anda membuat, menjalankan, dan mengelola tugas dari satu layanan terpusat yang dikelola. Ini menyediakan fungsionalitas penjadwalan satu kali dan berulang yang independen dari bus dan aturan acara. EventBridge Scheduler sangat dapat disesuaikan, dan menawarkan skalabilitas yang ditingkatkan dibandingkan aturan EventBridge terjadwal, dengan serangkaian operasi dan layanan API target yang lebih luas. AWS EventBridge Scheduler menyediakan jadwal berikut yang dapat Anda konfigurasi untuk tugas Anda di konsol EventBridge Scheduler:

- Berbasis tarif
- Berbasis cron

  Anda dapat mengonfigurasi jadwal berbasis cron di zona waktu mana pun.
- Jadwal satu kali

  Anda dapat mengonfigurasi jadwal satu kali di zona waktu mana pun.

Anda dapat menjadwalkan Amazon ECS Anda menggunakan Amazon EventBridge Scheduler.

Meskipun Anda dapat membuat tugas terjadwal di konsol Amazon ECS, saat ini konsol EventBridge Scheduler menyediakan lebih banyak fungsionalitas.

Selesaikan langkah-langkah berikut sebelum Anda menjadwalkan tugas:

1. Gunakan konsol VPC untuk mendapatkan subnet IDs tempat tugas dijalankan dan grup keamanan IDs untuk subnet. Untuk informasi selengkapnya, lihat Subnet untuk VPC Anda, dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan di Panduan Pengguna Amazon VPC.
2. Konfigurasikan peran eksekusi EventBridge Scheduler. Untuk informasi selengkapnya, lihat Mengatur peran eksekusi di Panduan Pengguna EventBridge Penjadwal Amazon.
3. Jika Anda ingin menggunakan strategi penyedia kapasitas untuk menjalankan tugas, Anda harus memiliki penyedia kapasitas yang terkait dengan cluster.

Untuk membuat jadwal baru menggunakan konsol

1. Buka konsol Amazon EventBridge Scheduler di https://console.aws.amazon.com/scheduler/rumah.
2. Pada halaman Jadwal, pilih Buat jadwal.
3. Pada halaman Tentukan detail jadwal, di bagian Nama jadwal dan deskripsi, lakukan hal berikut:
   a. Untuk nama Jadwal, masukkan nama untuk jadwal Anda. Misalnya, **`MyTestSchedule`**.
   b. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk jadwal Anda. Misalnya, **`TestSchedule`**.
   c. Untuk grup Jadwal, pilih grup jadwal. Jika Anda tidak memiliki grup, pilih default. Untuk membuat grup jadwal, pilih buat jadwal Anda sendiri.

      Anda menggunakan grup jadwal untuk menambahkan tag ke grup jadwal.
4. Pilih opsi jadwal Anda.

Kejadian	Lakukan ini...	
Jadwal satu kali Jadwal satu kali memanggil target hanya sekali pada	Untuk tanggal dan waktu, lakukan hal berikut:	

Kejadian	Lakukan ini...
tanggal dan waktu yang Anda tentukan.	• Masukkan tanggal yang valid dalam `YYYY/MM/DD` format. • Masukkan stempel waktu dalam format 24 jam`hh:mm`. • Untuk Timezone, pilih zona waktu.

Kejadian	Lakukan ini...
Jadwal berulang Jadwal berulang memanggil target pada tingkat yang Anda tentukan menggunak an cron ekspresi atau ekspresi tingkat.	a. Untuk jenis Jadwal, lakukan salah satu hal berikut: • Untuk menggunakan ekspresi cron untuk menentukan jadwal, pilih Jadwal berbasis Cron dan masukkan ekspresi cron. • Untuk menggunakan ekspresi tingkat untuk menentukan jadwal, pilih Jadwal berbasis tingkat dan masukkan ekspresi laju. Untuk informasi selengkapnya tentang ekspresi cron dan rate, lihat Menjadwalkan jenis pada EventBridge Scheduler di Panduan Pengguna EventBridge Penjadwal Amazon. b. Untuk jendela waktu Fleksibel, pilih Nonaktif untuk mematikan opsi, atau pilih salah satu jendela waktu yang telah ditentukan sebelumny a. Misalnya, jika Anda memilih 15 menit dan Anda menetapkan jadwal berulang untuk memanggil

Kejadian	Lakukan ini...
	targetnya setiap jam sekali, jadwal berjalan dalam 15 menit setelah dimulainya setiap jam.

5. (Opsional) Jika Anda memilih Jadwal berulang pada langkah sebelumnya, di bagian Jangka Waktu, lakukan hal berikut:

   a. Untuk Timezone, pilih zona waktu.

   b. Untuk Tanggal dan waktu mulai, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

   c. Untuk Tanggal dan waktu berakhir, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

6. Pilih Berikutnya.

7. Pada halaman Pilih target, lakukan hal berikut:

   a. Pilih Semua APIs, dan kemudian di kotak pencarian masukkan ECS.

   b. Pilih Amazon ECS.

   c. Di kotak pencarian, masukkan RunTask, lalu pilih RunTask.

   d. Untuk cluster ECS, pilih cluster.

   e. Untuk tugas ECS, pilih definisi tugas yang akan digunakan untuk tugas tersebut.

   f. Pilih bagaimana tugas Anda didistribusikan di seluruh infrastruktur klaster Anda. Perluas opsi Compute, lalu pilih salah satu opsi berikut

Opsi komputasi	Langkah-langkah
Strategi penyedia kapasitas	a. Pilih strategi penyedia Kapasitas. b. Pilih strategi: • Untuk menggunak an strategi penyedia kapasitas default, pilih Use cluster default.

Opsi komputasi	Langkah-langkah
	• Untuk menggunakan strategi kustom, pilih Gunakan kustom. Kemudian, masukkan Basis, Penyedia Kapasitas, dan Berat. Untuk EC2, penyedia Kapasitas adalah grup Auto Scaling.
Jenis peluncuran	a. Pilih Jenis Peluncuran. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. Ketika jenis peluncuran Fargate ditentukan, untuk versi Platform, tentukan versi platform yang akan digunakan.

g. Untuk Subnet, masukkan subnet IDs untuk menjalankan tugas di.

h. Untuk grup Keamanan, masukkan grup keamanan IDs untuk subnet.

i. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas batasan Penempatan, lalu masukkan batasan.

   Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

j. (Opsional) Untuk membantu mengidentifikasi tugas Anda, di bawah Tag, konfigurasikan tag Anda.

   Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS.

8. Pilih Berikutnya.

9. Pada halaman Pengaturan, lakukan hal berikut:

a. Untuk mengaktifkan jadwal, di bawah Status jadwal, alihkan Aktifkan jadwal.

b. Untuk mengonfigurasi kebijakan coba lagi untuk jadwal Anda, di bawah Kebijakan Coba ulang dan antrian surat mati (DLQ), lakukan hal berikut:

   - Beralih Coba lagi.
   - Untuk Waktu retensi maksimum acara, masukkan jam maksimum dan min yang harus disimpan oleh EventBridge Penjadwal acara yang belum diproses.
   - Waktu maksimum adalah 24 jam.
   - Untuk percobaan ulang Maksimum, masukkan jumlah maksimum kali EventBridge Scheduler mencoba ulang jadwal jika target mengembalikan kesalahan.

     Nilai maksimum adalah 185 percobaan ulang.

   Dengan kebijakan coba lagi, jika jadwal gagal memanggil targetnya, EventBridge Scheduler menjalankan kembali jadwal. Jika dikonfigurasi, Anda harus mengatur waktu retensi maksimum dan mencoba ulang untuk jadwal.

c. Pilih tempat EventBridge Scheduler menyimpan acara yang tidak terkirim.

Opsi antrian surat mati (DLQ)	Lakukan ini...
Jangan simpan	Pilih Tidak Ada.
Simpan acara di tempat yang sama Akun AWS di mana Anda membuat jadwal	a. Pilih Pilih antrian Amazon SQS di saya Akun AWS sebagai DLQ. b. Pilih Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.

Opsi antrian surat mati (DLQ)	Lakukan ini...
Simpan acara di tempat yang berbeda Akun AWS dari tempat Anda membuat jadwal	a. Pilih Tentukan antrean Amazon SQS di lain Akun AWS sebagai DLQ. b. Masukkan Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.

d. Untuk menggunakan kunci yang dikelola pelanggan untuk mengenkripsi input target Anda, di bawah Enkripsi, pilih Sesuaikan pengaturan enkripsi (lanjutan).

   Jika Anda memilih opsi ini, masukkan ARN kunci KMS yang ada atau pilih AWS KMS key Buat untuk menavigasi ke AWS KMS konsol. Untuk informasi selengkapnya tentang cara EventBridge Scheduler mengenkripsi data Anda saat istirahat, lihat Enkripsi saat istirahat di Panduan Pengguna EventBridge Penjadwal Amazon.

e. Untuk Izin, pilih Gunakan peran yang ada, lalu pilih peran.

   Agar EventBridge Scheduler membuat peran eksekusi baru untuk Anda, pilih Buat peran baru untuk jadwal ini. Kemudian, masukkan nama untuk nama Peran. Jika Anda memilih opsi ini, EventBridge Scheduler melampirkan izin yang diperlukan untuk target template Anda ke peran.

10. Pilih Berikutnya.
11. Di halaman Tinjau dan buat jadwal, tinjau detail jadwal Anda. Di setiap bagian, pilih Edit untuk kembali ke langkah itu dan mengedit detailnya.
12. Pilih Buat jadwal.

    Anda dapat melihat daftar jadwal baru dan yang sudah ada di halaman Jadwal. Di bawah kolom Status, verifikasi bahwa jadwal baru Anda Diaktifkan.

Langkah selanjutnya

Anda dapat menggunakan konsol EventBridge Scheduler atau AWS CLI untuk mengelola jadwal. Untuk informasi selengkapnya, lihat Mengelola jadwal di Panduan Pengguna EventBridge Penjadwal Amazon.

Menghentikan tugas Amazon ECS

Jika Anda tidak perlu lagi menjalankan tugas mandiri, Anda dapat menghentikan tugas tersebut. Konsol Amazon ECS memudahkan untuk menghentikan satu atau lebih tugas.

Anda tidak dapat memulai ulang tugas yang dihentikan mandiri.

Jika Anda ingin menghentikan layanan, lihatMenghapus layanan Amazon ECS menggunakan konsol.

Untuk menghentikan tugas mandiri ()AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster untuk menavigasi ke halaman detail cluster.
4. Pada halaman detail cluster, pilih tab Tugas.
5. Anda dapat memfilter tugas berdasarkan jenis peluncuran menggunakan daftar jenis peluncuran Filter.

Tugas untuk berhenti	Langkah-langkah
Satu atau lebih	a. Pilih tugas, lalu pilih Berhenti, Berhenti dipilih. b. Pada halaman Stop task confirmation, pilih Stop
Semua	**Important** Jika Anda memilih untuk menghenti kan semua tugas menggunakan

Tugas untuk berhenti	Langkah-langkah
	konsol, Amazon ECS menghentikan semua tugas dan tugas mandiri yang merupakan bagian dari layanan. Karena itu, kami sarankan berhati-hati saat menggunakan opsi ini. a. Pilih Stop, Stop all. b. Pada halaman Hentikan konfirmasi tugas, masukkan Hentikan semua tugas, lalu pilih Berhenti.

Layanan-layanan Amazon ECS

Anda dapat menggunakan layanan Amazon ECS untuk menjalankan dan mempertahankan jumlah instans definisi tugas yang ditentukan secara bersamaan di kluster Amazon ECS. Jika salah satu tugas Anda gagal atau berhenti, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantikannya. Ini membantu mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Anda juga dapat menjalankan layanan Anda secara opsional di belakang penyeimbang beban. Penyeimbang beban mendistribusikan lalu lintas ke seluruh tugas yang terkait dengan layanan.

Kami menyarankan Anda menggunakan penjadwal layanan untuk layanan dan aplikasi stateless yang berjalan lama. Penjadwal layanan memastikan bahwa strategi penjadwalan yang Anda tentukan diikuti dan menjadwal ulang tugas ketika tugas gagal. Misalnya, jika infrastruktur yang mendasarinya gagal, penjadwal layanan menjadwal ulang tugas. Anda dapat menggunakan strategi dan batasan penempatan tugas untuk menyesuaikan cara penjadwal menempatkan dan mengakhiri tugas. Jika

tugas dalam layanan berhenti, penjadwal meluncurkan tugas baru untuk menggantikannya. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas yang Anda inginkan berdasarkan strategi penjadwalan yang digunakan layanan. Strategi penjadwalan layanan juga disebut sebagai jenis layanan.

Penjadwal layanan juga menggantikan tugas yang ditentukan tidak sehat setelah pemeriksaan kesehatan kontainer atau pemeriksaan kesehatan kelompok sasaran penyeimbang beban gagal. Penggantian ini tergantung pada parameter definisi `maximumPercent` dan `desiredCount` layanan. Jika tugas ditandai tidak sehat, penjadwal layanan akan memulai tugas pengganti terlebih dahulu. Kemudian, hal berikut terjadi.

- Jika tugas penggantian memiliki status kesehatan`HEALTHY`, penjadwal layanan menghentikan tugas yang tidak sehat
- Jika tugas penggantian memiliki status kesehatan`UNHEALTHY`, penjadwal akan menghentikan tugas penggantian yang tidak sehat atau tugas tidak sehat yang ada untuk mendapatkan jumlah tugas total yang sama`desiredCount`.

Jika `maximumPercent` parameter membatasi penjadwal untuk memulai tugas penggantian terlebih dahulu, penjadwal akan menghentikan tugas yang tidak sehat satu per satu secara acak untuk membebaskan kapasitas, dan kemudian memulai tugas pengganti. Proses start dan stop berlanjut sampai semua tugas yang tidak sehat diganti dengan tugas-tugas yang sehat. Setelah semua tugas yang tidak sehat telah diganti dan hanya tugas sehat yang berjalan, jika jumlah tugas total melebihi`desiredCount`, tugas sehat dihentikan secara acak hingga jumlah tugas total sama. `desiredCount` Untuk informasi selengkapnya tentang `maximumPercent` dan`desiredCount`, lihat Parameter definisi layanan.

Penjadwal layanan menyertakan logika yang membatasi seberapa sering tugas dimulai ulang jika tugas berulang kali gagal dimulai. Jika tugas dihentikan tanpa memasukkan `RUNNING` status, penjadwal layanan mulai memperlambat upaya peluncuran dan mengirimkan pesan acara layanan. Perilaku ini mencegah sumber daya yang tidak perlu digunakan untuk tugas yang gagal sebelum Anda dapat menyelesaikan masalah. Setelah layanan diperbarui, penjadwal layanan melanjutkan perilaku penjadwalan normal. Untuk informasi selengkapnya, lihat Logika throttle layanan Amazon ECS dan Melihat pesan acara layanan Amazon ECS.

Opsi komputasi infrastruktur

Ada dua opsi komputasi yang mendistribusikan tugas Anda.

- Strategi penyedia kapasitas menyebabkan Amazon ECS mendistribusikan tugas Anda di satu atau di beberapa penyedia kapasitas.

  Untuk strategi penyedia kapasitas, konsol memilih opsi komputasi secara default. Berikut dijelaskan tentang urutan yang digunakan konsol untuk memilih default:

  - Jika klaster Anda memiliki strategi penyedia kapasitas default yang ditentukan, klaster akan dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki penyedia kapasitas Fargate yang ditambahkan ke klaster, strategi penyedia kapasitas khusus yang menggunakan penyedia `FARGATE` kapasitas dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki satu atau beberapa penyedia kapasitas grup Auto Scaling yang ditambahkan ke cluster, opsi Use custom (Advanced) akan dipilih dan Anda perlu menentukan strategi secara manual.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan dan tidak ada penyedia kapasitas yang ditambahkan ke cluster, jenis peluncuran Fargate akan dipilih.
- Jenis peluncuran menyebabkan Amazon ECS meluncurkan tugas kami secara langsung di Fargate atau pada instans yang terdaftar EC2 di kluster Anda.

  Secara default layanan dimulai di subnet di VPC cluster Anda.

Penskalaan otomatis layanan

Penskalaan otomatis layanan adalah kemampuan untuk menambah atau mengurangi jumlah tugas yang diinginkan di layanan Amazon ECS Anda secara otomatis. Amazon ECS memanfaatkan layanan Application Auto Scaling untuk menyediakan fungsionalitas ini.

Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.

Penyeimbangan beban layanan

Layanan Amazon ECS yang dihosting AWS Fargate mendukung Application Load Balancers, Network Load Balancers, dan Gateway Load Balancers. Gunakan tabel berikut untuk mempelajari jenis penyeimbang beban apa yang akan digunakan.

Jenis Load Balancer	Gunakan dalam kasus ini	
Penyeimbang Beban Aplikasi	Lalu lintas rute HTTP/HTTPS (atau lapisan 7). Application Load Balancers menawarkan beberapa fitur yang membuatnya menarik untuk digunakan dengan layanan Amazon ECS: • Setiap layanan dapat melayani lalu lintas dari beberapa penyeimbang beban dan mengekspo s beberapa port yang diseimbangkan beban dengan menentukan beberapa grup target. • Mereka didukung oleh tugas yang dihosting di Fargate dan EC2 instance. • Application Load Balancers memungkinkan kontainer untuk menggunakan pemetaan port host dinamis (sehingga beberapa tugas dari layanan yang sama diperbolehkan per instance kontainer). • Application Load Balancers mendukung routing berbasis jalur dan aturan prioritas (sehingga beberapa layanan dapat menggunakan port listener yang sama pada	

Jenis Load Balancer	Gunakan dalam kasus ini
	satu Application Load Balancer).
Penyeimbang Beban Jaringan	Rute lalu lintas TCP atau UDP (atau lapisan 4).
Penyeimbang Beban Gateway	Rute lalu lintas TCP atau UDP (atau lapisan 4). Gunakan peralatan virtual, seperti firewall, sistem deteksi dan pencegahan intrusi, dan sistem inspeksi paket dalam.

Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Layanan interkoneksi

Jika Anda memerlukan aplikasi untuk terhubung ke aplikasi lain yang berjalan sebagai layanan Amazon ECS, Amazon ECS menyediakan cara-cara berikut untuk melakukan ini tanpa penyeimbang beban:

- Service Connect - Memungkinkan service-to-service komunikasi dengan penemuan otomatis menggunakan nama pendek dan port standar.
- Penemuan layanan - Penemuan layanan menggunakan Route 53 untuk membuat namespace untuk layanan Anda, yang memungkinkannya dapat ditemukan melalui DNS.
- Amazon VPC Lattice - VPC Lattice adalah layanan jaringan aplikasi yang dikelola sepenuhnya untuk menghubungkan, mengamankan, dan memantau layanan Anda di beberapa akun dan. VPCs Ada biaya yang terkait dengannya.

Untuk informasi selengkapnya, lihat Interkoneksi layanan Amazon ECS.

Pengontrol dan strategi penyebaran layanan Amazon ECS

Sebelum Anda menerapkan layanan Anda, tentukan opsi untuk menerapkannya, dan fitur yang digunakan layanan.

Strategi penjadwalan

Ada dua strategi penjadwal layanan yang tersedia:

- `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh klaster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Strategi penjadwalan replika.
- `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Strategi penjadwalan daemon.

> **Note**
>
> Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan.

Strategi penjadwalan replika

Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di klaster Anda.

Untuk layanan yang menjalankan tugas di Fargate, saat penjadwal layanan meluncurkan tugas baru atau berhenti menjalankan tugas, penjadwal layanan menggunakan upaya terbaik untuk menjaga keseimbangan di seluruh Availability Zone. Anda tidak perlu menentukan strategi penempatan tugas atau kendala.

Saat membuat layanan yang menjalankan tugas pada EC2 instance, Anda dapat menentukan strategi dan batasan penempatan tugas secara opsional untuk menyesuaikan keputusan penempatan tugas. Jika tidak ada strategi penempatan tugas atau kendala yang ditentukan, maka secara default penjadwal layanan menyebarkan tugas di seluruh Availability Zones. Penjadwal layanan menggunakan logika berikut:

- Menentukan instance kontainer mana di klaster Anda yang dapat mendukung definisi tugas layanan Anda (misalnya, atribut CPU, memori, port, dan instance container yang diperlukan).
- Menentukan instance kontainer mana yang memenuhi batasan penempatan apa pun yang ditentukan untuk layanan.
- Bila Anda memiliki layanan replika yang bergantung pada layanan daemon (misalnya, tugas router log daemon yang perlu dijalankan sebelum tugas dapat menggunakan logging), buat batasan penempatan tugas yang memastikan bahwa tugas layanan daemon ditempatkan pada instance sebelum tugas layanan replika. EC2 Untuk informasi selengkapnya, lihat Contoh kendala penempatan tugas Amazon ECS.
- Ketika ada strategi penempatan yang ditentukan, gunakan strategi itu untuk memilih instance dari kandidat yang tersisa.
- Jika tidak ada strategi penempatan yang ditentukan, gunakan logika berikut untuk menyeimbangkan tugas di seluruh Availability Zone di klaster Anda:
  - Mengurutkan instance kontainer yang valid. Memberikan prioritas pada instance yang memiliki jumlah tugas yang berjalan paling sedikit untuk layanan ini di Availability Zone masing-masing. Misalnya, jika zona A memiliki satu tugas layanan berjalan, sedangkan zona B dan C masing-masing tidak memliki tugas layanan berjalan, maka instans kontainer yang valid, baik di zona B ataupun C dianggap optimal untuk penempatan.
  - Menempatkan tugas layanan baru pada instance container yang valid di Availability Zone yang optimal berdasarkan langkah sebelumnya. Menyukai instance kontainer dengan jumlah tugas yang berjalan paling sedikit untuk layanan ini.

Kami menyarankan Anda menggunakan fitur penyeimbangan kembali layanan ketika Anda menggunakan `REPLICA` strategi karena membantu memastikan ketersediaan tinggi untuk layanan Anda.

Strategi penjadwalan daemon

Strategi penjadwalan daemon men-deploy tepat satu tugas untuk setiap instans kontainer aktif yang memenuhi semua batasan penempatan tugas yang ditentukan di klaster Anda. Penjadwal layanan mengevaluasi batasan penempatan tugas untuk menjalankan tugas, dan menghentikan tugas yang tidak memenuhi batasan penempatan. Saat menggunakan strategi ini, Anda tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Service.

Amazon ECS menyimpan sumber daya komputasi instans kontainer termasuk CPU, memori, dan antarmuka jaringan untuk tugas daemon. Saat Anda meluncurkan layanan daemon di klaster dengan layanan replika lainnya, Amazon ECS memprioritaskan tugas daemon. Ini berarti bahwa tugas daemon adalah tugas pertama yang diluncurkan pada instance dan tugas terakhir untuk berhenti setelah semua tugas replika dihentikan. Strategi ini memastikan bahwa sumber daya tidak digunakan oleh tugas replika yang tertunda dan tersedia untuk tugas daemon.

Penjadwal layanan daemon tidak menempatkan tugas apa pun pada instance yang memiliki status. `DRAINING` Jika instance kontainer bertransisi ke `DRAINING` status, tugas daemon di atasnya dihentikan. Penjadwal layanan juga memantau saat instans kontainer baru ditambahkan ke klaster Anda dan menambahkan tugas daemon ke dalamnya.

Saat Anda menentukan konfigurasi penerapan, nilai untuk `maximumPercent` parameter harus `100` (ditentukan sebagai persentase), yang merupakan nilai default yang digunakan jika tidak disetel. Nilai default untuk `minimumHealthyPercent` parameter adalah `0` (ditentukan sebagai persentase).

Anda harus memulai ulang layanan ketika Anda mengubah batasan penempatan untuk layanan daemon. Amazon ECS secara dinamis memperbarui sumber daya yang dicadangkan pada instans yang memenuhi syarat untuk tugas daemon. Untuk instans yang sudah ada, penjadwal mencoba menempatkan tugas pada instans.

Penerapan baru dimulai ketika ada perubahan pada ukuran tugas atau reservasi sumber daya kontainer dalam definisi tugas. Penerapan baru juga dimulai saat memperbarui layanan atau mengatur revisi definisi tugas yang berbeda. Amazon ECS mengambil reservasi CPU dan memori yang diperbarui untuk daemon, dan kemudian memblokir kapasitas itu untuk tugas daemon.

Jika sumber daya tidak mencukupi untuk salah satu kasus di atas, hal-hal berikut akan terjadi:

- Penempatan tugas gagal.
- Sebuah CloudWatch peristiwa dihasilkan.
- Amazon ECS terus mencoba dan menjadwalkan tugas pada instance dengan menunggu sumber daya tersedia.
- Amazon ECS membebaskan instans cadangan yang tidak lagi memenuhi kriteria batasan penempatan dan menghentikan tugas daemon terkait.

Strategi penjadwalan daemon dapat digunakan dalam kasus-kasus berikut:

- Menjalankan kontainer aplikasi

- Menjalankan kontainer dukungan untuk tugas pencatatan, pemantauan dan pelacakan

Tugas yang menggunakan tipe peluncuran Fargate atau tipe pengontrol `EXTERNAL` penerapan `CODE_DEPLOY` atau tidak mendukung strategi penjadwalan daemon.

Saat penjadwal layanan berhenti menjalankan tugas, penjadwal layanan mencoba untuk menjaga keseimbangan di seluruh Availability Zone di klaster Anda. Penjadwal menggunakan logika berikut:

- Jika strategi penempatan ditentukan, gunakan strategi tersebut untuk memilih tugas mana yang akan diakhiri. Misalnya, jika layanan memiliki strategi penyebaran Availability Zone yang ditentukan, tugas dipilih yang meninggalkan tugas yang tersisa dengan spread terbaik.
- Jika tidak ada strategi penempatan yang ditentukan, gunakan logika berikut untuk menjaga keseimbangan di seluruh Availability Zone di klaster Anda:
  - Urutkan instance kontainer yang valid. Berikan prioritas pada instance yang memiliki jumlah tugas yang berjalan terbesar untuk layanan ini di Availability Zone masing-masing. Misalnya, jika zona A memiliki satu tugas layanan yang berjalan dan zona B dan C masing-masing memiliki dua tugas layanan yang berjalan, instance kontainer di zona B atau C dianggap optimal untuk penghentian.
  - Hentikan tugas pada instance container di Availability Zone optimal berdasarkan langkah sebelumnya. Mendukung instance kontainer dengan jumlah tugas yang berjalan terbesar untuk layanan ini.

Pengontrol penyebaran

Pengontrol penerapan adalah mekanisme yang menentukan bagaimana tugas digunakan untuk layanan Anda. Opsi yang valid adalah:

- ECS

  Saat membuat layanan yang menggunakan pengontrol `ECS` penyebaran, Anda dapat memilih di antara strategi penerapan berikut:

  - `ROLLING`: Saat Anda membuat layanan yang menggunakan strategi penerapan rolling update (`ROLLING`), penjadwal layanan Amazon ECS menggantikan tugas yang sedang berjalan dengan tugas baru. Jumlah tugas yang ditambahkan atau dihapus Amazon ECS dari layanan selama pembaruan bergulir dikendalikan oleh konfigurasi penyebaran layanan.

    Penerapan pembaruan bergulir paling cocok untuk skenario berikut:

- Pembaruan layanan bertahap: Anda perlu memperbarui layanan Anda secara bertahap tanpa membuat seluruh layanan offline sekaligus.
- Persyaratan sumber daya terbatas: Anda ingin menghindari biaya sumber daya tambahan untuk menjalankan dua lingkungan lengkap secara bersamaan (seperti yang dipersyaratkan oleh penerapan biru/hijau).
- Waktu penerapan yang dapat diterima: Aplikasi Anda dapat mentolerir proses penerapan yang lebih lama, karena pembaruan bergulir menggantikan tugas satu per satu.
- Tidak perlu roll back instan: Layanan Anda dapat mentolerir proses rollback yang memakan waktu beberapa menit, bukan detik.
- Proses penyebaran sederhana: Anda lebih memilih pendekatan penerapan langsung tanpa kerumitan mengelola beberapa lingkungan, grup target, dan pendengar.
- Tidak ada persyaratan penyeimbang beban: Layanan Anda tidak menggunakan atau memerlukan penyeimbang beban, Application Load Balancer, Network Load Balancer, atau Service Connect (yang diperlukan untuk penerapan). blue/green
- Aplikasi stateful: Aplikasi Anda mempertahankan status yang membuatnya sulit untuk menjalankan dua lingkungan paralel.
- Sensitivitas biaya: Anda ingin meminimalkan biaya penerapan dengan tidak menjalankan lingkungan duplikat selama penerapan.

Pembaruan bergulir adalah strategi penerapan default untuk layanan dan memberikan keseimbangan antara keamanan penerapan dan efisiensi sumber daya untuk banyak skenario aplikasi umum.

- `BLUE_GREEN`Strategi penyebaran biru/hijau (`BLUE_GREEN`) adalah metodologi rilis yang mengurangi waktu henti dan risiko dengan menjalankan dua lingkungan produksi identik yang disebut biru dan hijau. Dengan blue/green penerapan Amazon ECS, Anda dapat memvalidasi revisi layanan baru sebelum mengarahkan lalu lintas produksi kepada mereka. Pendekatan ini memberikan cara yang lebih aman untuk menerapkan perubahan dengan kemampuan untuk memutar kembali dengan cepat jika diperlukan.

  blue/green Penerapan Amazon ECS paling cocok untuk skenario berikut:

  - Validasi layanan: Ketika Anda perlu memvalidasi revisi layanan baru sebelum mengarahkan lalu lintas produksi ke mereka
  - Zero downtime: Saat layanan Anda memerlukan penerapan zero-downtime
  - Instant roll back: Ketika Anda membutuhkan kemampuan untuk memutar kembali dengan cepat jika masalah terdeteksi

    - Persyaratan penyeimbang beban: Ketika layanan Anda menggunakan Application Load Balancer, Network Load Balancer, atau Service Connect
- Eksternal

  Gunakan pengontrol penerapan pihak ketiga.
- Penerapan biru/hijau (didukung oleh) AWS CodeDeploy

  CodeDeploy menginstal versi aplikasi yang diperbarui sebagai set tugas pengganti baru dan mengalihkan lalu lintas produksi dari tugas aplikasi asli yang disetel ke set tugas pengganti. Set tugas asli dihentikan setelah penerapan berhasil. Gunakan pengontrol penerapan ini untuk memverifikasi penyebaran layanan baru sebelum mengirim lalu lintas produksi ke sana.

Deteksi kegagalan penerapan Amazon ECS

Amazon ECS menyediakan dua metode untuk mendeteksi kegagalan penerapan:

- Penyebaran Pemutus Sirkuit
- CloudWatch Alarm

Kedua metode dapat dikonfigurasi untuk secara otomatis mengembalikan penerapan yang gagal ke keadaan baik terakhir yang diketahui.

Pertimbangkan hal berikut:

- Kedua metode hanya mendukung jenis penerapan dan blue/green penerapan pembaruan bergulir.
- Ketika kedua metode digunakan, keduanya dapat memicu kegagalan penerapan.
- Metode rollback memerlukan penerapan sebelumnya dalam status COMPLETED.
- EventBridge peristiwa dihasilkan untuk perubahan status penerapan.

Bagaimana pemutus sirkuit penyebaran Amazon ECS mendeteksi kegagalan

Pemutus sirkuit penyebaran adalah mekanisme pembaruan bergulir yang menentukan apakah tugas mencapai kondisi mapan. Pemutus sirkuit penyebaran memiliki opsi yang secara otomatis akan mengembalikan penerapan yang gagal ke penerapan yang ada dalam status. `COMPLETED`

Saat penerapan layanan mengubah status, Amazon ECS mengirimkan peristiwa perubahan status penerapan layanan ke. EventBridge Hal ini menyediakan cara terprogram untuk memantau status

deployment layanan Anda. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS. Kami menyarankan Anda membuat dan memantau EventBridge aturan dengan angka `eventName` `SERVICE_DEPLOYMENT_FAILED` sehingga Anda dapat mengambil tindakan manual untuk memulai penerapan Anda. Untuk informasi selengkapnya, lihat Memulai EventBridge di Panduan EventBridge Pengguna Amazon.

Ketika pemutus sirkuit penyebaran menentukan bahwa penerapan gagal, ia mencari penerapan terbaru yang berada dalam keadaan. `COMPLETED` Ini adalah penerapan yang digunakannya sebagai penerapan roll-back. Saat rollback dimulai, penerapan berubah dari a ke. `COMPLETED` `IN_PROGRESS` Ini berarti bahwa penerapan tidak memenuhi syarat untuk rollback lain hingga mencapai status. `COMPLETED` Ketika pemutus sirkuit penyebaran tidak menemukan penyebaran yang dalam `COMPLETED` keadaan, pemutus sirkuit tidak meluncurkan tugas baru dan penyebaran terhenti.

Saat Anda membuat layanan, penjadwal melacak tugas yang gagal diluncurkan dalam dua tahap.

- Tahap 1 - Penjadwal memantau tugas untuk melihat apakah mereka bertransisi ke status RUNNING.
  - Sukses - Penerapan memiliki peluang untuk beralih ke status SELESAI karena ada lebih dari satu tugas yang dialihkan ke status RUNNING. Kriteria kegagalan dilewati dan pemutus sirkuit bergerak ke tahap 2.
  - Kegagalan - Ada tugas berturut-turut yang tidak bertransisi ke status RUNNING dan penerapan mungkin bertransisi ke status GAGAL.
- Tahap 2 - Penerapan memasuki tahap ini ketika setidaknya ada satu tugas dalam status RUNNING. Pemutus sirkuit memeriksa pemeriksaan kesehatan untuk tugas-tugas dalam penyebaran saat ini yang sedang dievaluasi. Pemeriksaan kesehatan yang divalidasi adalah Elastic Load Balancing AWS Cloud Map , pemeriksaan kesehatan pelayanan, dan pemeriksaan kesehatan kontainer.
  - Sukses - Setidaknya ada satu tugas dalam keadaan berjalan dengan pemeriksaan kesehatan yang telah berlalu.
  - Kegagalan - Tugas yang diganti karena kegagalan pemeriksaan kesehatan telah mencapai ambang kegagalan.

Pertimbangkan hal berikut ketika Anda menggunakan metode deployment circuit breaker pada layanan. EventBridge menghasilkan aturan.

- `DescribeServices`Respons memberikan wawasan tentang keadaan penyebaran, `rolloutState` dan`rolloutStateReason`. Saat deployment baru dimulai, status peluncuran dimulai dalam status `IN_PROGRESS`. Saat layanan mencapai kondisi stabil, status peluncuran bertransisi ke `COMPLETED`. Jika layanan gagal mencapai kondisi tunak dan pemutus sirkuit dihidupkan, penyebaran akan beralih ke keadaan. `FAILED` Penerapan dalam `FAILED` status tidak meluncurkan tugas baru apa pun.
- Selain peristiwa perubahan status penerapan layanan yang dikirim Amazon ECS untuk penerapan yang telah dimulai dan telah selesai, Amazon ECS juga mengirimkan peristiwa ketika penerapan dengan pemutus sirkuit diaktifkan gagal. Kejadian ini menyediakan detail tentang mengapa deployment gagal atau jika deployment dimulai karena rollback. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS.
- Jika penerapan baru dimulai karena penerapan sebelumnya gagal dan terjadi rollback, `reason` bidang peristiwa perubahan status penerapan layanan menunjukkan penerapan dimulai karena rollback.
- Pemutus sirkuit penyebaran hanya didukung untuk layanan Amazon ECS yang menggunakan pengontrol penyebaran pembaruan bergulir (`ECS`).
- Anda harus menggunakan konsol Amazon ECS, atau AWS CLI ketika Anda menggunakan pemutus sirkuit penyebaran dengan opsi. CloudWatch Untuk informasi selengkapnya, lihat the section called “Buat layanan menggunakan parameter yang ditentukan” dan buat layanan di Referensi.AWS Command Line Interface

`create-service` AWS CLI Contoh berikut menunjukkan cara membuat layanan Linux ketika pemutus sirkuit deployment digunakan dengan opsi rollback.

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
 \
     --task-definition sample-fargate:1 \
     --launch-type FARGATE \
     --platform-family LINUX \
     --platform-version 1.4.0 \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

Contoh:

Deployment 1 dalam `COMPLETED` keadaan.

Deployment 2 tidak dapat dimulai, sehingga pemutus sirkuit berputar kembali ke Deployment 1. Penerapan 1 transisi ke negara bagian. `IN_PROGRESS`

Deployment 3 dimulai dan tidak ada penerapan di `COMPLETED` status, jadi Deployment 3 tidak dapat memutar kembali, atau meluncurkan tugas.

Ambang batas kegagalan

Pemutus sirkuit penyebaran menghitung nilai ambang batas, dan kemudian menggunakan nilai untuk menentukan kapan harus memindahkan penyebaran ke status. `FAILED`

Pemutus sirkuit penyebaran memiliki ambang minimum 3 dan ambang maksimum 200. dan menggunakan nilai-nilai dalam rumus berikut untuk menentukan kegagalan penerapan.

```
Minimum threshold <= 0.5 * desired task count => maximum threshold
```

Ketika hasil perhitungan lebih besar dari minimum 3, tetapi lebih kecil dari maksimum 200, ambang kegagalan diatur ke ambang batas yang dihitung (dibulatkan ke atas).

> **Note**
>
> Anda tidak dapat mengubah salah satu nilai ambang batas.

Ada dua tahap untuk pemeriksaan status penerapan.

1. Pemutus sirkuit penyebaran memantau tugas-tugas yang merupakan bagian dari penyebaran dan memeriksa tugas-tugas yang ada di negara bagian. `RUNNING` Penjadwal mengabaikan kriteria kegagalan ketika tugas dalam penerapan saat ini dalam `RUNNING` keadaan dan melanjutkan ke tahap berikutnya. Ketika tugas gagal dicapai di `RUNNING` negara bagian, pemutus sirkuit penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan ambang batas, penerapan ditandai sebagai. `FAILED`
2. Tahap ini dimasukkan ketika ada satu atau lebih tugas di `RUNNING` negara bagian. Pemutus sirkuit penyebaran melakukan pemeriksaan kesehatan pada sumber daya berikut untuk tugas-tugas dalam penerapan saat ini:
   - Penyeimbang beban Elastic Load Balancing

- AWS Cloud Map layanan
- Pemeriksaan kesehatan wadah Amazon ECS

Ketika pemeriksaan kesehatan gagal untuk tugas tersebut, pemutus sirkuit penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan ambang batas, penerapan ditandai sebagai. `FAILED`

Tabel berikut menunjukkan beberapa contoh.

Hitungan tugas yang diinginkan	Penghitungan	Ambang
1	`3 <= 0.5 * 1 => 200`	3 (nilai yang dihitung kurang dari minimum)
25	`3 <= 0.5 * 25 => 200`	13 (nilainya dibulatkan ke atas)
400	`3 <= 0.5 * 400 => 200`	200
800	`3 <= 0.5 * 800 => 200`	200 (nilai yang dihitung lebih besar dari maksimum)

Misalnya, ketika ambang batas adalah 3, pemutus sirkuit dimulai dengan jumlah kegagalan yang ditetapkan pada 0. Ketika tugas gagal mencapai `RUNNING` status, pemutus sirkuit penyebaran meningkatkan jumlah kegagalan satu per satu. Ketika jumlah kegagalan sama dengan 3, penerapan ditandai sebagai. `FAILED`

Untuk contoh tambahan tentang cara menggunakan opsi rollback, lihat Mengumumkan pemutus sirkuit penyebaran Amazon ECS.

Bagaimana CloudWatch alarm mendeteksi kegagalan penerapan Amazon ECS

Anda dapat mengonfigurasi Amazon ECS untuk menyetel penerapan menjadi gagal saat mendeteksi bahwa CloudWatch alarm tertentu telah masuk ke status. `ALARM`

Anda dapat mengatur konfigurasi secara opsional untuk mengembalikan penerapan yang gagal ke penerapan yang terakhir selesai.

`create-service` AWS CLI Contoh berikut menunjukkan cara membuat layanan Linux ketika alarm penyebaran digunakan dengan opsi rollback.

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration
 "alarms={alarmNames=[alarm1Name,alarm2Name],enable=true,rollback=true}" \
     --task-definition sample-fargate:1 \
     --launch-type FARGATE \
     --platform-family LINUX \
     --platform-version 1.4.0 \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

Pertimbangkan hal berikut saat Anda menggunakan metode CloudWatch alarm Amazon pada suatu layanan.

- Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser. Amazon ECS menghitung periode waktu ini berdasarkan konfigurasi alarm yang terkait dengan penerapan. Anda tidak dapat menetapkan nilai ini.
- Parameter `deploymentConfiguration` permintaan sekarang berisi tipe `alarms` data. Anda dapat menentukan nama alarm, apakah akan menggunakan metode ini, dan apakah akan memulai rollback ketika alarm menunjukkan kegagalan penerapan. Untuk informasi selengkapnya, lihat CreateServicedi Referensi API Amazon Elastic Container Service.
- `DescribeServices`Respons memberikan wawasan tentang keadaan penyebaran, `rolloutState` dan`rolloutStateReason`. Ketika penerapan baru dimulai, status peluncuran dimulai dalam suatu keadaan. `IN_PROGRESS` Ketika layanan mencapai kondisi mapan dan waktu pemanggangan selesai, status peluncuran beralih ke. `COMPLETED` Jika layanan gagal mencapai kondisi mapan dan alarm telah masuk ke `ALARM` status, penyebaran akan beralih ke `FAILED` keadaan. Sebuah deployment dalam status `FAILED` tidak akan meluncurkan tugas baru apa pun.
- Selain peristiwa perubahan status penerapan layanan yang dikirim Amazon ECS untuk penerapan yang telah dimulai dan telah selesai, Amazon ECS juga mengirimkan peristiwa ketika penerapan yang menggunakan alarm gagal. Kejadian ini menyediakan detail tentang mengapa deployment gagal atau jika deployment dimulai karena rollback. Untuk informasi selengkapnya, lihat Acara perubahan status penerapan layanan Amazon ECS.

- Jika penerapan baru dimulai karena penerapan sebelumnya gagal dan rollback diaktifkan, `reason` bidang peristiwa perubahan status penerapan layanan akan menunjukkan penerapan dimulai karena rollback.
- Jika Anda menggunakan pemutus sirkuit penerapan dan CloudWatch alarm Amazon untuk mendeteksi kegagalan, salah satu dapat memulai kegagalan penerapan segera setelah kriteria untuk salah satu metode terpenuhi. Rollback terjadi ketika Anda menggunakan opsi rollback untuk metode yang memulai kegagalan penerapan.
- CloudWatch Alarm Amazon hanya didukung untuk layanan Amazon ECS yang menggunakan pengontrol penyebaran pembaruan bergulir (`ECS`).
- Anda dapat mengonfigurasi opsi ini dengan menggunakan konsol Amazon ECS, atau. AWS CLIUntuk informasi selengkapnya, lihat the section called “Buat layanan menggunakan parameter yang ditentukan” dan buat layanan di Referensi.AWS Command Line Interface
- Anda mungkin memperhatikan bahwa status penerapan tetap `IN_PROGRESS` untuk waktu yang lama. Alasan untuk ini adalah bahwa Amazon ECS tidak mengubah status sampai telah menghapus penerapan aktif, dan ini tidak terjadi sampai setelah waktu pemanggangan. Bergantung pada konfigurasi alarm Anda, penerapan mungkin tampak memakan waktu beberapa menit lebih lama daripada saat Anda tidak menggunakan alarm (meskipun set tugas utama baru ditingkatkan dan penerapan lama diperkecil). Jika Anda menggunakan CloudFormation batas waktu, pertimbangkan untuk meningkatkan batas waktu. Untuk informasi selengkapnya, lihat Membuat kondisi tunggu di templat di Panduan AWS CloudFormation Pengguna.
- Amazon ECS memanggil `DescribeAlarms` untuk melakukan polling alarm. Panggilan untuk `DescribeAlarms` menghitung kuota CloudWatch layanan yang terkait dengan akun Anda. Jika Anda memiliki AWS layanan lain yang meneleponDescribeAlarms, mungkin ada dampak pada Amazon ECS untuk melakukan polling alarm. Misalnya, jika layanan lain membuat `DescribeAlarms` panggilan yang cukup untuk mencapai kuota, layanan itu dibatasi dan Amazon ECS juga dibatasi dan tidak dapat melakukan polling alarm. Jika alarm dihasilkan selama periode pelambatan, Amazon ECS mungkin melewatkan alarm dan pemutaran kembali mungkin tidak terjadi. Tidak ada dampak lain pada penyebaran. Untuk informasi selengkapnya tentang kuota CloudWatch layanan, lihat kuota CloudWatch layanan di CloudWatch Panduan Pengguna.
- Jika alarm dalam `ALARM` keadaan di awal penerapan, Amazon ECS tidak akan memantau alarm selama penyebaran itu (Amazon ECS mengabaikan konfigurasi alarm). Perilaku ini membahas kasus di mana Anda ingin memulai penerapan baru untuk memperbaiki kegagalan penerapan awal.

Alarm-alarm yang direkomendasikan

Kami menyarankan Anda menggunakan metrik alarm berikut:

- Jika Anda menggunakan Application Load Balancer, gunakan metrik `HTTPCode_ELB_5XX_Count` dan `HTTPCode_ELB_4XX_Count` Application Load Balancer. Metrik ini memeriksa lonjakan HTTP. Untuk informasi selengkapnya tentang metrik Application Load Balancer, lihat CloudWatch metrik untuk Application Load Balancer di Panduan Pengguna untuk Application Load Balancer.
- Jika Anda memiliki aplikasi yang sudah ada, gunakan `MemoryUtilization` metrik `CPUUtilization` dan. Metrik ini memeriksa persentase CPU dan memori yang digunakan cluster atau layanan. Untuk informasi selengkapnya, lihat the section called “Pertimbangan”.
- Jika Anda menggunakan Amazon Simple Queue Service antrian dalam tugas, gunakan metrik Amazon `ApproximateNumberOfMessagesNotVisible` SQS. Metrik ini memeriksa jumlah pesan dalam antrian yang tertunda dan tidak tersedia untuk dibaca segera. Untuk informasi selengkapnya tentang metrik Amazon SQS, lihat Metrik yang tersedia untuk CloudWatch Amazon SQS di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Kait siklus hidup untuk penerapan layanan Amazon ECS

Ketika penerapan dimulai, ia melewati tahapan siklus hidup. Tahapan ini bisa dalam keadaan seperti IN_PROGRESS atau berhasil. Anda dapat menggunakan kait siklus hidup, yang merupakan fungsi Lambda yang dijalankan Amazon ECS atas nama Anda pada tahapan siklus hidup yang ditentukan. Fungsinya dapat berupa salah satu dari yang berikut:

- API asinkron yang memvalidasi pemeriksaan kesehatan dalam waktu 15 menit.
- API polling yang memulai proses asinkron lain yang mengevaluasi penyelesaian kait siklus hidup.

Setelah fungsi selesai berjalan, ia harus mengembalikan a `hookStatus` agar penerapan dapat dilanjutkan. Jika a `hookStatus` tidak dikembalikan, atau jika fungsi gagal, penerapan akan kembali. Berikut ini adalah `hookStatus` nilainya:

- `SUCCEEDED`- penyebaran berlanjut ke tahap siklus hidup berikutnya
- `FAILED`— penyebaran kembali ke penerapan terakhir yang berhasil.
- `IN_PROGRESS`- Amazon ECS menjalankan fungsi lagi setelah periode waktu yang singkat. Secara default ini adalah interval 30 detik, namun nilai ini dapat disesuaikan dengan mengembalikan a `callBackDelaySeconds` di samping. `hookStatus`

Contoh berikut menunjukkan cara mengembalikan `hookStatus` dengan penundaan callback kustom. Dalam contoh ini, Amazon ECS akan mencoba lagi hook ini dalam 60 detik, bukan default 30 detik:

```
{
    "hookStatus": "IN_PROGRESS",
    "callBackDelaySeconds": 60
}
```

Ketika roll back terjadi, Amazon ECS menjalankan kait siklus hidup untuk tahapan siklus hidup berikut:

- PRODUCTION_TRAFFIC_SHIFT
- TEST_TRAFFIC_SHIFT

Muatan siklus hidup

Saat Anda mengonfigurasi kait siklus hidup untuk penerapan layanan ECS, Amazon ECS akan memanggil kait ini pada tahapan tertentu dari proses penerapan. Setiap tahap siklus hidup menyediakan muatan JSON dengan informasi tentang status penerapan saat ini. Dokumen ini menjelaskan struktur payload untuk setiap tahap siklus hidup.

Struktur muatan umum

Semua muatan tahap siklus hidup mencakup bidang umum berikut:

- `serviceArn`- Nama Sumber Daya Amazon (ARN) dari layanan.
- `targetServiceRevisionArn`- ARN dari revisi layanan target sedang dikerahkan.
- `testTrafficWeights`- Peta revisi layanan ARNs untuk persentase berat lalu lintas uji yang sesuai.
- `productionTrafficWeights`- Peta revisi layanan ARNs untuk persentase berat lalu lintas produksi yang sesuai.

Muatan tahap siklus hidup

RECONCILE_SERVICE

Tahap ini terjadi pada awal proses penyebaran ketika layanan sedang direkonsiliasi. Berikut ini menunjukkan contoh payload untuk tahap siklus hidup ini.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
 100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  },
  "productionTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
 100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  }
}
```

Harapan pada tahap ini:

- Set tugas utama adalah pada skala 0%

PRE_SCALE_UP

Tahap ini terjadi sebelum tugas baru ditingkatkan. Berikut ini menunjukkan contoh payload untuk tahap siklus hidup ini.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

Harapan pada tahap ini:

- Tugas revisi layanan hijau berada pada skala 0%

POST_SCALE_UP

Tahap ini terjadi setelah tugas-tugas baru ditingkatkan dan sehat. Berikut ini menunjukkan contoh payload untuk tahap siklus hidup ini.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

Harapan pada tahap ini:

- Tugas revisi layanan hijau berada pada skala 100%
- Tugas dalam revisi layanan hijau sehat

TEST_TRAFFIC_SHIFT

Tahap ini terjadi ketika lalu lintas uji sedang dialihkan ke tugas revisi layanan hijau.

Berikut ini menunjukkan contoh payload untuk tahap siklus hidup ini.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
 100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
```

```
  },
  "productionTrafficWeights": {}
}
```

Harapan pada tahap ini:

- Lalu lintas uji sedang dalam proses bergerak menuju tugas revisi layanan hijau.

POST_TEST_TRAFFIC_SHIFT

Tahap ini terjadi setelah lalu lintas uji telah sepenuhnya bergeser ke tugas-tugas baru.

Berikut ini menunjukkan contoh payload untuk tahap siklus hidup ini.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

Harapan pada tahap ini:

- 100% lalu lintas uji bergerak menuju tugas revisi layanan hijau.

PRODUCTION_TRAFFIC_SHIFT

Tahap ini terjadi ketika lalu lintas produksi sedang dialihkan ke tugas revisi layanan hijau.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
 100,
```

```
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  }
}
```

Harapan pada tahap ini:

- Lalu lintas produksi sedang dalam proses pindah ke revisi layanan hijau.

POST_PRODUCTION_TRAFFIC_SHIFT

Tahap ini terjadi setelah lalu lintas produksi sepenuhnya dialihkan ke tugas revisi layanan hijau.

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

Harapan pada tahap ini:

- 100% lalu lintas produksi bergerak menuju tugas revisi layanan hijau.

Kategori tahap siklus hidup

Tahapan siklus hidup terbagi dalam dua kategori:

1. Tahap pemanggilan tunggal - Tahapan ini dipanggil hanya sekali selama penerapan layanan:
   - PRE_SCALE_UP
   - POST_SCALE_UP
   - POST_TEST_TRAFFIC_SHIFT
   - POST_PRODUCTION_TRAFFIC_SHIFT
2. Tahapan pemanggilan berulang - Tahapan ini mungkin dipanggil beberapa kali selama penerapan layanan, misalnya ketika operasi rollback terjadi:
   - TEST_TRAFFIC_SHIFT

- PRODUCTION_TRAFFIC_SHIFT

Status penerapan selama kait siklus hidup

Saat kait siklus hidup berjalan, status penerapan akan berlaku `IN_PROGRESS` untuk semua tahapan siklus hidup.

Tahap Siklus Hidup	Status Penerapan
RECONCILE_SERVICE	IN_PROGRESS
PRE_SCALE_UP	IN_PROGRESS
POST_SCALE_UP	IN_PROGRESS
TEST_TRAFFIC_SHIFT	IN_PROGRESS
POST_TEST_TRAFFIC_SHIFT	IN_PROGRESS
PRODUCTION_TRAFFIC_SHIFT	IN_PROGRESS
POST_PRODUCTION_TRAFFIC_SHIFT	IN_PROGRESS

Menghentikan penyebaran layanan Amazon ECS

Anda dapat menghentikan penerapan secara manual saat penerapan yang gagal tidak terdeteksi oleh pemutus sirkuit atau alarm. CloudWatch Jenis stop berikut tersedia:

- Rollback - Opsi ini mengembalikan penyebaran layanan ke revisi layanan sebelumnya.

  Anda dapat menggunakan opsi ini bahkan jika Anda tidak mengonfigurasi penyebaran layanan untuk opsi rollback.

Anda dapat menghentikan penerapan yang ada di salah satu status berikut. Untuk informasi selengkapnya tentang status penerapan layanan, lihatMelihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

- PENDING - Penyebaran layanan bergerak ke status ROLLBACK_REQUESTED, dan kemudian operasi rollback dimulai.

- IN_PROGRESS - Penyebaran layanan bergerak ke status ROLLBACK_REQUESTED, dan kemudian operasi rollback dimulai.
- STOP_REQUESTED - Penyebaran layanan terus berhenti.
- ROLLBACK_REQUESTED - Penyebaran layanan melanjutkan operasi rollback.
- ROLLBACK_IN_PROGRESS - Penyebaran layanan melanjutkan operasi rollback.

Prosedur

Sebelum Anda mulai, konfigurasikan izin yang diperlukan untuk melihat penerapan layanan. Untuk informasi selengkapnya, lihat Izin diperlukan untuk melihat penerapan layanan Amazon ECS.

Amazon ECS Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan ditampilkan.
4. Pada halaman detail layanan, pilih Deployment.

   Halaman penerapan ditampilkan.
5. Di bawah Penerapan yang sedang berlangsung, pilih Roll back. Kemudian, di jendela konfirmasi, pilih Gulung kembali.

AWS CLI

1. Jalankan `list-service-deployments` untuk mengambil ARN penyebaran layanan.

   Ganti *`user-input`* dengan nilai-nilai Anda.

   ```
   aws ecs list-service-deployments --cluster cluster-name --service service-name
   ```

   Perhatikan `serviceDeploymentArn` untuk penerapan yang ingin Anda hentikan.

   ```
   {
       "serviceDeployments": [
           {
   ```

```
            "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
deployment/cluster-name/service-name/NCWGC2ZR-taawPAYrIaU5",
            "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/cluster-
name/service-name",
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/cluster-
name",
            "targetServiceRevisionArn": "arn:aws:ecs:us-
west-2:123456789012:service-revision/cluster-name/service-
name/4980306466373577095",
            "status": "SUCCESSFUL"
        }
    ]
}
```

2. Jalankan `stop-service-deployments`. Gunakan `serviceDeploymentArn` yang dikembalikan dari`list-service-deployments`.

   Ganti *`user-input`* dengan nilai-nilai Anda.

```
aws ecs stop-service-deployment --service-deployment-arn
 arn:aws:ecs:region:123456789012:service-deployment/cluster-name/service-
name/NCWGC2ZR-taawPAYrIaU5 --stop-type ROLLBACK
```

Langkah selanjutnya

Putuskan perubahan apa yang perlu dilakukan pada layanan, dan kemudian perbarui layanan. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Menyebarkan layanan Amazon ECS dengan mengganti tugas

Saat Anda membuat layanan yang menggunakan tipe penerapan rolling update (`ECS`), penjadwal layanan Amazon ECS menggantikan tugas yang sedang berjalan dengan tugas baru. Jumlah tugas yang ditambahkan atau dihapus Amazon ECS dari layanan selama pembaruan bergulir dikendalikan oleh konfigurasi penyebaran layanan.

Amazon ECS menggunakan parameter berikut untuk menentukan jumlah tugas:

- `minimumHealthyPercent` mewakili batas bawah pada jumlah tugas yang seharusnya berjalan untuk layanan selama deployment atau saat instans kontainer dikosongkan, sebagai persentase dari jumlah tugas yang diinginkan untuk layanan. Nilai ini dibulatkan ke atas. Sebagai contoh, jika persentase minimum yang sehat adalah `50` dan jumlah tugas yang diinginkan adalah empat, maka

penjadwal dapat menghentikan dua tugas yang sudah ada sebelum memulai dua tugas baru. Demikian juga, jika persentase minimum yang sehat adalah 75% dan jumlah tugas yang diinginkan adalah dua, maka penjadwal tidak dapat menghentikan tugas apa pun karena nilai yang dihasilkan juga dua.

Jika tugas menjadi tidak sehat, penjadwal layanan Amazon ECS akan memulai tugas penggantian terlebih dahulu dan mempertahankan `minimumHealthyPercent` tugas hingga tugas penggantian menjadi sehat. Saat tugas pengganti diluncurkan dan menjadi sehat, tugas yang tidak sehat secara bertahap akan dihentikan.

- `maximumPercent` mewakili batas atas jumlah tugas yang harus berjalan untuk layanan selama deployment atau saat instans kontainer dikosongkan, sebagai persentase dari jumlah tugas yang diinginkan untuk layanan. Nilai ini dibulatkan ke bawah. Misalnya jika persentase maksimum adalah `200` dan jumlah tugas yang diinginkan adalah empat maka penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas yang ada. Demikian juga, jika persentase maksimum adalah `125` dan jumlah tugas yang diinginkan adalah tiga, maka penjadwal tidak dapat memulai tugas apa pun karena nilai yang dihasilkan juga tiga.

> **Important**
>
> Saat menetapkan persen sehat minimum atau persen maksimum, Anda harus memastikan bahwa penjadwal dapat menghentikan atau memulai setidaknya satu tugas saat penerapan dimulai. Jika layanan Anda memiliki deployment yang macet akibat konfigurasi deployment yang tidak valid, maka pesan kejadian layanan akan dikirimkan. Untuk informasi selengkapnya, lihat service (*service-name*) tidak dapat menghentikan atau memulai tugas selama penerapan karena konfigurasi penerapan layanan. Perbarui nilai minimumHealthyPercent atau MaximumPercent dan coba lagi..

Penerapan bergulir memiliki 2 metode yang menyediakan cara untuk mengidentifikasi dengan cepat kapan penerapan layanan gagal:

- the section called “ Bagaimana pemutus sirkuit penyebaran mendeteksi kegagalan”
- the section called “Bagaimana CloudWatch alarm mendeteksi kegagalan penerapan”

Metode ini dapat digunakan secara terpisah atau bersama-sama. Ketika kedua metode digunakan, penerapan disetel ke gagal segera setelah kriteria kegagalan untuk salah satu metode kegagalan terpenuhi.

Gunakan panduan berikut untuk membantu menentukan metode mana yang akan digunakan:

- Pemutus sirkuit - Gunakan metode ini saat Anda ingin menghentikan penerapan saat tugas tidak dapat dimulai.
- CloudWatch alarm - Gunakan metode ini ketika Anda ingin menghentikan penyebaran berdasarkan metrik aplikasi.

Kedua metode mendukung pengguliran kembali ke revisi layanan sebelumnya.

Resolusi gambar kontainer

Secara default, Amazon ECS menyelesaikan tag gambar kontainer yang ditentukan dalam definisi tugas ke intisari gambar kontainer. Jika Anda membuat layanan yang menjalankan dan mempertahankan satu tugas, tugas tersebut akan digunakan untuk membuat intisari gambar untuk kontainer dalam tugas tersebut. Jika Anda membuat layanan yang menjalankan dan memelihara beberapa tugas, tugas pertama yang dimulai oleh penjadwal layanan selama penerapan akan digunakan untuk membuat intisari gambar untuk kontainer dalam tugas.

Jika tiga atau lebih upaya untuk membuat intisari gambar kontainer gagal, penerapan berlanjut tanpa resolusi intisari gambar. Jika pemutus sirkuit penyebaran diaktifkan, penerapan juga gagal dan diputar kembali.

Setelah intisari gambar kontainer dibuat, Amazon ECS menggunakan intisari untuk memulai tugas lain yang diinginkan, dan untuk pembaruan layanan di masa mendatang. Ini mengarah ke semua tugas dalam layanan yang selalu menjalankan gambar kontainer yang identik, menghasilkan konsistensi versi untuk perangkat lunak Anda.

Anda dapat mengonfigurasi perilaku ini untuk setiap kontainer dalam tugas Anda dengan menggunakan `versionConsistency` parameter dalam definisi penampung. Untuk informasi selengkapnya, lihat `versionConsistency`.

> **Note**
>
> - Amazon ECS Agent versi lebih rendah dari `1.31.0` tidak mendukung resolusi intisari gambar. Versi agen `1.31.0` untuk `1.69.0` mendukung resolusi intisari gambar hanya

untuk gambar yang didorong ke repositori Amazon ECR. Versi agen `1.70.0` atau yang lebih tinggi mendukung resolusi intisari gambar untuk semua gambar.

- Versi platform Fargate Linux minimum untuk resolusi intisari gambar adalah. `1.3.0` Versi platform Fargate Windows minimum untuk resolusi intisari gambar adalah. `1.0.0`
- Amazon ECS tidak menangkap intisari kontainer sespan yang dikelola oleh Amazon ECS, seperti GuardDuty agen keamanan Amazon atau proxy Service Connect.
- Untuk mengurangi potensi latensi yang terkait dengan resolusi gambar kontainer dalam layanan dengan beberapa tugas, jalankan versi agen Amazon ECS `1.83.0` atau yang lebih tinggi pada instans EC2 penampung. Untuk menghindari potensi latensi, tentukan intisari gambar kontainer dalam definisi tugas Anda.
- Jika Anda membuat layanan dengan jumlah tugas nol yang diinginkan, Amazon ECS tidak dapat membuat intisari kontainer hingga Anda memicu penerapan layanan lain dengan jumlah tugas yang diinginkan lebih besar dari nol.
- Untuk membuat intisari gambar yang diperbarui, Anda dapat memaksa penerapan baru. Intisari yang diperbarui akan digunakan untuk memulai tugas baru dan tidak akan memengaruhi tugas yang sudah berjalan. Untuk informasi selengkapnya tentang pemaksaan penerapan baru, lihat forceNewDeploymentdi referensi Amazon ECS API.
- Saat menggunakan penyedia EC2 kapasitas, jika kapasitas tidak cukup untuk memulai tugas selama penerapan awal, konsistensi versi perangkat lunak mungkin gagal. Untuk memastikan konsistensi versi dipertahankan bahkan ketika kapasitas terbatas, setel secara eksplisit `versionConsistency: true` dalam konfigurasi kontainer definisi tugas Anda daripada mengandalkan perilaku default. Hal ini menyebabkan Amazon ECS menunggu hingga kapasitas tersedia sebelum melanjutkan penyebaran.

Praktik terbaik untuk parameter layanan Amazon ECS

Untuk memastikan tidak ada downtime aplikasi, proses penerapan adalah sebagai berikut:

1. Mulai wadah aplikasi baru sambil menjaga kontainer yang ada tetap berjalan.
2. Periksa apakah wadah baru itu sehat.
3. Hentikan wadah lama.

Bergantung pada konfigurasi penerapan Anda dan jumlah ruang kosong dan tanpa cadangan di klaster Anda, mungkin diperlukan beberapa putaran untuk menyelesaikannya, ganti semua tugas lama dengan tugas baru.

Ada dua opsi konfigurasi layanan yang dapat Anda gunakan untuk mengubah nomor:

- `minimumHealthyPercent`: 100% (default)

  Batas bawah pada jumlah tugas untuk layanan Anda yang harus tetap dalam `RUNNING` status selama penerapan. Ini adalah persentase dari yang `desiredCount` dibulatkan ke bilangan bulat terdekat. Parameter ini memungkinkan Anda untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan.

- `maximumPercent`: 200% (default)

  Batas atas jumlah tugas untuk layanan Anda yang diizinkan di `RUNNING` atau `PENDING` status selama penerapan. Ini adalah persentase dari yang `desiredCount` dibulatkan ke bawah ke bilangan bulat terdekat.

Contoh: Opsi konfigurasi default

Pertimbangkan layanan berikut yang memiliki enam tugas, digunakan dalam cluster yang memiliki ruang untuk total delapan tugas. Opsi konfigurasi layanan default tidak memungkinkan penerapan berada di bawah 100% dari enam tugas yang diinginkan.

Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti enam tugas.
2. Penjadwal memulai dua tugas baru karena pengaturan default mengharuskan ada enam tugas yang berjalan.

   Sekarang ada enam tugas yang ada dan dua tugas baru.
3. Penjadwal menghentikan dua tugas yang ada.

   Sekarang ada empat tugas yang ada dan dua yang baru.
4. Penjadwal memulai dua tugas baru tambahan.

   Sekarang ada empat tugas yang ada dan empat tugas baru.
5. Penjadwal menutup dua tugas yang ada.

Sekarang ada dua tugas yang ada dan empat yang baru.

6. Penjadwal memulai dua tugas baru tambahan.

   Sekarang ada dua tugas yang ada dan enam tugas baru

7. Penjadwal menutup dua tugas terakhir yang ada.

   Sekarang ada enam tugas baru.

Dalam contoh di atas, jika Anda menggunakan nilai default untuk opsi, ada 2,5 menit menunggu untuk setiap tugas baru yang dimulai. Selain itu, penyeimbang beban mungkin harus menunggu 5 menit agar tugas lama berhenti.

Contoh: Modifikasi **`minimumHealthyPercent`**

Anda dapat mempercepat penerapan dengan menyetel `minimumHealthyPercent` nilainya menjadi 50%.

Pertimbangkan layanan berikut yang memiliki enam tugas, digunakan dalam cluster yang memiliki ruang untuk total delapan tugas. Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti enam tugas.
2. Penjadwal menghentikan tiga tugas yang ada.

   Masih ada tiga tugas yang berjalan yang memenuhi `minimumHealthyPercent` nilai.

3. Penjadwal memulai lima tugas baru.

   Ada tiga tugas tugas yang ada dan lima tugas baru.

4. Penjadwal menghentikan tiga tugas yang tersisa.

   Ada lima tugas baru

5. Penjadwal memulai tugas baru terakhir.

   Ada enam tugas baru.

Contoh: Memodifikasi ruang kosong cluster

Anda juga dapat menambahkan ruang kosong tambahan sehingga Anda dapat menjalankan tugas tambahan.

Pertimbangkan layanan berikut yang memiliki enam tugas, digunakan dalam cluster yang memiliki ruang untuk total sepuluh tugas. Proses penyebaran adalah sebagai berikut:

1. Tujuannya adalah untuk mengganti tugas yang ada.
2. Penjadwal menghentikan tiga tugas yang ada,

   Ada tiga tugas yang ada.
3. Penjadwal memulai enam tugas baru.

   Ada tugas yang ada dan enam tugas baru
4. Penjadwal menghentikan tiga tugas yang ada.

   Ada enam tugas baru.

Rekomendasi

Gunakan nilai berikut untuk opsi konfigurasi layanan saat tugas Anda menganggur selama beberapa waktu dan tidak memiliki tingkat pemanfaatan yang tinggi.

- `minimumHealthyPercent`: 50%
- `maximumPercent`: 200%

Membuat penyebaran pembaruan bergulir Amazon ECS

Buat layanan untuk menjalankan dan mempertahankan sejumlah contoh tertentu dari definisi tugas secara bersamaan dalam sebuah cluster. Jika salah satu tugas Anda gagal atau berhenti, penjadwal layanan Amazon ECS meluncurkan contoh lain dari definisi tugas Anda untuk menggantikannya. Ini membantu mempertahankan jumlah tugas yang Anda inginkan dalam layanan.

Tentukan parameter konfigurasi berikut sebelum Anda membuat layanan:

- Ada dua opsi komputasi yang mendistribusikan tugas Anda.
  - Strategi penyedia kapasitas menyebabkan Amazon ECS mendistribusikan tugas Anda di satu atau di beberapa penyedia kapasitas.
  - Jenis peluncuran menyebabkan Amazon ECS meluncurkan tugas kami secara langsung di Fargate atau pada instans EC2 yang terdaftar di cluster Anda.
- Penentuan tugas yang menggunakan mode atau layanan jaringan `awsvpc` yang dikonfigurasi untuk menggunakan penyeimbang beban harus memiliki konfigurasi jaringan. Secara default,

konsol memilih VPC Amazon default bersama dengan semua subnet dan grup keamanan default dalam VPC Amazon default.

- Strategi penempatan, Strategi penempatan tugas default mendistribusikan tugas secara merata di seluruh Availability Zone.

  Kami menyarankan Anda menggunakan penyeimbangan kembali Availability Zone untuk membantu memastikan ketersediaan yang tinggi untuk layanan Anda. Untuk informasi selengkapnya, lihat Menyeimbangkan layanan Amazon ECS di seluruh Availability Zone.
- Saat Anda menggunakan Jenis Peluncuran untuk penyebaran layanan Anda, secara default layanan dimulai di subnet di VPC klaster Anda.
- Untuk strategi penyedia kapasitas, konsol memilih opsi komputasi secara default. Berikut dijelaskan tentang urutan yang digunakan konsol untuk memilih default:
  - Jika klaster Anda memiliki strategi penyedia kapasitas default yang ditentukan, klaster akan dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki penyedia kapasitas Fargate yang ditambahkan ke klaster, strategi penyedia kapasitas khusus yang menggunakan penyedia `FARGATE` kapasitas dipilih.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan tetapi Anda memiliki satu atau beberapa penyedia kapasitas grup Auto Scaling yang ditambahkan ke cluster, opsi Use custom (Advanced) akan dipilih dan Anda perlu menentukan strategi secara manual.
  - Jika klaster Anda tidak memiliki strategi penyedia kapasitas default yang ditentukan dan tidak ada penyedia kapasitas yang ditambahkan ke cluster, jenis peluncuran Fargate akan dipilih.
- Opsi default deteksi kegagalan penerapan default adalah menggunakan opsi pemutus sirkuit penyebaran Amazon ECS dengan opsi Rollback on failure.

  Untuk informasi selengkapnya, lihat Bagaimana pemutus sirkuit penyebaran Amazon ECS mendeteksi kegagalan.
- Jika Anda ingin menggunakan opsi blue/green penyebaran, tentukan caranya CodeDeploy memindahkan aplikasi. Pilihan berikut tersedia:
  - CodeDeployDefault. ECSAllAtOnce: Menggeser semua lalu lintas ke wadah Amazon ECS yang diperbarui sekaligus
  - CodeDeployDefault. ECSLinear10 PercentEvery 1Menit: Menggeser 10 persen lalu lintas setiap menit sampai semua lalu lintas bergeser.
  - CodeDeployDefault. ECSLinear10 PercentEvery 3Menit: Menggeser 10 persen lalu lintas setiap 3 menit sampai semua lalu lintas bergeser.

  - CodeDeployDefault. ECSCanary10Percent5Minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisanya 90 persen dikerahkan lima menit kemudian.
  - CodeDeployDefault. ECSCanary10Percent15Minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisa 90 persen di-deploy 15 menit kemudian.
- Putuskan apakah Anda ingin Amazon ECS menambah atau mengurangi jumlah tugas yang diinginkan dalam layanan Anda secara otomatis. Untuk informasi lihat,Secara otomatis menskalakan layanan Amazon ECS Anda.
- Jika Anda memerlukan aplikasi untuk terhubung ke aplikasi lain yang berjalan di Amazon ECS, tentukan opsi yang sesuai dengan arsitektur Anda. Untuk informasi selengkapnya, lihat Interkoneksi layanan Amazon ECS.
- Saat Anda membuat layanan yang menggunakan pemutus sirkuit Amazon ECS, Amazon ECS membuat penyebaran layanan dan revisi layanan. Sumber daya ini memungkinkan Anda untuk melihat informasi terperinci tentang riwayat layanan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

  Untuk informasi tentang cara membuat layanan menggunakan AWS CLI, lihat create-servicedi AWS Command Line Interface Referensi.

  Untuk informasi tentang cara membuat layanan menggunakan AWS CloudFormation, lihat AWS::ECS::Servicedi Panduan AWS CloudFormation Pengguna.

Buat layanan dengan opsi default

Anda dapat menggunakan konsol untuk membuat dan menyebarkan layanan dengan cepat. Layanan ini memiliki konfigurasi berikut:

- Menyebarkan di VPC dan subnet yang terkait dengan cluster Anda
- Menyebarkan satu tugas
- Menggunakan penerapan bergulir
- Menggunakan strategi penyedia kapasitas dengan penyedia kapasitas default Anda
- Menggunakan pemutus sirkuit penyebaran untuk mendeteksi kegagalan dan menetapkan opsi untuk secara otomatis memutar kembali penerapan pada kegagalan

Untuk menerapkan layanan menggunakan parameter default ikuti langkah-langkah ini.

Untuk membuat layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di halaman navigasi, pilih Cluster.
3. Pada halaman Clusters, pilih cluster untuk membuat layanan di.
4. Dari tab Layanan, pilih Buat.

   Halaman Create Service akan muncul.
5. Di bawah rincian Layanan, lakukan hal berikut:

   a. Untuk definisi Tugas, masukkan keluarga definisi tugas dan revisi yang akan digunakan.

   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
6. Di bawah konfigurasi Deployment, lakukan hal berikut:

   - Untuk tugas yang diinginkan, masukkan jumlah tugas untuk diluncurkan dan dipelihara dalam layanan.
7. (Opsional) Untuk membantu mengidentifikasi layanan dan tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

   Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Definisi tugas.

   Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag layanan, pilih Aktifkan tag terkelola Amazon ECS, lalu pilih Layanan.

   Menambah atau menghapus tanda.

   - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
     - Untuk Kunci, masukkan nama kunci.
     - Untuk Nilai, masukkan nilai kunci.
   - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.

Buat layanan menggunakan parameter yang ditentukan

Untuk membuat layanan dengan menggunakan parameter yang ditentukan, ikuti langkah-langkah ini.

Untuk membuat layanan (konsol Amazon ECS)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Tentukan sumber daya dari tempat Anda meluncurkan layanan.

Untuk memulai layanan dari	Langkah-langkah
Klaster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. b. Dari tab Layanan, pilih Buat.
Ketentuan tugas	a. Pada halaman Definisi tugas, pilih tombol opsi di sebelah definisi tugas. b. Pada menu Deploy, pilih Buat layanan.

   Halaman Create Service akan muncul.
3. Di bawah rincian Layanan, lakukan hal berikut:
   a. Untuk definisi Tugas, masukkan definisi tugas yang akan digunakan. Kemudian, untuk Revisi, pilih revisi yang akan digunakan.
   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.
4. Untuk klaster yang ada, pilih klaster.

   Pilih Buat klaster untuk menjalankan tugas di klaster baru
5. Pilih bagaimana tugas Anda didistribusikan di seluruh infrastruktur klaster Anda. Di bawah Konfigurasi komputasi, pilih opsi Anda.

Opsi komputasi	Langkah-langkah
Strategi penyedia kapasitas	a. Di bawah Opsi komputasi , pilih Strategi penyedia kapasitas. b. Pilih strategi: • Untuk menggunakan strategi penyedia kapasitas default cluster, pilih Use cluster default. • Jika klaster Anda tidak memiliki strategi penyedia kapasitas default, atau menggunakan strategi kustom, pilih Gunakan strategi penyedia kustom, Tambahkan kapasitas, lalu tentukan strategi penyedia kapasitas khusus Anda dengan menentukan Basis, penyedia Kapasitas, dan Berat. **Note** Untuk menggunakan penyedia kapasitas dalam suatu strategi, penyedia kapasitas

Opsi komputasi	Langkah-langkah
	harus dikaitkan dengan cluster.
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. (Opsional) Ketika tipe peluncuran Fargate diwajibkan, untuk Versi platform, tentukan versi platform yang akan digunakan. Jika versi platform tidak ditentuka n, versi LATEST platform akan digunakan.

6. Di bawah konfigurasi Deployment, lakukan hal berikut:

   a. Untuk jenis Layanan, pilih strategi penjadwalan layanan.

      - Agar penjadwal menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas, pilih Daemon.
      - Untuk menempatkan penjadwal dan mempertahankan jumlah tugas yang diinginkan di cluster Anda, pilih Replika.

   b. Jika Anda memilih Replika, untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan dan dipelihara dalam layanan.

   c. Jika Anda memilih Replika, agar Amazon ECS memantau distribusi tugas di seluruh Availability Zone, dan mendistribusikannya kembali ketika ada ketidakseimbangan, di bawah penyeimbangan kembali layanan Availability Zone, pilih Availability Zone service rebalancing.

   d. Untuk Health check masa tenggang, masukkan jumlah waktu (dalam detik) yang memasukkan jumlah waktu (dalam detik) bahwa penjadwal layanan mengabaikan Elastic Load Balancing yang tidak sehat, VPC Lattice, dan pemeriksaan kesehatan kontainer

setelah tugas pertama kali dimulai. Jika Anda tidak menentukan nilai tenggang pemeriksaan kesehatan, nilai default 0 digunakan.

e. Tentukan jenis penerapan untuk layanan Anda. Perluas opsi Deployment, lalu tentukan parameter berikut.

Jenis deployment	Langkah-langkah	
Pembaruan bergulir	a. Untuk tugas yang menjalankan Min, masukkan batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Untuk informasi selengkap nya, lihat Konfigurasi penerapan. b. Untuk tugas yang berjalan Max, masukkan batas atas jumlah tugas dalam layanan yang diizinkan dalam `PENDING` status `RUNNING` atau selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat).	

Jenis deployment	Langkah-langkah	
Deployment blue/green	a. Untuk konfigurasi Deployment, pilih cara CodeDeploy merutekan lalu lintas produksi ke tugas pengganti yang ditetapkan selama penerapan. b. Untuk peran Service CodeDeploy, pilih peran IAM yang digunakan layanan untuk membuat permintaan API ke otorisasi Layanan AWS.	

f. Untuk mengonfigurasi cara Amazon ECS mendeteksi dan menangani kegagalan penerapan, perluas deteksi kegagalan Deployment, lalu pilih opsi Anda.

   i. Untuk menghentikan penerapan saat tugas tidak dapat dimulai, pilih Gunakan pemutus sirkuit penyebaran Amazon ECS.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat pemutus sirkuit penyebaran mengatur penerapan ke status gagal, pilih Rollback on failure.

   ii. Untuk menghentikan penerapan berdasarkan metrik aplikasi, pilih Gunakan CloudWatch alarm. Kemudian, dari nama CloudWatch alarm, pilih alarm. Untuk membuat alarm baru, buka CloudWatch konsol.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat CloudWatch alarm menyetel penerapan ke status gagal, pilih Rollback on failure.

7. Jika definisi tugas Anda menggunakan mode `awsvpc` jaringan, Anda dapat menentukan konfigurasi jaringan kustom memperluas Jaringan, dan kemudian melakukan hal berikut untuk.

   a. Untuk VPC, pilih VPC yang akan digunakan.

b. Untuk Subnet, pilih satu atau beberapa subnet di VPC yang dipertimbangkan oleh penjadwal tugas saat menempatkan tugas Anda.

c. Untuk Grup keamanan, Anda dapat memilih grup keamanan yang sudah ada atau menciptakan grup keamanan baru. Untuk menggunakan grup keamanan yang sudah ada, pilih grup keamanan dan lanjutkan ke langkah selanjutnya. Untuk membuat grup keamanan baru, pilih Buat grup keamanan baru. Anda harus menentukan nama grup keamanan, deskripsi, dan kemudian tambahkan satu atau beberapa aturan masuk untuk grup keamanan.

d. Untuk IP Publik, pilih apakah akan menetapkan alamat IP publik secara otomatis ke antarmuka jaringan elastis (ENI) tugas.

   AWS Fargate tugas dapat diberikan alamat IP publik ketika dijalankan di subnet publik sehingga mereka memiliki rute ke internet. EC2 tugas tidak dapat diberikan IP publik menggunakan bidang ini. Untuk informasi selengkapnya, lihat opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate dan Alokasikan antarmuka jaringan untuk tugas Amazon ECS.

8. (Opsional) Untuk menghubungkan layanan Anda menggunakan Service Connect, perluas Service Connect, lalu tentukan yang berikut ini:

   a. Pilih Aktifkan Service Connect.

   b. Di bawah konfigurasi Service Connect, tentukan mode klien.

      - Jika layanan Anda menjalankan aplikasi klien jaringan yang hanya perlu terhubung ke layanan lain di namespace, pilih sisi Klien saja.
      - Jika layanan Anda menjalankan aplikasi jaringan atau layanan web dan perlu menyediakan titik akhir untuk layanan ini, dan terhubung ke layanan lain di namespace, pilih Klien dan server.

   c. Untuk menggunakan namespace yang bukan namespace cluster default, untuk Namespace, pilih namespace layanan.

   d. (Opsional) Tentukan konfigurasi log. Pilih Gunakan koleksi log. Opsi default mengirimkan log kontainer ke CloudWatch Log. Opsi driver log lainnya dikonfigurasi menggunakan AWS FireLens. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

      Berikut ini menjelaskan setiap tujuan log kontainer secara lebih rinci.

- Amazon CloudWatch — Konfigurasikan tugas untuk mengirim log kontainer ke CloudWatch Log. Opsi driver log default disediakan, yang membuat grup CloudWatch log atas nama Anda. Untuk menentukan nama grup log yang berbeda, ubah nilai opsi driver.
- Amazon Data Firehose — Konfigurasikan tugas untuk mengirim log kontainer ke Firehose. Opsi driver log default disediakan, yang mengirim log ke aliran pengiriman Firehose. Untuk menentukan nama aliran pengiriman yang berbeda, ubah nilai opsi driver.
- Amazon Kinesis Data Streams — Konfigurasikan tugas untuk mengirim log kontainer ke Kinesis Data Streams. Opsi driver log default disediakan, yang mengirim log ke aliran Kinesis Data Streams. Untuk menentukan nama aliran yang berbeda, ubah nilai opsi driver.
- Amazon OpenSearch Service - Konfigurasikan tugas untuk mengirim log kontainer ke domain OpenSearch Layanan. Opsi driver log harus disediakan.
- Amazon S3 — Konfigurasikan tugas untuk mengirim log kontainer ke bucket Amazon S3. Opsi driver log default disediakan, tetapi Anda harus menentukan nama bucket Amazon S3 yang valid.

9. (Opsional) Untuk menghubungkan layanan Anda menggunakan Service Discovery, perluas penemuan Layanan, lalu lakukan hal berikut.

   a. Pilih Gunakan penemuan layanan.

   b. Untuk menggunakan namespace baru, pilih Buat namespace baru di bawah Konfigurasi namespace, lalu berikan nama dan deskripsi namespace. Untuk menggunakan namespace yang ada, pilih Pilih namespace yang ada lalu pilih namespace yang ingin Anda gunakan.

   c. Memberikan informasi layanan Service Discovery seperti nama dan deskripsi layanan.

   d. Agar Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer secara berkala, pilih Aktifkan propagasi kesehatan tugas Amazon ECS.

   e. Untuk jenis rekaman DNS, pilih jenis rekaman DNS yang akan dibuat untuk layanan Anda. Penemuan layanan Amazon ECS hanya mendukung catatan A dan SRV, tergantung pada mode jaringan yang ditentukan oleh definisi tugas Anda. Untuk informasi selengkapnya tentang jenis rekaman ini, lihat Jenis Rekaman DNS yang Didukung di Panduan Pengembang Amazon Route 53.

      - Jika definisi tugas yang ditentukan oleh tugas layanan Anda menggunakan mode `host` jaringan `bridge` atau, hanya jenis catatan SRV yang didukung. Pilih nama kontainer dan kombinasi port untuk dikaitkan dengan catatan.

- Jika definisi tugas yang ditentukan oleh tugas layanan Anda menggunakan mode `awsvpc` jaringan, pilih jenis catatan A atau SRV. Jika Anda memilih A, lewati ke langkah berikutnya. Jika Anda memilih SRV, tentukan port tempat layanan dapat ditemukan atau nama kontainer dan kombinasi port untuk dikaitkan dengan catatan.

Untuk TTL, masukkan waktu dalam hitungan detik berapa lama set rekaman di-cache oleh resolver DNS dan oleh browser web.

10. (Opsional) Untuk menghubungkan layanan Anda menggunakan VPC Lattice, xxpand VPC Lattice, lalu lakukan hal berikut:

    a. Pilih Aktifkan Kisi VPC

    b. Untuk peran Infrastruktur, pilih peran infrastruktur.

    Jika Anda belum membuat peran, pilih Buat peran infrastruktur.

    c. Di bawah Grup Target pilih grup target atau grup. Anda harus memilih setidaknya satu kelompok sasaran dan dapat memiliki maksimal lima. Pilih Tambahkan grup target untuk menambahkan grup target tambahan. Pilih nama Port, Protokol, dan Port untuk setiap grup target yang Anda pilih.

    Untuk menghapus grup target, pilih Hapus.

    > **Note**
    >
    > - Jika Anda ingin menambahkan grup target yang ada, Anda perlu menggunakan AWS CLI. Untuk petunjuk tentang cara menambahkan grup target menggunakan AWS CLI, lihat daftar target di Referensi. AWS Command Line Interface
    > - Meskipun layanan VPC Lattice dapat memiliki beberapa grup target, setiap grup target hanya dapat ditambahkan ke satu layanan.

    d. Untuk menyelesaikan konfigurasi VPC Lattice, dengan menyertakan grup target baru Anda dalam tindakan default listener atau dalam aturan layanan VPC Lattice yang ada di konsol VPC Lattice. Untuk informasi selengkapnya, lihat Aturan pendengar untuk layanan Kisi VPC Anda.

11. (Opsional) Untuk mengonfigurasi penyeimbang beban untuk layanan Anda, perluas Load balancing.

Pilih penyeimbang beban.

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
Penyeimbang Beban Aplikasi	a. Untuk tipe Load balancer, pilih Application Load Balancer. b. Pilih Buat penyeimbang beban baru untuk membuat Application Load Balancer baru atau Gunakan penyeimbang beban yang ada untuk memilih Application Load Balancer yang ada. c. Untuk nama Load balancer, masukkan nama unik. d. Untuk Pilih kontainer untuk memuat keseimbangan, pilih kontainer yang menampung layanan. e. Untuk Listener, masukkan port dan protokol untuk Application Load Balancer untuk mendengarkan permintaan koneksi. Secara default, penyeimbang beban akan dikonfigurasi untuk menggunakan port 80 dan HTTP. f. Untuk nama grup Target, masukkan nama dan protokol untuk grup	

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
	target yang diminta oleh Application Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda. g. Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. h. Untuk jalur pemeriksaan Kesehatan, masukkan jalur yang ada di dalam wadah Anda di mana Application Load Balancer secara berkala mengirimkan permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (`/`).	

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
Penyeimbang Beban Jaringan	a. Untuk jenis Load balancer, pilih Network Load Balancer. b. Untuk Load Balancer, pilih Network Load Balancer yang sudah ada. c. Untuk Pilih kontainer untuk memuat keseimban gan, pilih kontainer yang menampung layanan. d. Untuk nama grup Target, masukkan nama dan protokol untuk grup target yang diminta oleh Network Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda. e. Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. f. Untuk jalur pemeriksaan Kesehatan, masukkan jalur yang ada di dalam wadah Anda di mana Network Load Balancer	

Untuk menggunakan penyeimbang beban ini	Lakukan hal berikut	
	secara berkala mengirimkan permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (/).	

12. (Opsional) Untuk mengonfigurasi Auto Scaling layanan, perluas Service auto scaling, lalu tentukan parameter berikut.Untuk menggunakan predicte auto scaling, yang melihat data beban masa lalu dari arus lalu lintas, konfigurasikan setelah Anda membuat layanan. Untuk informasi selengkapnya, lihat Gunakan pola historis untuk menskalakan layanan Amazon ECS dengan penskalaan prediktif.

    a. Untuk menggunakan penskalaan otomatis servis, pilih Penskalaan otomatis layanan.

    b. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

    c. Untuk Jumlah tugas maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

    d. Pilih jenis kebijakan. Di bawah Jenis kebijakan penskalaan, pilih salah satu opsi berikut.

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
Pelacakan target	a. Untuk Jenis kebijakan penskalaan, pilih Pelacakan target. b. Untuk nama Kebijakan, masukkan nama kebijakan.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	c. Untuk metrik layanan ECS, pilih salah satu metrik berikut. • ECSServiceRata-rata CPUUtilization - Pemanfaatan CPU rata-rata layanan. • ECSServiceAverageMemoryUtilization— Pemanfaatan memori rata-rata dari layanan. • ALBRequestCountPerTarget— Jumlah permintaan yang diselesaikan per target dalam kelompok sasaran Application Load Balancer. d. Untuk nilai Target, masukkan nilai yang dipertahankan layanan untuk metrik yang dipilih. e. Untuk periode cooldown scale-out, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-out (tambahkan tugas) yang harus dilewati sebelum aktivitas scale-out lainnya dapat dimulai.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	f. Untuk periode cooldown Scale-in, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-in (hapus tugas) yang harus lulus sebelum aktivitas scale-in lainnya dapat dimulai. g. Untuk mencegah kebijakan melakukan aktivitas penskalaan, pilih Matikan penskalaan. h. • (Opsional) Pilih Matikan penskalaan jika Anda ingin kebijakan penskalaan Anda ditingkatkan untuk meningkatkan lalu lintas tetapi tidak membutuhkannya untuk menskalakan saat lalu lintas menurun.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
Penskalaan langkah	a. Untuk Jenis kebijakan penskalaan, pilih Penskalaan langkah. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk Nama alarm, masukkan nama yang unik untuk alarm. d. Untuk metrik layanan Amazon ECS, pilih metrik yang akan digunakan untuk alarm. e. Untuk Statistik, pilih statistik alarm. f. Untuk Periode, pilih periode untuk alarm. g. Untuk kondisi Alarm, pilih cara membandin gkan metrik yang dipilih dengan ambang batas yang ditentukan. h. Untuk Ambang untuk membandingkan metrik dan periode Evaluasi untuk memulai alarm, masukkan ambang batas yang digunakan untuk alarm dan berapa lama untuk mengevaluasi ambang batas.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	i. Di bawah Tindakan penskalaan, lakukan hal berikut: • Untuk Tindakan, pilih apakah akan menambah, menghapus, atau menetapkan jumlah tertentu yang diinginkan untuk layanan Anda. • Jika Anda memilih untuk menambah atau menghapus tugas, untuk Nilai, masukkan jumlah tugas (atau persen tugas yang ada) untuk ditambahkan atau dihapus saat tindakan penskalaan dimulai. Jika Anda memilih untuk mengatur hitungan yang diinginkan, masukkan jumlah tugas. Untuk Jenis, pilih apakah Nilai adalah bilangan bulat atau nilai persen dari jumlah yang diinginkan yang ada. • Untuk batas Bawah dan Batas atas, masukkan batas	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	bawah dan batas atas penyesuai an penskalaan langkah Anda. Secara default, batas bawah untuk kebijakan penambahan adalah ambang batas alarm, sedangkan batas atas adalah positif (+) tak terbatas. Secara default, batas atas untuk kebijakan penghapusan adalah ambang batas alarm, sedangkan batas bawah adalah negatif (-) tak terbatas. • (Opsional) Tambahkan opsi penskalaan tambahan. Pilih Tambahkan tindakan penskalaan baru, lalu ulangi langkah tindakan Penskalaan. • Untuk periode Cooldown, masukkan jumlah waktu, dalam detik, untuk menunggu aktivitas penskalaan sebelumnya diterapka n. Untuk kebijakan add, ini adalah waktu	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	setelah aktivitas scale-out yang kebijakan penskalaan memblokir aktivitas scale-in dan membatasi berapa banyak tugas yang dapat diskalakan pada suatu waktu. Untuk kebijakan penghapusan, ini adalah waktu setelah aktivitas penskalaan yang harus diteruskan sebelum aktivitas penskalaan lainnya dapat dimulai.	

13. (Opsional) Untuk menggunakan strategi penempatan tugas selain default, perluas Penempatan Tugas, lalu pilih dari opsi berikut.

    Untuk informasi selengkapnya, lihat Bagaimana Amazon ECS menempatkan tugas pada instans kontainer.

    - AZ Balanced Spread — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instance container di Availability Zone.
    - AZ Balanced BinPack — Mendistribusikan tugas di seluruh Availability Zone dan di seluruh instans kontainer dengan memori yang paling sedikit tersedia.
    - BinPack— Mendistribusikan tugas berdasarkan jumlah CPU atau memori yang paling sedikit tersedia.
    - Satu Tugas Per Host - Tempatkan, paling banyak, satu tugas dari layanan pada setiap instance kontainer.
    - Kustom - Tentukan strategi penempatan tugas Anda sendiri.

Jika Anda memilih Kustom, tentukan algoritme untuk menempatkan tugas dan aturan yang dipertimbangkan selama penempatan tugas.

- Di bawah Strategi, untuk Jenis dan Bidang, pilih algoritma dan entitas yang akan digunakan untuk algoritme.

  Anda dapat memasukkan maksimal 5 strategi.
- Di bawah Constraint, untuk Type dan Expression, pilih aturan dan atribut untuk kendala.

  Misalnya, untuk menyetel batasan untuk menempatkan tugas pada instance T2, untuk Ekspresi, masukkan atribut:ecs.instance-type =~ t2. *.

  Anda dapat memasukkan maksimal 10 kendala.

14. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

    Nama volume dan jenis volume dikonfigurasi saat Anda membuat revisi definisi tugas dan tidak dapat diubah saat membuat layanan. Untuk memperbarui nama dan jenis volume, Anda harus membuat revisi definisi tugas baru dan membuat layanan dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda. b. Untuk Ukuran (GiB), masukkan nilai yang valid untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentuka n minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah input/output operasi maksimum (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk`io1`,`io2`, dan jenis `gp3` volume. d. Untuk Throughput (MIB/s), masukkan throughput yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis `gp3` volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat volume baru yang kosong dengan tidak memilih atau memasukkan ID snapshot. f. Jika Anda menentukan ID Snapshot, Anda dapat menentukan tingkat inisialisasi Volume (MIB/s). Masukkan nilai antara	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	100 dan 300, dalam MIB/s, yang akan menentukan seberapa cepat data dimuat dari snapshot yang ditentukan menggunakan ID Snapshot untuk pembuatan volume. g. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. h. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas. Anda dapat	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	melampirkan kebijakan `AmazonECSInfrastructureRole PolicyForVolumes` terkelola ke peran, atau Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang diperlukan, lihat Peran IAM infrastruktur Amazon ECS. i. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigurasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan enkripsi default Amazon	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	EBS tidak diaktifkan, volume tidak akan dienkripsi. Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume akan tidak dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat volume dari snapshot terenkripsi. j. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentukan AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam kebijakan Anda. AWS	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	KMS key Untuk informasi selengkapnya, lihat Enkripsi data untuk volume Amazon EBS. k. (Opsional) Di bawah Tag, Anda dapat menambahk an tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau layanan, atau dengan menyediakan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih Definisi tugas untuk menyebarkan tag dari. Jika Anda ingin menyebarkan tag dari layanan, pilih Layanan untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	setiap tag yang Anda tambahkan. Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

15. (Opsional) Untuk membantu mengidentifikasi layanan dan tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Definisi tugas.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag layanan, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Layanan.

    Menambah atau menghapus tanda.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.
16. Pilih Create (Buat).

Langkah selanjutnya

Berikut ini adalah tindakan tambahan setelah Anda membuat layanan.

- Konfigurasikan penskalaan auto predicte, yang melihat data pemuatan sebelumnya dari arus lalu lintas. Untuk informasi selengkapnya, lihat Gunakan pola historis untuk menskalakan layanan Amazon ECS dengan penskalaan prediktif.

- Lacak penyebaran Anda dan lihat riwayat layanan Anda untuk layanan yang merupakan pemutus sirkuit Amazon ECS. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

Penerapan Amazon ECS blue/green

blue/green Penyebaran adalah metodologi rilis yang mengurangi waktu henti dan risiko dengan menjalankan dua lingkungan produksi identik yang disebut biru dan hijau. Dengan blue/green penerapan Amazon ECS, Anda dapat memvalidasi revisi layanan baru sebelum mengarahkan lalu lintas produksi kepada mereka. Pendekatan ini memberikan cara yang lebih aman untuk menerapkan perubahan dengan kemampuan untuk memutar kembali dengan cepat jika diperlukan.

Manfaat

Berikut ini adalah manfaat menggunakan blue/green penerapan:

- Mengurangi risiko melalui pengujian dengan lalu lintas produksi sebelum beralih produksi. Anda dapat memvalidasi penerapan baru dengan lalu lintas uji sebelum mengarahkan lalu lintas produksi ke sana.
- Nol penerapan downtime. Lingkungan produksi tetap tersedia selama proses penyebaran, memastikan ketersediaan layanan berkelanjutan.
- Rollback mudah jika masalah terdeteksi. Jika masalah muncul dengan penerapan hijau, Anda dapat dengan cepat kembali ke penerapan biru tanpa gangguan layanan yang diperpanjang.
- Lingkungan pengujian yang terkendali. Lingkungan hijau menyediakan ruang terisolasi untuk menguji fitur baru dengan pola lalu lintas nyata sebelum penyebaran penuh.
- Proses penyebaran yang dapat diprediksi. Pendekatan terstruktur dengan tahapan siklus hidup yang ditentukan membuat penerapan lebih konsisten dan andal.
- Validasi otomatis melalui kait siklus hidup. Anda dapat menerapkan pengujian otomatis pada berbagai tahap penerapan untuk memverifikasi fungsionalitas.

Terminologi

Berikut ini adalah ketentuan blue/green penerapan Amazon ECS:

- Waktu panggang - Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
- Blue deployment - Revisi layanan produksi saat ini yang ingin Anda ganti.

- Penyebaran hijau - Revisi layanan baru yang ingin Anda terapkan.
- Tahap siklus hidup - Serangkaian peristiwa dalam operasi penyebaran, seperti “setelah pergeseran lalu lintas produksi”.
- Lifecycle hook - Fungsi Lambda yang memverifikasi penerapan pada tahap siklus hidup tertentu.
- Listener - Sumber daya Elastic Load Balancing yang memeriksa permintaan koneksi menggunakan protokol dan port yang Anda konfigurasikan. Aturan yang Anda tetapkan untuk pendengar menentukan cara Amazon ECS merutekan permintaan ke target terdaftarnya.
- Aturan - Sumber daya Elastic Load Balancing yang terkait dengan pendengar. Aturan mendefinisikan bagaimana permintaan dirutekan dan terdiri dari tindakan, kondisi, dan prioritas.
- Kelompok sasaran - Sumber daya Elastic Load Balancing yang digunakan untuk merutekan permintaan ke satu atau lebih target terdaftar (misalnya, EC2 instance). Bila Anda membuat pendengar, Anda menentukan grup target untuk tindakan default-nya. Lalu lintas diteruskan ke grup target yang ditentukan dalam aturan pendengar.
- Pergeseran lalu lintas - Proses yang digunakan Amazon ECS untuk mengalihkan lalu lintas dari penyebaran biru ke penyebaran hijau. Untuk blue/green penyebaran Amazon ECS, semua lalu lintas dialihkan dari layanan biru ke layanan hijau sekaligus.

Pertimbangan

Pertimbangkan hal berikut saat memilih jenis penerapan:

- Penggunaan sumber daya: Blue/green penerapan sementara menjalankan revisi layanan biru dan hijau secara bersamaan, yang dapat menggandakan penggunaan sumber daya Anda selama penerapan.
- Pemantauan penyebaran: Blue/green penerapan memberikan informasi status penyebaran yang lebih rinci, memungkinkan Anda memantau setiap tahap proses penyebaran.
- Rollback: Blue/green penerapan memudahkan untuk memutar kembali ke versi sebelumnya jika masalah terdeteksi, karena revisi biru terus berjalan hingga waktu pemanggangan berakhir.

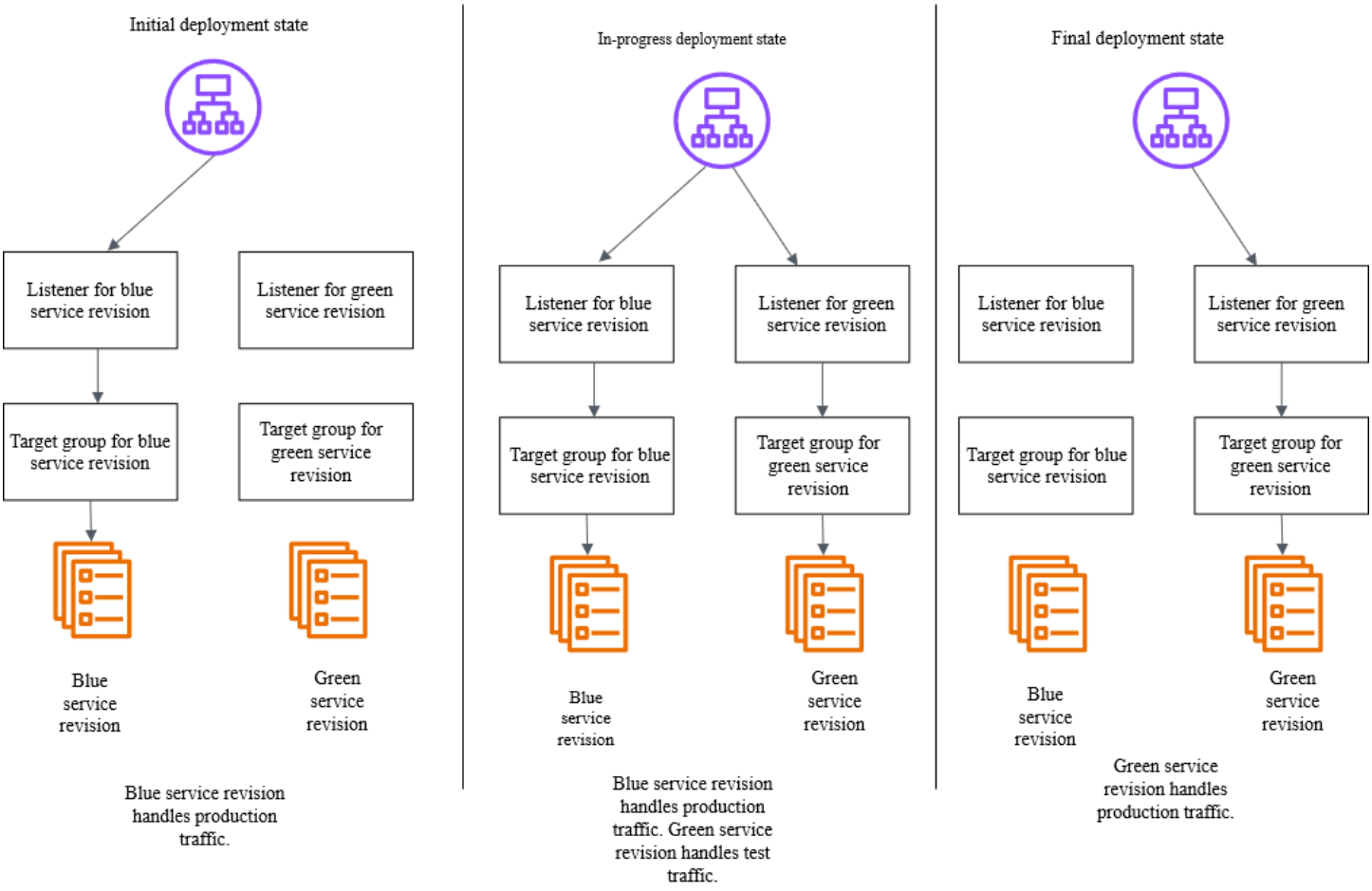
Alur kerja penerapan blue/green layanan Amazon ECS

Proses blue/green penyebaran Amazon ECS mengikuti pendekatan terstruktur dengan enam fase berbeda yang memastikan pembaruan aplikasi yang aman dan andal. Setiap fase melayani tujuan tertentu dalam memvalidasi dan mentransisikan aplikasi Anda dari versi saat ini (biru) ke versi baru (hijau).

1. Tahap Persiapan: Ciptakan lingkungan hijau di samping lingkungan biru yang ada. Ini termasuk penyediaan revisi layanan baru, dan menyiapkan kelompok sasaran.
2. Fase Deployment: Menyebarkan revisi layanan baru ke lingkungan hijau. Amazon ECS meluncurkan tugas baru menggunakan revisi layanan yang diperbarui sementara lingkungan biru terus melayani lalu lintas produksi.
3. Fase Pengujian: Validasi lingkungan hijau menggunakan perutean lalu lintas uji. Application Load Balancer mengarahkan permintaan pengujian ke lingkungan hijau sementara lalu lintas produksi tetap biru.
4. Fase Pergeseran Lalu Lintas: Menggeser lalu lintas produksi dari biru ke hijau berdasarkan strategi penerapan yang dikonfigurasi. Fase ini mencakup pos pemeriksaan pemantauan dan validasi.
5. Fase Pemantauan: Pantau kesehatan aplikasi, metrik kinerja, dan status alarm selama periode waktu pemanggangan. Operasi rollback dimulai ketika masalah terdeteksi.
6. Fase Penyelesaian: Selesaikan penerapan dengan menghentikan lingkungan biru atau mempertahankannya untuk skenario rollback potensial, tergantung pada konfigurasi Anda.

Alur kerja

Diagram berikut mengilustrasikan alur kerja blue/green penerapan komprehensif, yang menunjukkan interaksi antara Amazon ECS, dan Application Load Balancer:



Alur kerja penerapan yang disempurnakan mencakup langkah-langkah rinci berikut:

1. Keadaan Awal: Layanan biru (produksi saat ini) menangani 100% lalu lintas produksi. Application Load Balancer memiliki satu pendengar dengan aturan yang merutekan semua permintaan ke grup target biru yang berisi tugas biru sehat.
2. Penyediaan Lingkungan Hijau: Amazon ECS membuat tugas baru menggunakan definisi tugas yang diperbarui. Tugas-tugas ini terdaftar dengan kelompok target hijau baru tetapi tidak menerima lalu lintas pada awalnya.
3. Validasi Pemeriksaan Kesehatan: Application Load Balancer melakukan pemeriksaan kesehatan pada tugas-tugas hijau. Hanya ketika tugas hijau lulus pemeriksaan kesehatan barulah penerapan dilanjutkan ke fase berikutnya.
4. Uji Perutean Lalu Lintas: Jika dikonfigurasi, aturan pendengar Application Load Balancer merutekan pola lalu lintas tertentu (seperti permintaan dengan header pengujian) ke lingkungan hijau untuk validasi sementara lalu lintas produksi tetap berwarna biru. Ini dikendalikan oleh pendengar yang sama yang menangani lalu lintas produksi, menggunakan aturan berbeda berdasarkan atribut permintaan.

5. Pergeseran Lalu Lintas Produksi: Berdasarkan konfigurasi penyebaran, lalu lintas bergeser dari biru ke hijau. Dalam blue/green penyebaran ECS, ini adalah pergeseran langsung (all-at-once) di mana 100% lalu lintas dipindahkan dari lingkungan biru ke hijau. Application Load Balancer menggunakan pendengar tunggal dengan aturan pendengar yang mengontrol distribusi lalu lintas antara grup target biru dan hijau berdasarkan bobot.
6. Pemantauan dan Validasi: Sepanjang pergeseran lalu lintas, Amazon ECS memantau CloudWatch metrik, status alarm, dan kesehatan penyebaran. Pemicu rollback otomatis diaktifkan jika masalah terdeteksi.
7. Periode Waktu Panggang: Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
8. Pengakhiran Lingkungan Biru: Setelah pergeseran lalu lintas dan validasi berhasil, lingkungan biru dihentikan untuk membebaskan sumber daya cluster, atau dipertahankan untuk kemampuan rollback yang cepat.
9. Keadaan Akhir: Lingkungan hijau menjadi lingkungan produksi baru, menangani 100% lalu lintas. Penerapan ditandai sebagai berhasil.

Tahapan siklus hidup penerapan

Proses blue/green penyebaran berlangsung melalui tahapan siklus hidup yang berbeda (serangkaian peristiwa dalam operasi penyebaran, seperti “setelah pergeseran lalu lintas produksi”), masing-masing dengan tanggung jawab khusus dan pos pemeriksaan validasi. Memahami tahapan ini membantu Anda memantau kemajuan penerapan dan memecahkan masalah secara efektif.

Setiap tahap siklus hidup dapat bertahan hingga 24 jam. Kami menyarankan agar nilainya tetap di bawah tanda 24 jam. Ini karena proses asinkron membutuhkan waktu untuk memicu kait. Waktu sistem habis, gagal penerapan, dan kemudian memulai rollback setelah tahap mencapai 24 jam. AWS CloudFormation penerapan memiliki batasan batas waktu tambahan. Sementara batas tahap 24 jam tetap berlaku, AWS CloudFormation memberlakukan batas 36 jam pada seluruh penyebaran. AWS CloudFormation gagal penerapan, dan kemudian memulai rollback jika proses tidak selesai dalam waktu 36 jam.

Tahapan siklus hidup	Deskripsi	Gunakan tahap ini untuk kait siklus hidup?
RECONCILE_SERVICE	Tahap ini hanya terjadi ketika Anda memulai penyebaran layanan baru dengan lebih dari 1 revisi layanan dalam status AKTIF.	Ya
PRE_SCALE_UP	Revisi layanan hijau belum dimulai. Revisi layanan biru menangani 100% lalu lintas produksi. Tidak ada lalu lintas uji.	Ya
SKALA_UP	Waktu ketika revisi layanan hijau menskalakan hingga 100% dan meluncurkan tugas baru. Revisi layanan hijau tidak melayani lalu lintas apa pun pada saat ini.	Tidak
POST_SCALE_UP	Revisi layanan hijau telah dimulai. Revisi layanan biru menangani 100% lalu lintas produksi. Tidak ada lalu lintas uji.	Ya
TEST_TRAFFIC_SHIFT	Revisi layanan biru dan hijau sedang berjalan. Revisi layanan biru menangani 100% lalu lintas produksi. Revisi layanan hijau bermigrasi dari 0% ke 100% lalu lintas pengujian.	Ya
POST_TEST_TRAFFIC_ SHIFT	Pergeseran lalu lintas uji selesai. Revisi layanan hijau menangani 100% lalu lintas pengujian.	Ya
PRODUCTION_TRAFFIC _SHIFT	Lalu lintas produksi bergeser ke revisi layanan hijau. Revisi layanan hijau bermigrasi dari 0% menjadi 100% dari lalu lintas produksi.	Ya

Tahapan siklus hidup	Deskripsi	Gunakan tahap ini untuk kait siklus hidup?
POST_PRODUCTION_TRAFFIC_SHIFT	Pergeseran lalu lintas produksi selesai.	Ya
BAKE_WAKTU	Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan.	Tidak
MEMBERSIHKAN_UP	Revisi layanan biru telah sepenuhnya diperkecil menjadi 0 tugas yang sedang berjalan. Revisi layanan hijau sekarang menjadi revisi layanan produksi setelah tahap ini.	Tidak

Setiap tahap siklus hidup mencakup pos pemeriksaan validasi bawaan yang harus diteruskan sebelum melanjutkan ke tahap berikutnya. Jika validasi gagal, penerapan dapat secara otomatis digulirkan kembali untuk menjaga ketersediaan dan keandalan layanan.

Sumber daya yang diperlukan untuk penerapan Amazon ECS blue/green

Untuk menggunakan blue/green penyebaran dengan pemindahan lalu lintas terkelola, layanan Anda harus menggunakan salah satu fitur berikut:

- Penyeimbang Beban Elastis
- Layanan Connect

Layanan yang tidak menggunakan Service Discovery, Service Connect, VPC Lattice atau Elastic Load Balancing juga dapat blue/green menggunakan penerapan, tetapi tidak mendapatkan manfaat pemindahan lalu lintas terkelola.

Daftar berikut memberikan ikhtisar tingkat tinggi tentang apa yang perlu Anda konfigurasi untuk penerapan Amazon ECS blue/green :

- Layanan Anda menggunakan Application Load Balancer, Network Load Balancer, atau Service Connect. Konfigurasikan sumber daya yang sesuai.

  - Application Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Application Load Balancer untuk penerapan blue/green
  - Network Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS
  - Service Connect - Untuk informasi selengkapnya, lihatSumber daya Service Connect untuk penerapan biru/hijau Amazon ECS.
- Setel pengontrol penyebaran layanan ke`ECS`.
- Konfigurasikan strategi penerapan seperti `blue/green` dalam definisi layanan Anda.
- Secara opsional, konfigurasikan parameter tambahan seperti:
  - Waktu panggang untuk penerapan baru
  - CloudWatch alarm untuk rollback otomatis
  - Kait siklus hidup penerapan untuk pengujian (ini adalah fungsi Lambda yang berjalan pada tahap penerapan tertentu)

Praktik terbaik

Ikuti praktik terbaik berikut untuk penerapan Amazon ECS blue/green yang sukses:

- Konfigurasikan pemeriksaan kesehatan yang sesuai yang secara akurat mencerminkan kesehatan aplikasi Anda.
- Tetapkan waktu pemanggangan yang memungkinkan pengujian penerapan hijau yang memadai.
- Menerapkan CloudWatch alarm untuk secara otomatis mendeteksi masalah dan memicu rollback.
- Gunakan kait siklus hidup untuk melakukan pengujian otomatis pada setiap tahap penerapan.
- Pastikan aplikasi Anda dapat menangani lingkungan biru dan hijau yang berjalan secara bersamaan.
- Rencanakan kapasitas cluster yang cukup untuk menangani kedua lingkungan selama penerapan.
- Uji prosedur rollback Anda sebelum menerapkannya dalam produksi.

Sumber daya Application Load Balancer untuk penerapan blue/green

Untuk menggunakan Application Load Balancers dengan blue/green penerapan Amazon ECS, Anda perlu mengonfigurasi sumber daya tertentu yang memungkinkan perutean lalu lintas antara revisi layanan biru dan hijau.

Batasan

Batasan berikut saat ini berlaku untuk blue/green penerapan dengan Application Load Balancers:

- Penerapan biru/hijau tidak didukung saat menggunakan pendengar Application Load Balancer yang memiliki beberapa tipe tindakan yang dikonfigurasi dengan tindakan non-penerusan. Jika aturan pendengar Application Load Balancer berisi campuran tindakan penerusan dan non-penerusan (seperti tindakan pengalihan, respons tetap, atau mengautentikasi), penerapan akan gagal.

Kelompok-kelompok target

Untuk blue/green penerapan dengan Elastic Load Balancing, Anda perlu membuat dua grup target:

- Kelompok sasaran utama untuk revisi layanan biru (lalu lintas produksi saat ini)
- Grup target alternatif untuk revisi layanan hijau (versi baru)

Kedua grup target harus dikonfigurasi dengan pengaturan berikut:

- Jenis target: `IP` (untuk Fargate atau EC2 dengan mode `awsvpc` jaringan)
- Protokol: `HTTP` (atau protokol yang digunakan aplikasi Anda)
- Port: Port yang didengarkan aplikasi Anda (biasanya `80` untuk HTTP)
- VPC: VPC yang sama dengan tugas Amazon ECS Anda
- Pengaturan pemeriksaan Kesehatan: Dikonfigurasi untuk memeriksa kesehatan aplikasi Anda dengan benar

Selama blue/green penerapan, Amazon ECS secara otomatis mendaftarkan tugas dengan grup target yang sesuai berdasarkan tahap penerapan.

Example Membuat grup target untuk Application Load Balancer

Perintah CLI berikut membuat dua kelompok target untuk digunakan dengan Application Load Balancer dalam penerapan: blue/green

```
aws elbv2 create-target-group \
    --name blue-target-group \
    --protocol HTTP \
    --port 80 \
```

```
    --vpc-id vpc-abcd1234 \
    --target-type ip \
    --health-check-path / \
    --health-check-protocol HTTP \
    --health-check-interval-seconds 30 \
    --health-check-timeout-seconds 5 \
    --healthy-threshold-count 2 \
    --unhealthy-threshold-count 2

aws elbv2 create-target-group \
    --name green-target-group \
    --protocol HTTP \
    --port 80 \
    --vpc-id vpc-abcd1234 \
    --target-type ip \
    --health-check-path / \
    --health-check-protocol HTTP \
    --health-check-interval-seconds 30 \
    --health-check-timeout-seconds 5 \
    --healthy-threshold-count 2 \
    --unhealthy-threshold-count 2
```

Penyeimbang Beban Aplikasi

Anda perlu membuat Application Load Balancer dengan konfigurasi berikut:

- Skema: Menghadap internet atau internal, tergantung pada kebutuhan Anda
- Jenis alamat IP: IPv4
- VPC: VPC yang sama dengan tugas Amazon ECS Anda
- Subnet: Setidaknya dua subnet di Availability Zone yang berbeda
- Grup keamanan: Grup keamanan yang memungkinkan lalu lintas di port pendengar

Grup keamanan yang dilampirkan pada Application Load Balancer harus memiliki aturan keluar yang memungkinkan lalu lintas ke grup keamanan yang dilampirkan ke tugas Amazon ECS Anda.

Example Membuat Application Load Balancer

Perintah CLI berikut membuat Load Balancer Aplikasi untuk digunakan dalam penerapan biru/hijau:

```
aws elbv2 create-load-balancer \
    --name my-application-load-balancer \
```

```
    --type application \
    --security-groups sg-abcd1234 \
    --subnets subnet-12345678 subnet-87654321
```

Pendengar dan aturan

Untuk blue/green penerapan, Anda perlu mengonfigurasi listener di Application Load Balancer Anda:

- Pendengar produksi: Menangani lalu lintas produksi (biasanya di port 80 atau 443)
  - Awalnya meneruskan lalu lintas ke kelompok sasaran utama (revisi layanan biru)
  - Setelah penyebaran, teruskan lalu lintas ke grup target alternatif (revisi layanan hijau)
- Uji pendengar (opsional): Menangani lalu lintas pengujian untuk memvalidasi revisi layanan hijau sebelum mengalihkan lalu lintas produksi
  - Dapat dikonfigurasi pada port yang berbeda (misalnya, 8080 atau 8443)
  - Meneruskan lalu lintas ke kelompok sasaran alternatif (revisi layanan hijau) selama pengujian

Selama blue/green penerapan, Amazon ECS secara otomatis memperbarui aturan pendengar untuk merutekan lalu lintas ke grup target yang sesuai berdasarkan tahap penerapan.

Example Membuat pendengar produksi

Perintah CLI berikut membuat pendengar produksi pada port 80 yang meneruskan lalu lintas ke grup target utama (biru):

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/
app/my-application-load-balancer/abcdef123456 \
    --protocol HTTP \
    --port 80 \
    --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
blue-target-group/abcdef123456
```

Example Membuat pendengar tes

Perintah CLI berikut membuat pendengar pengujian pada port 8080 yang meneruskan lalu lintas ke grup target alternatif (hijau):

```
aws elbv2 create-listener \
```

```
    --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/
app/my-application-load-balancer/abcdef123456 \
    --protocol HTTP \
    --port 8080 \
    --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
green-target-group/ghijkl789012
```

Example Membuat aturan pendengar untuk perutean berbasis jalur

Perintah CLI berikut membuat aturan yang meneruskan lalu lintas untuk jalur tertentu ke grup target hijau untuk pengujian:

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:region:123456789012:listener/app/my-
application-load-balancer/abcdef123456/ghijkl789012 \
    --priority 10 \
    --conditions Field=path-pattern,Values='/test/*' \
    --actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
green-target-group/ghijkl789012
```

Example Membuat aturan pendengar untuk perutean berbasis header

Perintah CLI berikut membuat aturan yang meneruskan lalu lintas dengan header tertentu ke grup target hijau untuk pengujian:

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:region:123456789012:listener/app/my-
application-load-balancer/abcdef123456/ghijkl789012 \
    --priority 20 \
    --conditions Field=http-header,HttpHeaderConfig='{Name=X-
Environment,Values=[test]}' \
    --actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
green-target-group/ghijkl789012
```

Konfigurasi layanan

Anda harus memiliki izin untuk mengizinkan Amazon ECS mengelola sumber daya penyeimbang beban di klaster atas nama Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban.

Saat membuat atau memperbarui layanan Amazon ECS untuk blue/green penerapan dengan Elastic Load Balancing, Anda perlu menentukan konfigurasi berikut.

Ganti *user-input* dengan nilai-nilai Anda.

Komponen kunci dalam konfigurasi ini adalah:

- `targetGroupArn`: ARN dari kelompok sasaran utama (revisi layanan biru).
- `alternateTargetGroupArn`: ARN dari kelompok sasaran alternatif (revisi layanan hijau).
- `productionListenerRule`: ARN dari aturan pendengar untuk lalu lintas produksi.
- `roleArn`: ARN dari peran yang memungkinkan Amazon ECS mengelola sumber daya Elastic Load Balancing.
- `strategy`: Setel `BLUE_GREEN` untuk mengaktifkan penerapan biru/hijau.
- `bakeTimeInMinutes`: Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
- `TestListenerRule`: ARN dari aturan pendengar untuk lalu lintas uji. Ini adalah parameter opsional.

```
{
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/primary-target-group/
abcdef123456",
            "containerName": "container-name",
            "containerPort": 80,
            "advancedConfiguration": {
                "alternateTargetGroupArn":
 "arn:aws:elasticloadbalancing:region:account-id:targetgroup/alternate-target-group/
ghijkl789012",
                "productionListenerRule": "arn:aws:elasticloadbalancing:region:account-
id:listener-rule/app/load-balancer-name/abcdef123456/listener/ghijkl789012/rule/
mnopqr345678",
                "roleArn": "arn:aws:iam::123456789012:role/ecs-elb-role"
            }
        }
    ],
    "deploymentConfiguration": {
        "strategy": "BLUE_GREEN",
```

```
        "maximumPercent": 200,
        "minimumHealthyPercent": 100,
        "bakeTimeInMinutes": 5
    }
}
```

Arus lalu lintas selama penyebaran

Selama blue/green penyebaran dengan Elastic Load Balancing, lalu lintas mengalir melalui sistem sebagai berikut:

1. Status awal: Semua lalu lintas produksi diarahkan ke kelompok sasaran utama (revisi layanan biru).
2. Penyebaran revisi layanan hijau: Amazon ECS menyebarkan tugas baru dan mendaftarkannya ke grup target alternatif.
3. Lalu lintas pengujian: Jika pendengar pengujian dikonfigurasi, lalu lintas pengujian dirutekan ke grup target alternatif untuk memvalidasi revisi layanan hijau.
4. Pergeseran lalu lintas produksi: Amazon ECS memperbarui aturan pendengar produksi untuk merutekan lalu lintas ke grup target alternatif (revisi layanan hijau).
5. Waktu memanggang: Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
6. Penyelesaian: Setelah penerapan berhasil, revisi layanan biru dihentikan.

Jika masalah terdeteksi selama penerapan, Amazon ECS dapat secara otomatis memutar kembali dengan merutekan lalu lintas kembali ke grup target utama (revisi layanan biru).

Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS

Untuk menggunakan Network Load Balancer dengan blue/green penerapan Amazon ECS, Anda perlu mengonfigurasi sumber daya tertentu yang memungkinkan perutean lalu lintas antara revisi layanan biru dan hijau. Bagian ini menjelaskan komponen yang diperlukan dan konfigurasinya.

Jika konfigurasi Anda menyertakan Network Load Balancer, Amazon ECS menambahkan penundaan 10 menit ke tahapan siklus hidup berikut:

- PRE_SCALE_UP
- TEST_TRAFFIC_SHIFT
- PRODUCTION_TRAFFIC_SHIFT

Penundaan ini menyebabkan masalah waktu Network Load Balancer yang dapat menyebabkan ketidakcocokan antara bobot lalu lintas yang dikonfigurasi dan perutean lalu lintas aktual di bidang data.

Kelompok-kelompok target

Untuk blue/green penerapan dengan Network Load Balancer, Anda perlu membuat dua grup target:

- Kelompok sasaran utama untuk revisi layanan biru (lalu lintas produksi saat ini)
- Kelompok sasaran alternatif untuk revisi layanan hijau (revisi layanan baru)

Kedua grup target harus dikonfigurasi dengan pengaturan berikut:

- Jenis target: `ip` (untuk Fargate atau EC2 dengan mode `awsvpc` jaringan)
- Protokol: `TCP` (atau protokol yang digunakan aplikasi Anda)
- Port: Port yang didengarkan aplikasi Anda (biasanya `80` untuk HTTP)
- VPC: VPC yang sama dengan tugas Amazon ECS Anda
- Pengaturan pemeriksaan Kesehatan: Dikonfigurasi untuk memeriksa kesehatan aplikasi Anda dengan benar

  Untuk pemeriksaan kesehatan TCP, Network Load Balancer membuat koneksi TCP dengan target. Jika koneksi berhasil, target dianggap sehat.

  Untuk pemeriksaan HTTP/HTTPS kesehatan, Network Load Balancer mengirimkan HTTP/HTTPS permintaan ke target dan memverifikasi respons.

Selama blue/green penerapan, Amazon ECS secara otomatis mendaftarkan tugas dengan grup target yang sesuai berdasarkan tahap penerapan.

Example Membuat grup target untuk Network Load Balancer

Perintah AWS CLI berikut membuat dua grup target untuk digunakan dengan Network Load Balancer dalam penerapan: blue/green

```
aws elbv2 create-target-group \
    --name blue-target-group \
    --protocol TCP \
    --port 80 \
```

```
    --vpc-id vpc-abcd1234 \
    --target-type ip \
    --health-check-protocol TCP

aws elbv2 create-target-group \
    --name green-target-group \
    --protocol TCP \
    --port 80 \
    --vpc-id vpc-abcd1234 \
    --target-type ip \
    --health-check-protocol TCP
```

Penyeimbang Beban Jaringan

Anda perlu membuat Network Load Balancer dengan konfigurasi berikut:

- Skema: Menghadap internet atau internal, tergantung pada kebutuhan Anda
- Jenis alamat IP: IPv4
- VPC: VPC yang sama dengan tugas Amazon ECS Anda
- Subnet: Setidaknya dua subnet di Availability Zone yang berbeda

Tidak seperti Application Load Balancer, Network Load Balancer beroperasi pada lapisan transport (Layer 4) dan tidak menggunakan grup keamanan. Sebagai gantinya, Anda perlu memastikan bahwa grup keamanan yang terkait dengan tugas Amazon ECS Anda mengizinkan lalu lintas dari Network Load Balancer pada port listener.

Example Membuat Network Load Balancer

Perintah AWS CLI berikut membuat Network Load Balancer untuk digunakan dalam penerapan: blue/green

```
aws elbv2 create-load-balancer \
    --name my-network-load-balancer \
    --type network \
    --subnets subnet-12345678 subnet-87654321
```

Pertimbangan untuk menggunakan NLB dengan penerapan blue/green

Saat menggunakan Network Load Balancer untuk blue/green penerapan, pertimbangkan hal berikut:

- Operasi Layer 4: Network Load Balancer beroperasi pada lapisan transport (Layer 4) dan tidak memeriksa konten layer aplikasi (Layer 7). Ini berarti Anda tidak dapat menggunakan header atau jalur HTTP untuk keputusan perutean.
- Pemeriksaan kesehatan: Pemeriksaan kesehatan Network Load Balancer terbatas pada protokol TCP, HTTP, atau HTTPS. Untuk pemeriksaan kesehatan TCP, Network Load Balancer hanya memverifikasi bahwa koneksi dapat dibuat.
- Pelestarian koneksi: Network Load Balancers mempertahankan alamat IP sumber klien, yang dapat berguna untuk tujuan keamanan dan pencatatan.
- Alamat IP statis: Network Load Balancer menyediakan alamat IP statis untuk setiap subnet, yang dapat berguna untuk daftar putih atau ketika klien perlu terhubung ke alamat IP tetap.
- Lalu lintas uji: Karena Network Load Balancer tidak mendukung routing berbasis konten, lalu lintas uji harus dikirim ke port yang berbeda dari lalu lintas produksi.

Pendengar dan aturan

Untuk blue/green penerapan dengan Network Load Balancer, Anda perlu mengonfigurasi listener:

- Pendengar produksi: Menangani lalu lintas produksi (biasanya di port 80 atau 443)
  - Awalnya meneruskan lalu lintas ke kelompok sasaran utama (revisi layanan biru)
  - Setelah penyebaran, teruskan lalu lintas ke grup target alternatif (revisi layanan hijau)
- Uji pendengar (opsional): Menangani lalu lintas pengujian untuk memvalidasi revisi layanan hijau sebelum mengalihkan lalu lintas produksi
  - Dapat dikonfigurasi pada port yang berbeda (misalnya, 8080 atau 8443)
  - Meneruskan lalu lintas ke kelompok sasaran alternatif (revisi layanan hijau) selama pengujian

Tidak seperti Application Load Balancer, Network Load Balancer tidak mendukung aturan routing berbasis konten. Sebaliknya, lalu lintas dirutekan berdasarkan port dan protokol pendengar.

Perintah AWS CLI berikut membuat pendengar produksi dan pengujian untuk Network Load Balancer:

Ganti *user-input* dengan nilai-nilai Anda.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/
net/my-network-lb/1234567890123456 \
```

```
    --protocol TCP \
    --port 80 \
    --default-actions Type=forward,
 TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/blue-
target-group/1234567890123456

aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/
net/my-network-lb/1234567890123456 \
    --protocol TCP \
    --port 8080 \
    --default-actions Type=forward,
 TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/green-
target-group/1234567890123456
```

Konfigurasi layanan

Anda harus memiliki izin untuk mengizinkan Amazon ECS mengelola sumber daya penyeimbang beban di klaster atas nama Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban.

Saat membuat atau memperbarui layanan Amazon ECS untuk blue/green penerapan dengan Network Load Balancer, Anda perlu menentukan konfigurasi berikut:

Ganti *user-input* dengan nilai-nilai Anda.

Komponen kunci dalam konfigurasi ini adalah:

- `targetGroupArn`: ARN dari kelompok sasaran utama (revisi layanan biru)
- `alternateTargetGroupArn`: ARN dari kelompok sasaran alternatif (revisi layanan hijau)
- `productionListenerArn`: ARN pendengar untuk lalu lintas produksi
- `testListenerArn`: (Opsional) ARN pendengar untuk lalu lintas uji
- `roleArn`: ARN dari peran yang memungkinkan Amazon ECS mengelola sumber daya Network Load Balancer
- `strategy`: Setel `BLUE_GREEN` untuk mengaktifkan blue/green penerapan
- `bakeTimeInMinutes`: Durasi untuk menunggu setelah revisi layanan hijau dikerahkan sebelum mengalihkan lalu lintas produksi

```
{
```

```
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/blue-target-
group/1234567890123456",
            "containerName": "container-name",
            "containerPort": 80,
            "advancedConfiguration": {
                "alternateTargetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/green-target-
group/1234567890123456",
                "productionListenerArn":
 "arn:aws:elasticloadbalancing:region:123456789012:listener/net/my-network-
lb/1234567890123456/1234567890123456",
                "testListenerArn":
 "arn:aws:elasticloadbalancing:region:123456789012:listener/net/my-network-
lb/1234567890123456/2345678901234567",
                "roleArn": "arn:aws:iam::123456789012:role/ecs-nlb-role"
            }
        }
    ],
    "deploymentConfiguration": {
        "strategy": "BLUE_GREEN",
        "maximumPercent": 200,
        "minimumHealthyPercent": 100,
        "bakeTimeInMinutes": 5
    }
}
```

Arus lalu lintas selama penyebaran

Selama blue/green penyebaran dengan Network Load Balancer, lalu lintas mengalir melalui sistem sebagai berikut:

1. Status awal: Semua lalu lintas produksi diarahkan ke kelompok sasaran utama (revisi layanan biru).
2. Penyebaran revisi layanan hijau: Amazon ECS menyebarkan tugas baru dan mendaftarkannya ke grup target alternatif.
3. Lalu lintas pengujian: Jika pendengar pengujian dikonfigurasi, lalu lintas pengujian dirutekan ke grup target alternatif untuk memvalidasi revisi layanan hijau.

4. Pergeseran lalu lintas produksi: Amazon ECS memperbarui pendengar produksi untuk merutekan lalu lintas ke grup target alternatif (revisi layanan hijau).
5. Waktu memanggang: Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
6. Penyelesaian: Setelah penerapan berhasil, revisi layanan biru dihentikan.

Jika masalah terdeteksi selama penerapan, Amazon ECS dapat secara otomatis memutar kembali dengan merutekan lalu lintas kembali ke grup target utama (revisi layanan biru).

Sumber daya Service Connect untuk penerapan biru/hijau Amazon ECS

Saat menggunakan Service Connect dengan blue/green penerapan, Anda perlu mengonfigurasi komponen tertentu untuk mengaktifkan perutean lalu lintas yang tepat antara revisi layanan biru dan hijau. Bagian ini menjelaskan komponen yang diperlukan dan konfigurasinya.

Gambaran umum arsitektur

Service Connect membangun kemampuan service discovery dan service mesh melalui proxy sespan terkelola yang secara otomatis disuntikkan ke tugas Amazon ECS Anda. Proksi ini menangani keputusan perutean, percobaan ulang, dan pengumpulan metrik, sambil AWS Cloud Map menyediakan backend registri layanan. Saat Anda menerapkan layanan dengan Service Connect diaktifkan, layanan akan mendaftarkan dirinya sendiri AWS Cloud Map, dan layanan klien menemukannya melalui namespace.

Dalam implementasi Service Connect standar, layanan klien terhubung ke nama layanan logis, dan proxy sidecar menangani perutean ke instance layanan yang sebenarnya. Dengan blue/green penerapan, model ini diperluas untuk menyertakan perutean lalu lintas uji melalui konfigurasi. `testTrafficRules`

Selama blue/green penerapan, komponen-komponen kunci berikut bekerja sama:

- Service Connect Proxy: Semua lalu lintas antar layanan melewati proxy Service Connect, yang membuat keputusan perutean berdasarkan konfigurasi.
- AWS Cloud Map Pendaftaran: Penerapan biru dan hijau mendaftar AWS Cloud Map, tetapi penerapan hijau awalnya terdaftar sebagai titik akhir “uji”.
- Uji Perutean Lalu Lintas: Konfigurasi `testTrafficRules` dalam Service Connect menentukan cara mengidentifikasi dan merutekan lalu lintas pengujian ke penerapan hijau. Hal ini dicapai melalui routing berbasis header, di mana header HTTP tertentu dalam permintaan mengarahkan

lalu lintas ke revisi pengujian. Secara default, Service Connect mengenali `x-amzn-ecs-blue-green-test` header untuk protokol berbasis HTTP ketika tidak ada aturan khusus yang ditentukan.

- Konfigurasi Klien: Semua klien di namespace secara otomatis menerima rute produksi dan pengujian, tetapi hanya permintaan yang cocok dengan aturan pengujian yang akan masuk ke penerapan hijau.

Apa yang membuat pendekatan ini kuat adalah menangani kompleksitas penemuan layanan selama transisi. Saat lalu lintas bergeser dari penyebaran biru ke hijau, semua konektivitas dan mekanisme penemuan diperbarui secara otomatis. Tidak perlu memperbarui catatan DNS, mengkonfigurasi ulang penyeimbang beban, atau menerapkan perubahan penemuan layanan secara terpisah karena mesh layanan menangani semuanya.

Perutean dan pengujian lalu lintas

Service Connect menyediakan kemampuan perutean lalu lintas tingkat lanjut untuk blue/green penerapan, termasuk routing berbasis header dan konfigurasi alias klien untuk skenario pengujian.

Uji aturan header lalu lintas

Selama blue/green penerapan, Anda dapat mengonfigurasi aturan header lalu lintas pengujian untuk merutekan permintaan tertentu ke revisi layanan hijau (baru) untuk tujuan pengujian. Ini memungkinkan Anda untuk memvalidasi versi baru dengan lalu lintas terkontrol sebelum menyelesaikan penerapan.

Service Connect menggunakan perutean berbasis header untuk mengidentifikasi lalu lintas pengujian. Secara default, Service Connect mengenali `x-amzn-ecs-blue-green-test` header untuk protokol berbasis HTTP ketika tidak ada aturan khusus yang ditentukan. Ketika header ini hadir dalam permintaan, proxy Service Connect secara otomatis merutekan permintaan ke penerapan hijau untuk pengujian.

Menguji aturan header lalu lintas memungkinkan Anda untuk:

- Permintaan rute dengan header tertentu ke revisi layanan hijau
- Uji fungsionalitas baru dengan subset lalu lintas
- Validasi perilaku layanan sebelum pemotongan lalu lintas penuh
- Menerapkan strategi pengujian kenari

- Lakukan pengujian integrasi dalam lingkungan seperti produksi

Mekanisme routing berbasis header bekerja mulus dengan arsitektur aplikasi yang ada. Layanan klien tidak perlu mengetahui proses blue/green penerapan - mereka hanya menyertakan header yang sesuai saat mengirim permintaan pengujian, dan proxy Service Connect menangani logika perutean secara otomatis.

Untuk informasi selengkapnya tentang mengonfigurasi aturan header lalu lintas pengujian, lihat ServiceConnectTestTrafficHeaderRulesdi Referensi API Amazon Elastic Container Service.

Aturan pencocokan header

Aturan pencocokan header menentukan kriteria untuk merutekan lalu lintas pengujian selama blue/green penerapan. Anda dapat mengonfigurasi beberapa kondisi pencocokan untuk secara tepat mengontrol permintaan mana yang dirutekan ke revisi layanan hijau.

Pencocokan header mendukung:

- Pencocokan nilai header yang tepat
- Pemeriksaan kehadiran header
- Pencocokan berbasis pola
- Beberapa kombinasi header

Contoh kasus penggunaan termasuk permintaan perutean dengan string agen pengguna tertentu, versi API, atau flag fitur ke layanan hijau untuk pengujian.

Untuk informasi selengkapnya tentang konfigurasi pencocokan header, lihat ServiceConnectTestTrafficHeaderMatchRulesdi Referensi API Amazon Elastic Container Service.

Alias klien untuk penerapan blue/green

Alias klien menyediakan titik akhir DNS yang stabil untuk layanan selama penerapan. blue/green Mereka memungkinkan perutean lalu lintas yang mulus antara revisi layanan biru dan hijau tanpa memerlukan aplikasi klien untuk mengubah titik akhir koneksi mereka.

Selama blue/green penerapan, alias klien:

- Pertahankan nama DNS yang konsisten untuk koneksi klien

- Aktifkan peralihan lalu lintas otomatis antara revisi layanan
- Mendukung strategi migrasi lalu lintas bertahap
- Memberikan kemampuan rollback dengan mengarahkan lalu lintas ke revisi biru

Anda dapat mengonfigurasi beberapa alias klien untuk port atau protokol yang berbeda, memungkinkan arsitektur layanan yang kompleks untuk mempertahankan konektivitas selama penerapan.

Untuk informasi selengkapnya tentang konfigurasi alias klien, lihat ServiceConnectClientAliasdi Referensi API Amazon Elastic Container Service.

Praktik terbaik untuk perutean lalu lintas

Saat menerapkan perutean lalu lintas untuk blue/green penerapan dengan Service Connect, pertimbangkan praktik terbaik berikut:

- Mulai dengan pengujian berbasis header: Gunakan aturan header lalu lintas uji untuk memvalidasi layanan hijau dengan lalu lintas terkontrol sebelum mengalihkan semua lalu lintas.
- Konfigurasikan pemeriksaan kesehatan: Pastikan layanan biru dan hijau memiliki pemeriksaan kesehatan yang sesuai yang dikonfigurasi untuk mencegah lalu lintas perutean ke instance yang tidak sehat.
- Monitor metrik layanan: Lacak indikator kinerja utama untuk kedua revisi layanan selama penerapan untuk mengidentifikasi masalah lebih awal.
- Strategi rollback rencana: Konfigurasikan alias klien dan aturan perutean untuk mengaktifkan rollback cepat ke layanan biru jika masalah terdeteksi.
- Uji logika pencocokan header: Validasi aturan pencocokan header Anda di lingkungan non-produksi sebelum menerapkannya ke penerapan produksi.

Alur kerja blue/green penerapan Service Connect

Memahami cara Service Connect mengelola proses blue/green penerapan membantu Anda menerapkan dan memecahkan masalah penerapan secara efektif. Alur kerja berikut menunjukkan bagaimana komponen yang berbeda berinteraksi selama setiap fase penerapan.

Fase penyebaran

blue/green Penerapan Service Connect berlangsung melalui beberapa fase berbeda:

1. Keadaan Awal: Layanan biru menangani 100% lalu lintas produksi. Semua layanan klien di namespace terhubung ke layanan biru melalui nama layanan logis yang dikonfigurasi di Service Connect.
2. Registrasi Layanan Hijau: Ketika penerapan hijau dimulai, ia mendaftar AWS Cloud Map sebagai titik akhir “uji”. Proxy Service Connect di layanan klien secara otomatis menerima konfigurasi rute produksi dan pengujian.
3. Uji Perutean Lalu Lintas: Permintaan yang berisi header lalu lintas pengujian (sepertix-amzn-ecs-blue-green-test) secara otomatis dirutekan ke layanan hijau oleh proxy Service Connect. Lalu lintas produksi terus mengalir ke layanan biru.
4. Persiapan Pergeseran Lalu Lintas: Setelah pengujian berhasil, proses penyebaran mempersiapkan pergeseran lalu lintas produksi. Layanan biru dan hijau tetap terdaftar dan sehat.
5. Production Traffic Shift: Konfigurasi Service Connect memperbarui untuk merutekan lalu lintas produksi ke layanan hijau. Ini terjadi secara otomatis tanpa memerlukan pembaruan layanan klien atau perubahan DNS.
6. Periode Waktu Panggang: Durasi ketika revisi layanan biru dan hijau berjalan secara bersamaan setelah lalu lintas produksi bergeser.
7. Deregistrasi Layanan Biru: Setelah pergeseran lalu lintas dan validasi berhasil, layanan biru dideregistrasi dari dan dihentikan, menyelesaikan penerapan. AWS Cloud Map

Perilaku proxy Service Connect

Proxy Service Connect memainkan peran penting dalam mengelola lalu lintas selama blue/green penerapan. Memahami perilakunya membantu Anda merancang strategi pengujian dan penerapan yang efektif.

Perilaku proxy kunci selama blue/green penerapan:

- Penemuan Rute Otomatis: Proxy secara otomatis menemukan rute produksi dan pengujian dari AWS Cloud Map tanpa memerlukan restart aplikasi atau perubahan konfigurasi.
- Perutean Berbasis Header: Proxy memeriksa header permintaan masuk dan merutekan lalu lintas ke revisi layanan yang sesuai berdasarkan aturan lalu lintas pengujian yang dikonfigurasi.
- Integrasi Pemeriksaan Kesehatan: Proxy hanya mengarahkan lalu lintas ke instance layanan sehat, secara otomatis mengecualikan tugas yang tidak sehat dari kumpulan perutean.
- Coba lagi dan Pemutusan Sirkuit: Proxy menyediakan logika coba ulang bawaan dan kemampuan pemutusan sirkuit, meningkatkan ketahanan selama penerapan.

- Koleksi Metrik: Proxy mengumpulkan metrik terperinci untuk layanan biru dan hijau, memungkinkan pemantauan komprehensif selama penerapan.

Pembaruan penemuan layanan

Salah satu keuntungan utama menggunakan Service Connect untuk blue/green penerapan adalah penanganan otomatis pembaruan penemuan layanan. blue/green Penerapan tradisional seringkali memerlukan pembaruan DNS yang kompleks atau konfigurasi ulang penyeimbang beban, tetapi Service Connect mengelola perubahan ini secara transparan.

Selama penerapan, Service Connect menangani:

- Pembaruan Namespace: Namespace Service Connect secara otomatis menyertakan titik akhir layanan biru dan hijau, dengan aturan perutean yang sesuai.
- Konfigurasi Klien: Semua layanan klien di namespace secara otomatis menerima informasi perutean yang diperbarui tanpa memerlukan restart atau pemindahan.
- Transisi Bertahap: Pembaruan penemuan layanan terjadi secara bertahap dan aman, memastikan tidak ada gangguan pada permintaan yang sedang berlangsung.
- Rollback Support: Jika rollback diperlukan, Service Connect dapat dengan cepat mengembalikan konfigurasi penemuan layanan untuk mengarahkan lalu lintas kembali ke layanan biru.

Membuat penyebaran Amazon ECS blue/green

Dengan menggunakan blue/green penerapan Amazon ECS, Anda dapat membuat dan menguji perubahan layanan sebelum menerapkannya di lingkungan produksi.

Prasyarat

Lakukan operasi berikut sebelum Anda memulai blue/green penerapan.

1. Konfigurasikan izin yang sesuai.
   - Untuk informasi tentang izin Elastic Load Balancing, lihat. Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban
   - Untuk informasi tentang izin Lambda, lihat Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green
2. blue/green Penerapan Amazon ECS mengharuskan layanan Anda menggunakan salah satu fitur berikut: Konfigurasikan sumber daya yang sesuai.

- Application Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Application Load Balancer untuk penerapan blue/green
- Network Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS
- Service Connect - Untuk informasi selengkapnya, lihatSumber daya Service Connect untuk penerapan biru/hijau Amazon ECS.

3. Buat aturan untuk mengarahkan lalu lintas ke revisi layanan hijau Anda. Untuk informasi selengkapnya, lihat Aturan Listener di Panduan Pengguna Network Load Balancer.
4. Buat grup target untuk revisi layanan hijau Anda. Saat Anda menggunakan mode `awsvpc` jaringan untuk tugas Anda, jenis target harus`ip`. Untuk informasi tentang grup target, lihat Grup target di Panduan Pengguna Network Load Balancer.
5. Putuskan apakah Anda ingin menjalankan fungsi Lambda untuk peristiwa siklus hidup.
   - Pra skala naik
   - Setelah skala naik
   - Uji pergeseran lalu lintas
   - Setelah uji pergeseran lalu lintas
   - Pergeseran lalu lintas produksi
   - Setelah pergeseran lalu lintas produksi

   Buat fungsi Lambda untuk setiap peristiwa siklus hidup. Untuk informasi selengkapnya, lihat Membuat fungsi Lambda dengan konsol di Panduan AWS Lambda Pengembang.

Prosedur

Anda dapat menggunakan konsol atau AWS CLI untuk membuat blue/green layanan Amazon ECS.

Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Tentukan sumber daya dari tempat Anda meluncurkan layanan.

Untuk memulai layanan dari	Langkah-langkah
Klaster	a. Pada halaman Klaster, pilih klaster untuk membuat layanan. Halaman detail cluster ditampilkan. b. Pada tab Layanan, pilih Buat.
Ketentuan tugas	a. Pada halaman Definisi tugas, pilih definisi tugas. b. Dari menu Deploy, pilih Buat layanan.

Halaman Create service ditampilkan.

3. Di bawah rincian Layanan, lakukan hal berikut:

   a. Untuk keluarga definisi Tugas, pilih definisi tugas yang akan digunakan. Kemudian, untuk revisi definisi Tugas, masukkan revisi yang akan digunakan.

   b. Untuk nama Layanan, masukkan nama untuk layanan Anda.

4. Untuk menjalankan layanan di klaster yang ada, untuk klaster yang ada, pilih klaster. Untuk menjalankan layanan di klaster baru, pilih Buat klaster

5. Pilih bagaimana tugas Anda didistribusikan di seluruh infrastruktur klaster Anda. Di bawah Konfigurasi komputasi, pilih opsi Anda.

Opsi komputasi	Langkah-langkah
Strategi penyedia kapasitas	a. Di bawah Opsi komputasi , pilih Strategi penyedia kapasitas. b. Pilih strategi:

Opsi komputasi	Langkah-langkah	
	• Untuk menggunakan strategi penyedia kapasitas default cluster, pilih Use cluster default. • Jika klaster Anda tidak memiliki strategi penyedia kapasitas default, atau menggunakan strategi kustom, pilih Gunakan strategi penyedia kustom, Tambahkan kapasitas, lalu tentukan strategi penyedia kapasitas khusus Anda dengan menentukan Basis, penyedia Kapasitas, dan Berat. **Note** Untuk menggunakan penyedia kapasitas dalam suatu strategi, penyedia kapasitas harus dikaitkan dengan cluster.	

Opsi komputasi	Langkah-langkah
Jenis peluncuran	a. Di bagian Compute options, pilih Launch type. b. Untuk jenis Peluncuran, pilih jenis peluncuran. c. (Opsional) Ketika tipe peluncuran Fargate diwajibkan, untuk Versi platform, tentukan versi platform yang akan digunakan. Jika versi platform tidak ditentuka n, versi LATEST platform akan digunakan.

6. Di bawah konfigurasi Deployment, lakukan hal berikut:

   a. Untuk jenis Layanan, pilih Replika.

   b. Untuk tugas yang diinginkan, masukkan jumlah tugas untuk diluncurkan dan dipelihara dalam layanan.

   c. Agar Amazon ECS memantau distribusi tugas di seluruh Availability Zone, dan mendistribusikannya kembali saat terjadi ketidakseimbangan, di bawah penyeimbangan kembali layanan Availability Zone, pilih penyeimbangan kembali layanan Availability Zone.

   d. Untuk masa tenggang pemeriksaan Kesehatan, masukkan jumlah waktu (dalam detik) bahwa penjadwal layanan mengabaikan Elastic Load Balancing, VPC Lattice, dan pemeriksaan kesehatan kontainer yang tidak sehat setelah tugas pertama kali dimulai. Jika Anda tidak menentukan nilai tenggang pemeriksaan kesehatan, nilai default 0 digunakan.

7. a. Untuk waktu Panggang, masukkan jumlah menit revisi layanan biru dan hijau akan berjalan secara bersamaan sebelum revisi biru dihentikan. Ini memungkinkan waktu untuk verifikasi dan pengujian.

   b. (Opsional) Jalankan fungsi Lambda untuk dijalankan pada tahap penerapan tertentu. Di bawah Deployment lifecycle hooks, pilih tahapan untuk menjalankan kait siklus hidup.

Untuk menambahkan kait siklus hidup:

a. Pilih Tambahkan.

b. Untuk fungsi Lambda, masukkan nama fungsi atau ARN.

c. Untuk Peran, pilih peran IAM yang memiliki izin untuk menjalankan fungsi Lambda.

d. Untuk tahapan Siklus Hidup, pilih tahapan saat fungsi Lambda harus dijalankan.

8. Untuk mengonfigurasi cara Amazon ECS mendeteksi dan menangani kegagalan penerapan, perluas deteksi kegagalan Deployment, lalu pilih opsi Anda.

   a. Untuk menghentikan penerapan saat tugas tidak dapat dimulai, pilih Gunakan pemutus sirkuit penyebaran Amazon ECS.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat pemutus sirkuit penyebaran mengatur penerapan ke status gagal, pilih Rollback on failure.

   b. Untuk menghentikan penerapan berdasarkan metrik aplikasi, pilih Gunakan CloudWatch alarm. Kemudian, dari nama CloudWatch alarm, pilih alarm. Untuk membuat alarm baru, buka CloudWatch konsol.

      Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat CloudWatch alarm menyetel penerapan ke status gagal, pilih Rollback on failure.

9. (Opsional) Untuk menghubungkan layanan Anda menggunakan Service Connect, perluas Service Connect, lalu tentukan yang berikut ini:

   a. Pilih Aktifkan Service Connect.

   b. Di bawah konfigurasi Service Connect, tentukan mode klien.

      - Jika layanan Anda menjalankan aplikasi klien jaringan yang hanya perlu terhubung ke layanan lain di namespace, pilih sisi Klien saja.
      - Jika layanan Anda menjalankan aplikasi jaringan atau layanan web dan perlu menyediakan titik akhir untuk layanan ini, dan terhubung ke layanan lain di namespace, pilih Klien dan server.

   c. Untuk menggunakan namespace yang bukan namespace cluster default, untuk Namespace, pilih namespace layanan.

d. (Opsional) Konfigurasikan aturan header lalu lintas uji untuk blue/green penerapan. Di bawah Uji perutean lalu lintas, tentukan yang berikut ini:

   i. Pilih Aktifkan aturan header lalu lintas pengujian untuk merutekan permintaan tertentu ke revisi layanan hijau selama pengujian.

   ii. Untuk aturan pencocokan Header, konfigurasikan kriteria untuk lalu lintas pengujian perutean:

   - Nama header: Masukkan nama header HTTP untuk dicocokkan (misalnya, `X-Test-Version` atau`User-Agent`).
   - Jenis kecocokan: Pilih kriteria yang cocok:
     - Pencocokan tepat: Permintaan rute di mana nilai header sama persis dengan nilai yang ditentukan
     - Header hadir: Permintaan rute yang berisi header yang ditentukan, terlepas dari nilainya
     - Pencocokan pola: Permintaan rute di mana nilai header cocok dengan pola tertentu
   - Nilai header (jika menggunakan kecocokan persis atau kecocokan pola): Masukkan nilai atau pola yang cocok.

   Anda dapat menambahkan beberapa aturan pencocokan header untuk membuat logika perutean yang kompleks. Permintaan yang cocok dengan aturan yang dikonfigurasi akan diarahkan ke revisi layanan hijau untuk pengujian.

   iii. Pilih Tambahkan aturan header untuk mengonfigurasi kondisi pencocokan header tambahan.

> **Note**
>
> Aturan header lalu lintas uji memungkinkan Anda memvalidasi fungsionalitas baru dengan lalu lintas terkontrol sebelum menyelesaikan penerapan penuh. Ini memungkinkan Anda untuk menguji revisi layanan hijau dengan permintaan khusus (seperti yang dari alat pengujian internal atau pengguna beta) sambil mempertahankan arus lalu lintas normal ke revisi layanan biru.

e. (Opsional) Tentukan konfigurasi log. Pilih Gunakan koleksi log. Opsi default mengirimkan log kontainer ke CloudWatch Log. Opsi driver log lainnya dikonfigurasi menggunakan AWS FireLens. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.

   Berikut ini menjelaskan setiap tujuan log kontainer secara lebih rinci.

   - Amazon CloudWatch — Konfigurasikan tugas untuk mengirim log kontainer ke CloudWatch Log. Opsi driver log default disediakan, yang membuat grup CloudWatch log atas nama Anda. Untuk menentukan nama grup log yang berbeda, ubah nilai opsi driver.
   - Amazon Data Firehose — Konfigurasikan tugas untuk mengirim log kontainer ke Firehose. Opsi driver log default disediakan, yang mengirim log ke aliran pengiriman Firehose. Untuk menentukan nama aliran pengiriman yang berbeda, ubah nilai opsi driver.
   - Amazon Kinesis Data Streams — Konfigurasikan tugas untuk mengirim log kontainer ke Kinesis Data Streams. Opsi driver log default disediakan, yang mengirim log ke aliran Kinesis Data Streams. Untuk menentukan nama aliran yang berbeda, ubah nilai opsi driver.
   - Amazon OpenSearch Service - Konfigurasikan tugas untuk mengirim log kontainer ke domain OpenSearch Layanan. Opsi driver log harus disediakan.
   - Amazon S3 — Konfigurasikan tugas untuk mengirim log kontainer ke bucket Amazon S3. Opsi driver log default disediakan, tetapi Anda harus menentukan nama bucket Amazon S3 yang valid.

10. (Opsional) Konfigurasikan Load balancing untuk penerapan biru/hijau.

Jenis Elastic Load Balancing	Langkah-langkah	
Penyeimbang Beban Aplikasi	a. Untuk tipe Load balancer, pilih Application Load Balancer. b. Pilih Buat penyeimba ng beban baru untuk membuat Application Load Balancer baru atau	

Jenis Elastic Load Balancing	Langkah-langkah	
	Gunakan penyeimbang beban yang ada untuk memilih Application Load Balancer yang ada. c. Untuk Container, pilih wadah yang menghosting layanan. d. Untuk nama Load balancer, masukkan nama unik. e. Untuk Listener, masukkan port dan protokol untuk Application Load Balancer untuk mendengarkan permintaan koneksi. Secara default, penyeimbang beban akan dikonfigurasi untuk menggunakan port 80 dan HTTP. • Untuk aturan Produksi, masukkan urutan Evaluasi dan pola Path untuk aturan tersebut. Aturan ini untuk lalu lintas revisi layanan produksi (biru) Anda. • Untuk aturan Uji, masukkan urutan Evaluasi dan pola Path untuk aturan tersebut.	

Jenis Elastic Load Balancing	Langkah-langkah	
	Aturan ini untuk lalu lintas revisi layanan pengujian (hijau) Anda. f. Untuk grup Target, konfigurasikan berikut ini: • Untuk nama grup Target, masukkan nama dan protokol untuk grup target yang diminta oleh Application Load Balancer. • Untuk Protocol, pilih protokol untuk grup target yang diminta oleh Application Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda. • Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. • Untuk jalur pemeriksaan Kesehatan,	

Jenis Elastic Load Balancing	Langkah-langkah	
	masukkan jalur yang ada di dalam wadah Anda di mana Application Load Balancer secara berkala mengirimkan permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (/). • Untuk nama grup Alternatif, masukkan nama grup untuk grup target untuk revisi layanan pengujian (hijau) Anda.	

Jenis Elastic Load Balancing	Langkah-langkah	
Penyeimbang Beban Jaringan	a. Untuk jenis Load balancer, pilih Network Load Balancer. b. Untuk Load Balancer, pilih Network Load Balancer yang sudah ada. c. Untuk Pilih kontainer untuk memuat keseimbangan, pilih kontainer yang menampung layanan. d. Untuk Production listener, pilih port Production listener, dan protokol Production listener. Ini adalah pendengar untuk lalu lintas revisi layanan produksi (biru) Anda. e. Untuk Test listener, pilih port Test listener, dan protokol Test listener. Ini adalah pendengar untuk lalu lintas revisi layanan pengujian (hijau) Anda. f. Untuk grup Target, konfigurasikan berikut ini:	

Jenis Elastic Load Balancing	Langkah-langkah	
	• Untuk nama grup Target, masukkan nama dan protokol untuk grup target yang diminta oleh Network Load Balancer. • Untuk Protocol, pilih protokol untuk grup target yang diminta oleh Network Load Balancer. Secara default, grup target merutekan permintaan ke kontainer pertama yang ditentukan dalam definisi tugas Anda. • Untuk penundaan degregistrasi, masukkan jumlah detik untuk penyeimbang beban untuk mengubah status target menjadi. `UNUSED` Waktu default adalah 300 detik. • Untuk jalur pemeriksaan Kesehatan, masukkan jalur yang ada di dalam wadah Anda di mana Application Load Balancer secara berkala mengirimk	

Jenis Elastic Load Balancing	Langkah-langkah	
	an permintaan untuk memverifikasi kesehatan koneksi antara Application Load Balancer dan container. Defaultnya adalah root directory (/). • Untuk nama grup Alternatif, masukkan nama grup untuk grup target untuk revisi layanan pengujian (hijau) Anda.	

11. (Opsional) Untuk membantu mengidentifikasi layanan dan tugas Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag definisi tugas, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Definisi tugas.

    Agar Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dengan nama cluster dan tag layanan, pilih Aktifkan tag terkelola Amazon ECS, lalu untuk menyebarkan tag dari, pilih Layanan.

    Menambah atau menghapus tanda.

    - [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:
      - Untuk Kunci, masukkan nama kunci.
      - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.
12. Pilih Buat.

AWS CLI

1. Buat file dengan nama `service-definition.json` dengan konten berikut ini.

   Ganti *`user-input`* dengan nilai-nilai Anda.

```
{
  "serviceName": "myBlueGreenService",
  "cluster": "arn:aws:ecs:us-west-2:123456789012:cluster/sample-fargate-
cluster",
  "taskDefinition": "sample-fargate:1",
  "desiredCount": 5,
  "launchType": "FARGATE",
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-09ce6e74c116a2299",
        "subnet-00bb3bd7a73526788",
        "subnet-0048a611aaec65477"
      ],
      "securityGroups": [
        "sg-09d45005497daa123"
      ],
      "assignPublicIp": "ENABLED"
    }
  },
  "deploymentController": {
    "type": "ECS"
  },
  "deploymentConfiguration": {
    "strategy": "BLUE_GREEN",
    "maximumPercent": 200,
    "minimumHealthyPercent": 100,
    "bakeTimeInMinutes": 2,
    "alarms": {
      "alarmNames": [
        "myAlarm"
      ],
      "rollback": true,
      "enable": true
    },
    "lifecycleHooks": [
      {
```

```
        "hookTargetArn": "arn:aws:lambda:us-
west-2:7123456789012:function:checkExample",
        "roleArn": "arn:aws:iam::123456789012:role/ECSLifecycleHookInvoke",
        "lifecycleStages": [
          "PRE_SCALE_UP"
        ]
      }
    ]
  },
  "loadBalancers": [
    {
      "targetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-target-group/54402ff563af1197",
      "containerName": "fargate-app",
      "containerPort": 80,
      "advancedConfiguration": {
        "alternateTargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-target-group/cad10a56f5843199",
        "productionListenerRule": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-blue-green-
demo/32e0e4f946c3c05b/9cfa8c482e204f7d/831dbaf72edb911",
        "roleArn": "arn:aws:iam::123456789012:role/LoadBalancerManagementforECS"
      }
    }
  ]
}
```

2. Jalankan `create-service`.

   Ganti *user-input* dengan nilai-nilai Anda.

   ```
   aws ecs create-service --cli-input-json file://service-definition.json
   ```

   Atau, Anda dapat menggunakan contoh berikut yang membuat layanan blue/green penyebaran dengan konfigurasi penyeimbang beban:

   ```
   aws ecs create-service \
      --cluster "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster" \
      --service-name "blue-green-example-service" \
      --task-definition "nginxServer:1" \
      --launch-type "FARGATE" \
   ```

```
   --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12345,subnet-67890,subnet-abcdef,subnet-
fedcba],securityGroups=[sg-12345],assignPublicIp=ENABLED}" \
   --desired-count 3 \
   --deployment-controller "type=ECS" \
   --deployment-configuration
 "strategy=BLUE_GREEN,maximumPercent=200,minimumHealthyPercent=100,bakeTimeInMinutes=0"
 \
   --load-balancers "targetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/MyBGtg1/
abcdef1234567890,containerName=nginx,containerPort=80,advancedConfiguration={alternateTa
west-2:123456789012:targetgroup/
MyBGtg2/0987654321fedcba,productionListenerRule=arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/
MyLB/1234567890abcdef/1234567890abcdef,roleArn=arn:aws:iam::123456789012:role/
ELBManagementRole}"
```

Langkah selanjutnya

- Perbarui layanan untuk memulai penyebaran. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.
- Pantau proses penyebaran untuk memastikannya mengikuti blue/green pola:
  - Revisi layanan hijau dibuat dan ditingkatkan
  - Lalu lintas uji diarahkan ke revisi hijau (jika dikonfigurasi)
  - Lalu lintas produksi dialihkan ke revisi hijau
  - Setelah waktu memanggang, revisi biru dihentikan

Menyebarkan layanan Amazon ECS menggunakan pengontrol pihak ketiga

Jenis penyebaran eksternal memungkinkan Anda menggunakan pengontrol penyebaran pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS. Detail untuk layanan Anda dikelola, baik oleh tindakan API manajemen layanan (`CreateService`, `UpdateService`, dan `DeleteService`) ataupun tindakan API manajemen set tugas (`CreateTaskSet`, `UpdateTaskSet`, `UpdateServicePrimaryTaskSet`, dan `DeleteTaskSet`). Setiap tindakan API mengelola subset parameter definisi layanan.

Tindakan API `UpdateService` memperbarui jumlah yang diinginkan dan parameter masa tenggang pemeriksaan kondisi untuk suatu layanan. Jika tipe peluncuran, versi platform, detail penyeimbang

beban, konfigurasi jaringan, atau penentuan tugas perlu diperbarui, maka Anda harus membuat set tugas baru.

Tindakan API `UpdateTaskSet` hanya memperbarui parameter skala untuk set tugas.

Tindakan API `UpdateServicePrimaryTaskSet` memodifikasi set tugas mana dalam layanan yang merupakan set tugas utama. Saat Anda memanggil tindakan API `DescribeServices`, tindakan tersebut mengembalikan semua bidang yang ditentukan untuk set tugas utama. Jika set tugas utama untuk layanan diperbarui, maka nilai parameter set tugas apa pun yang ada di set tugas utama baru yang berbeda dari set tugas utama lama di layanan akan diperbarui ke nilai baru saat set tugas utama baru ditentukan. Jika tidak ada set tugas utama yang ditentukan untuk layanan, maka saat menjelaskan layanan, bidang set tugas adalah nol.

Pertimbangan penyebaran eksternal

Pertimbangkan hal berikut saat menggunakan tipe deployment eksternal:

- Jenis penyeimbang beban yang didukung adalah Application Load Balancer atau Penyeimbang Beban Jaringan.
- Jenis peluncuran Fargate atau tipe pengontrol `EXTERNAL` penerapan tidak mendukung strategi penjadwalan. `DAEMON`
- AutoScaling Integrasi aplikasi dengan Amazon ECS hanya mendukung layanan Amazon ECS sebagai target. Dimensi skalabel yang didukung untuk Amazon ECS adalah `ecs:service:DesiredCount` - jumlah tugas layanan Amazon ECS. Tidak ada integrasi langsung antara Application AutoScaling dan Amazon ECS task set. Set tugas Amazon ECS menghitung `ComputedDesiredCount` berdasarkan layanan Amazon ECS. `DesiredCount`

Alur kerja penerapan eksternal

Berikut ini adalah alur kerja dasar untuk mengelola penyebaran eksternal di Amazon ECS.

Untuk mengelola layanan Amazon ECS menggunakan pengontrol penyebaran eksternal

1. Buat layanan Amazon ECS. Satu-satunya parameter yang diperlukan adalah nama layanan. Anda dapat menentukan parameter berikut saat membuat layanan menggunakan pengendali deployment eksternal. Semua parameter layanan lainnya ditentukan saat membuat set tugas dalam layanan.

`serviceName`

Tipe: String

Diperlukan: Ya

Nama layanan Anda. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Nama layanan harus unik dalam sebuah klaster, tetapi Anda dapat memiliki layanan yang bernama sama di beberapa klaster dalam satu Wilayah atau lebih.

`desiredCount`

Jumlah instantiasi dari tugas yang ditentukan menetapkan penentuan tugas untuk ditempatkan dan tetap berjalan dalam layanan.

`deploymentConfiguration`

Parameter deployment opsional yang mengendalikan seberapa banyak tugas yang dijalankan selama deployment dan urutan penghentian serta mulainya tugas.

`tags`

Tipe: Array objek

Wajib: Tidak

Metadata yang Anda terapkan ke layanan untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tag terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Ketika layanan dihapus, tag akan dihapus juga. Maksimal 50 tag dapat diterapkan ke layanan. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`enableECSManagedTags`

Menentukan apakah akan menggunakan tag terkelola Amazon ECS untuk tugas dalam layanan. Untuk informasi selengkapnya, lihat Gunakan tag untuk penagihan.

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE`

Wajib: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan untuk tugas-tugas dalam layanan. Jika tidak ada nilai yang ditentukan, tag tidak disalin. Tag hanya dapat disalin ke tugas dalam layanan selama pembuatan layanan. Untuk menambahkan tag ke tugas setelah pembuatan layanan atau pembuatan tugas, gunakan tindakan `TagResource` API.

`schedulingStrategy`

Strategi penjadwalan yang akan digunakan. Layanan yang menggunakan pengendali deployment eksternal hanya mendukung strategi penjadwalan `REPLICA`.

`placementConstraints`

Susunan objek batasan penempatan yang akan digunakan untuk tugas di layanan Anda. Anda dapat menentukan maksimal 10 batasan per tugas (batas ini mencakup batasan dalam penentuan tugas dan batasan yang ditentukan pada waktu aktif). Jika Anda menggunakan tipe peluncuran Fargate, batasan penempatan tugas tidak didukung.

`placementStrategy`

Strategi batasan objek yang akan digunakan untuk tugas-tugas dalam layanan Anda. Anda dapat menentukan maksimal empat aturan strategi per layanan.

Berikut ini adalah contoh penentuan layanan untuk membuat layanan menggunakan pengendali deployment eksternal.

```
{
    "cluster": "",
    "serviceName": "",
    "desiredCount": 0,
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
```

```
}
```

2. Buat set tugas awal. Set tugas berisi detail berikut tentang layanan Anda:

`taskDefinition`

Penentuan tugas untuk tugas-tugas dalam set tugas yang akan digunakan.

`launchType`

Tipe: String

Nilai yang valid: `EC2` | `FARGATE` | `EXTERNAL`

Wajib: Tidak

Jenis peluncuran untuk menjalankan layanan Anda. Jika jenis peluncuran tidak ditentukan, default `capacityProviderStrategy` digunakan secara default. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Jika `launchType` ditentukan, parameter `capacityProviderStrategy` harus dihilangkan.

`platformVersion`

Tipe: String

Wajib: Tidak

Versi platform tempat tugas Anda dalam layanan berjalan. Versi platform hanya ditentukan untuk tugas yang menggunakan tipe peluncuran Fargate. Jika tidak ditentukan, versi terbaru (`LATEST`) digunakan secara default.

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Saat menentukan versi `LATEST` platform saat menjalankan tugas atau membuat layanan, Anda mendapatkan versi platform terbaru yang tersedia untuk tugas Anda. Saat Anda meningkatkan layanan Anda, tugas-tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.

> **Note**
>
> Versi platform tidak ditentukan untuk tugas yang menggunakan jenis EC2 peluncuran.

`loadBalancers`

Objek penyeimbang beban yang mewakili penyeimbang beban yang akan digunakan bersama layanan Anda. Saat menggunakan pengontrol penyebaran eksternal, hanya Application Load Balancer dan Network Load Balancer yang didukung. Jika Anda menggunakan Application Load Balancer, hanya satu grup target Application Load Balancer yang diizinkan per set tugas.

Cuplikan berikut menunjukkan contoh `loadBalancer` objek untuk digunakan.

```
"loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
]
```

> **Note**
>
> Saat menentukan objek `loadBalancer`, Anda harus menentukan `targetGroupArn` dan menghilangkan parameter `loadBalancerName`.

`networkConfiguration`

Konfigurasi jaringan untuk layanan. Parameter ini diperlukan untuk penentuan tugas yang menggunakan mode jaringan `awsvpc` untuk menerima antarmuka jaringan elastisnya sendiri, dan tidak didukung untuk mode jaringan lainnya. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat. Opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate

`serviceRegistries`

Detail registri penemuan layanan yang akan ditetapkan ke layanan ini. Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.

`scale`

Persentase titik mengambang dari jumlah tugas yang diinginkan untuk menempatkan dan agar tetap berjalan di serangkaian tugas. Nilai ditetapkan sebagai total persentase dari `desiredCount` layanan. Nilai yang diterima adalah angka antara 0 hingga 100.

Berikut ini adalah contoh JSON untuk membuat set tugas untuk pengendali deployment eksternal.

```
{
    "service": "",
    "cluster": "",
    "externalId": "",
    "taskDefinition": "",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "launchType": "EC2",
    "capacityProviderStrategy": [
```

```
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "scale": {
        "value": null,
        "unit": "PERCENT"
    },
    "clientToken": ""
}
```

3. Saat perubahan layanan dibutuhkan, gunakan tindakan API `UpdateService`, `UpdateTaskSet`, atau `CreateTaskSet` tergantung pada parameter mana yang Anda perbarui. Jika Anda membuat set tugas, maka gunakan parameter `scale` untuk setiap set tugas di layanan untuk menentukan berapa banyak tugas yang harus tetap berjalan di layanan. Misalnya, jika Anda memiliki layanan yang berisi `tasksetA` dan Anda membuat `tasksetB`, maka Anda dapat menguji validitas `tasksetB` sebelum akan menransisikan lalu lintas produksi ke sana. Anda dapat mengatur `scale` untuk kedua set tugas ke `100`, dan saat Anda siap untuk menransisikan semua lalu lintas produksi ke `tasksetB`, Anda dapat memperbarui `scale` untuk `tasksetA` ke `0` untuk menurunkan skalanya.

CodeDeploy penerapan biru/hijau untuk Amazon ECS

Kami menyarankan Anda menggunakan blue/green penyebaran Amazon ECS. Untuk informasi selengkapnya, lihat Membuat penyebaran Amazon ECS blue/green .

Tipe penyebaran biru/hijau menggunakan model blue/green penerapan yang dikendalikan oleh. CodeDeploy Gunakan jenis penyebaran ini untuk memverifikasi penyebaran baru layanan sebelum mengirim lalu lintas produksi ke sana. Untuk informasi selengkapnya, lihat Apa yang ada CodeDeploy di Panduan AWS CodeDeploy Pengguna. Memvalidasi status layanan Amazon ECS sebelum penerapan

Ada tiga cara lalu lintas dapat bergeser selama blue/green penyebaran:

- Canary — Lalu lintas digeser dalam dua peningkatan. Anda dapat memilih dari opsi Canary yang telah ditentukan sebelumnya yang menentukan persentase lalu lintas yang dialihkan ke set tugas

yang diperbarui pada kenaikan pertama dan interval, dalam menit, sebelum lalu lintas yang tersisa dialihkan pada kenaikan kedua.

- Linear — Lalu lintas digeser dalam peningkatan yang sama dengan jumlah menit yang sama antara setiap kenaikan. Anda dapat memilih dari opsi linier yang telah ditentukan sebelumnya yang menentukan persentase lalu lintas yang dialihkan pada setiap kenaikan dan jumlah menit di antara setiap kenaikan.
- A ll-at-once - Semua lalu lintas digeser dari set tugas asli ke set tugas yang diperbarui sekaligus.

Berikut ini adalah komponen Amazon ECS CodeDeploy yang digunakan saat layanan menggunakan jenis blue/green penerapan:

CodeDeploy aplikasi

Kumpulan sumber CodeDeploy daya. Ini terdiri dari satu atau beberapa grup deployment.

CodeDeploy kelompok penyebaran

Pengaturan deployment. Pengaturan tersebut terdiri dari hal-hal berikut:

- Cluster dan layanan Amazon ECS
- Grup target penyeimbang beban dan informasi listener
- Strategi roll back deployment
- Pengaturan perutean ulang lalu lintas
- Pengaturan pengakhiran revisi asli
- Konfigurasi deployment
- CloudWatch konfigurasi alarm yang dapat diatur untuk menghentikan penerapan
- Pengaturan SNS atau CloudWatch Acara untuk notifikasi

Untuk informasi selengkapnya, lihat Bekerja dengan Grup Penerapan di Panduan AWS CodeDeploy Pengguna.

CodeDeploy konfigurasi penerapan

Menentukan bagaimana CodeDeploy rute lalu lintas produksi ke tugas pengganti Anda ditetapkan selama penerapan. Konfigurasi deployment canary dan linier yang telah ditentukan berikut ini tersedia. Anda juga dapat membuat deployment canary dan linier yang ditentukan secara khusus. Untuk informasi selengkapnya, lihat Bekerja dengan Konfigurasi Penerapan di AWS CodeDeploy Panduan Pengguna.

- CodeDeployDefault. ECSAllAtOnce: Menggeser semua lalu lintas ke wadah Amazon ECS yang diperbarui sekaligus
- CodeDeployDefault. ECSLinear10 PercentEvery 1Menit: Menggeser 10 persen lalu lintas setiap menit sampai semua lalu lintas bergeser.
- CodeDeployDefault. ECSLinear10 PercentEvery 3Menit: Menggeser 10 persen lalu lintas setiap 3 menit sampai semua lalu lintas bergeser.
- CodeDeployDefault. ECSCanary10Percent5Minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisanya 90 persen dikerahkan lima menit kemudian.
- CodeDeployDefault. ECSCanary10Percent15Minutes: Menggeser 10 persen lalu lintas pada kenaikan pertama. Sisa 90 persen di-deploy 15 menit kemudian.

Revisi

Revisi adalah file spesifikasi CodeDeploy aplikasi (AppSpec file). Dalam AppSpec file tersebut, Anda menentukan ARN lengkap dari definisi tugas dan wadah serta port set tugas pengganti Anda di mana lalu lintas akan dirutekan saat penerapan baru dibuat. Nama kontainer harus merupakan salah satu dari nama kontainer yang direferensikan dalam penentuan tugas Anda. Jika konfigurasi jaringan atau versi platform telah diperbarui dalam definisi layanan, Anda juga harus menentukan detail tersebut dalam AppSpec file. Anda juga dapat menentukan fungsi Lambda yang akan dijalankan selama peristiwa siklus hidup penerapan. Fungsi Lambda memungkinkan Anda menjalankan pengujian dan mengembalikan metrik selama penerapan. Untuk informasi selengkapnya, lihat Referensi AppSpec File di Panduan AWS CodeDeploy Pengguna.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan jenis blue/green penerapan:

- Saat layanan Amazon ECS yang menggunakan jenis blue/green penerapan awalnya dibuat, kumpulan tugas Amazon ECS dibuat.
- Anda harus mengonfigurasi layanan untuk menggunakan Application Load Balancer atau Network Load Balancer. Berikut ini adalah persyaratan penyeimbang beban:
  - Anda harus menambahkan listener produksi ke penyeimbang beban, yang digunakan untuk merutekan lalu lintas produksi.
  - Listener uji opsional dapat ditambahkan ke penyeimbang beban, yang digunakan untuk merutekan lalu lintas uji. Jika Anda menentukan listener pengujian, CodeDeploy rute lalu lintas pengujian Anda ke set tugas pengganti selama penerapan.

  - Baik listener produksi maupun listener uji, harus memiliki penyeimbang beban yang sama.
  - Anda harus menentukan grup target untuk penyeimbang beban. Grup target merutekan lalu lintas ke set tugas asli yang di layanan melalui listener produksi.
  - Ketika Network Load Balancer digunakan, hanya konfigurasi `CodeDeployDefault.ECSAllAtOnce` penerapan yang didukung.
- Untuk layanan yang dikonfigurasi untuk menggunakan penskalaan otomatis layanan dan tipe deployment biru/hijau, penskalaan otomatis tidak akan diblokir selama deployment, akan tetapi deployment mungkin akan mengalami kegagalan di beberapa keadaan. Hal-hal berikut menjelaskan perilaku tersebut secara lebih detail.
  - Jika layanan sedang diskalakan dan penerapan dimulai, set tugas hijau dibuat dan CodeDeploy akan menunggu hingga satu jam untuk tugas hijau yang disetel mencapai kondisi tunak dan tidak akan mengalihkan lalu lintas apa pun hingga selesai.
  - Jika layanan sedang dalam proses blue/green penyebaran dan peristiwa penskalaan terjadi, lalu lintas akan terus bergeser selama 5 menit. Jika layanan tidak mencapai kondisi tunak dalam 5 menit, CodeDeploy akan menghentikan penerapan dan menandainya sebagai gagal.
- Tugas yang menggunakan tipe peluncuran Fargate atau tipe pengontrol `CODE_DEPLOY` penerapan tidak mendukung strategi penjadwalan. `DAEMON`
- Saat Anda awalnya membuat grup CodeDeploy aplikasi dan penyebaran, Anda harus menentukan yang berikut:
  - Anda harus menentukan dua grup target untuk penyeimbang beban. Satu grup target harus menjadi grup target awal yang ditentukan untuk penyeimbang beban saat layanan Amazon ECS dibuat. Satu-satunya persyaratan grup target kedua adalah tidak dapat dikaitkan dengan penyeimbang beban yang berbeda dari yang digunakan oleh layanan.
- Saat Anda membuat CodeDeploy penerapan untuk layanan Amazon ECS, CodeDeploy buat set tugas pengganti (atau set tugas hijau) dalam penerapan. Jika Anda menambahkan pendengar pengujian ke penyeimbang beban, CodeDeploy rutekan lalu lintas pengujian Anda ke set tugas pengganti. Ini adalah saat Anda dapat menjalankan uji validasi apa pun. Kemudian CodeDeploy merutekan ulang lalu lintas produksi dari set tugas asli ke set tugas pengganti sesuai dengan pengaturan perutean ulang lalu lintas untuk grup deployment.

Izin IAM yang diperlukan

Blue/green deployments are made possible by a combination of the Amazon ECS and CodeDeploy APIs. Users must have the appropriate permissions for these services before they can use Amazon ECS blue/greenpenyebaran di AWS Management Console atau dengan atau. AWS CLI SDKs

Selain izin IAM standar untuk membuat dan memperbarui layanan, Amazon ECS memerlukan izin berikut. Izin ini telah ditambahkan ke kebijakan `AmazonECS_FullAccess` IAM. Untuk informasi selengkapnya, lihat Amazonecs_ FullAccess.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "cloudwatch:DescribeAlarms",
                "lambda:ListFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

> **Note**
>
> Selain izin Amazon ECS standar yang diperlukan untuk menjalankan tugas dan layanan, pengguna juga memerlukan `iam:PassRole` izin untuk menggunakan peran IAM untuk tugas.

CodeDeploy memerlukan izin untuk memanggil Amazon ECS APIs, memodifikasi Elastic Load Balancing, menjalankan fungsi Lambda, dan CloudWatch menjelaskan alarm, serta izin untuk mengubah jumlah layanan yang diinginkan atas nama Anda. Sebelum membuat layanan Amazon ECS yang menggunakan jenis blue/green penyebaran, Anda harus membuat peran IAM (). `ecsCodeDeployRole` Untuk informasi selengkapnya, lihat Peran Amazon ECS CodeDeploy IAM.

Contoh kebijakan Perbarui contoh layanan Amazon ECS IAM Buat contoh layanan Amazon ECS dan IAM menunjukkan izin yang diperlukan bagi pengguna untuk blue/green menggunakan penerapan Amazon ECS di file. AWS Management Console

Migrasikan penerapan CodeDeploy blue/green deployments to Amazon ECS blue/green

CodeDeploy blue/green and Amazon ECS blue/greenpenerapan menyediakan fungsionalitas serupa, tetapi berbeda dalam cara Anda mengonfigurasi dan mengelolanya.

CodeDeploy ikhtisar penyebaran biru/hijau

Saat membuat layanan Amazon ECS menggunakan CodeDeploy, Anda:

1. Buat penyeimbang beban dengan pendengar produksi dan (opsional) pendengar pengujian. Setiap pendengar dikonfigurasi dengan aturan tunggal (default) yang merutekan semua lalu lintas ke satu grup target (grup target utama).
2. Buat layanan Amazon ECS, dikonfigurasi untuk menggunakan pendengar dan grup target, dengan `deploymentController` jenis yang disetel ke. `CODE_DEPLOY` Pembuatan layanan menghasilkan pembuatan set tugas (biru) yang terdaftar dengan grup target yang ditentukan.
3. Buat grup CodeDeploy penyebaran (sebagai bagian dari CodeDeploy aplikasi), dan konfigurasikan dengan detail klaster Amazon ECS, nama layanan, pendengar penyeimbang beban, dua grup target (grup target utama yang digunakan dalam aturan pendengar produksi, dan grup target sekunder yang akan digunakan untuk tugas pengganti), peran layanan (untuk memberikan izin CodeDeploy untuk memanipulasi Amazon ECS dan sumber daya Elastic Load Balancing) dan berbagai parameter yang mengontrol perilaku penerapan.

Dengan CodeDeploy, versi baru layanan digunakan menggunakan`CreateDeployment()`, menentukan nama CodeDeploy aplikasi, nama grup penyebaran, dan AppSpec file yang memberikan rincian revisi baru dan kait siklus hidup opsional. CodeDeploy Penerapan membuat set tugas pengganti (hijau) dan mendaftarkan tugasnya dengan kelompok target sekunder. Ketika ini menjadi sehat, tersedia untuk pengujian (opsional) dan untuk produksi. Dalam kedua kasus, perutean ulang dicapai dengan mengubah aturan pendengar masing-masing untuk menunjuk ke kelompok target sekunder yang terkait dengan set tugas hijau. Rollback dicapai dengan mengubah aturan pendengar produksi kembali ke grup target utama.

Ikhtisar blue/green penyebaran Amazon ECS

Dengan blue/green penerapan Amazon ECS, Konfigurasi penyebaran adalah bagian dari layanan Amazon ECS itu sendiri:

1. Anda harus melakukan pra-konfigurasi pendengar produksi penyeimbang beban dengan aturan yang mencakup dua grup target dengan bobot 1 dan 0.
2. Anda perlu menentukan sumber daya berikut, atau memperbarui sumber daya layanan:
   - ARN dari aturan pendengar ini
   - Kedua kelompok sasaran
   - Peran IAM untuk memberikan izin Amazon ECS untuk memanggil Elastic Load Balancing APIs
   - Peran IAM opsional untuk menjalankan fungsi Lambda
   - Atur `deploymentController` tipe ke `ECS` dan `deploymentConfiguration.strategy` ke`BLUE_GREEN`. Ini menghasilkan pembuatan penyebaran layanan (biru) yang tugasnya terdaftar pada kelompok target utama.

Dengan Amazon ECS biru/hijau, revisi layanan baru dibuat dengan menelepon Amazon ECS`UpdateService()`, memberikan rincian revisi baru. Penyebaran layanan membuat tugas revisi layanan baru (hijau) dan mendaftarkannya ke grup target sekunder. Amazon ECS menangani operasi perutean ulang dan rollback dengan mengalihkan bobot pada aturan pendengar.

Perbedaan implementasi utama

Sementara kedua pendekatan menghasilkan pembuatan serangkaian tugas awal, implementasi yang mendasarinya berbeda:

- CodeDeploy menggunakan set tugas, sedangkan Amazon ECS menggunakan revisi layanan. Kumpulan tugas Amazon ECS adalah konstruksi lama yang telah digantikan oleh revisi dan

penerapan layanan Amazon ECS. Yang terakhir menawarkan visibilitas yang lebih besar ke dalam proses penyebaran, serta penyebaran layanan dan riwayat revisi layanan.

- Dengan CodeDeploy, kait siklus hidup ditentukan sebagai bagian dari AppSpec file yang dipasok ke. `CreateDeployment()` Ini berarti bahwa kait dapat diubah dari satu penerapan ke penerapan berikutnya. Dengan Amazon ECS biru/hijau, kait ditentukan sebagai bagian dari konfigurasi layanan, dan pembaruan apa pun akan memerlukan panggilan. `UpdateService()`
- Keduanya CodeDeploy dan Amazon ECS blue/green menggunakan Lambda untuk implementasi hook, tetapi input dan output yang diharapkan berbeda.

  Dengan CodeDeploy, fungsi harus memanggil `PutLifecycleEventHookExecutionStatus()` untuk mengembalikan status hook, yang bisa berupa `SUCCEEDED` atau`FAILED`. Dengan Amazon ECS, respons Lambda digunakan untuk menunjukkan status hook.
- CodeDeploy memanggil setiap hook sebagai panggilan satu kali, dan mengharapkan status eksekusi akhir akan dikembalikan dalam waktu satu jam. Kait Amazon ECS lebih fleksibel karena dapat mengembalikan `IN_PROGRESS` indikator, yang menandakan bahwa kait harus dipanggil kembali berulang kali hingga menghasilkan atau. `SUCCEEDED` `FAILED` Untuk informasi selengkapnya, lihat Kait siklus hidup untuk penerapan layanan Amazon ECS.

Pendekatan migrasi

Ada tiga pendekatan utama untuk bermigrasi dari CodeDeploy blue/green to Amazon ECS blue/green penerapan. Setiap pendekatan memiliki karakteristik yang berbeda dalam hal kompleksitas, risiko, kemampuan rollback, dan potensi downtime.

Menggunakan kembali sumber daya Elastic Load Balancing yang sama dengan yang digunakan CodeDeploy

Anda memperbarui layanan Amazon ECS yang ada untuk menggunakan pengontrol penyebaran Amazon ECS dengan strategi penerapan, bukan pengontrol blue/green penerapan. CodeDeploy Pertimbangkan hal berikut saat menggunakan pendekatan ini:

- Prosedur migrasi lebih sederhana karena Anda memperbarui pengontrol penyebaran dan strategi penyebaran layanan Amazon ECS yang ada.
- Tidak ada downtime saat dikonfigurasi dan dimigrasikan dengan benar.
- Rollback mengharuskan Anda mengembalikan revisi layanan.
- Risikonya tinggi karena tidak ada konfigurasi paralel biru/hijau.

Anda menggunakan pendengar penyeimbang beban dan grup target yang sama dengan yang digunakan. CodeDeploy Jika Anda menggunakan AWS CloudFormation, lihatMigrasi template AWS CloudFormation CodeDeploy blue/green deployment template to an Amazon ECS blue/green AWS CloudFormation.

1. Ubah aturan default production/test pendengar untuk menyertakan grup target alternatif dan atur bobot grup target utama menjadi 1 dan grup target alternatif ke 0.

   Untuk CodeDeploy, pendengar penyeimbang beban yang dilampirkan ke layanan dikonfigurasi dengan aturan tunggal (default) yang merutekan semua lalu lintas ke satu grup target. Untuk Amazon ECS biru/hijau, pendengar penyeimbang beban harus dikonfigurasi sebelumnya dengan aturan yang mencakup dua grup target dengan bobot.
2. Perbarui layanan Amazon ECS yang ada dengan memanggil `UpdateService` API dan menyetel parameter `deploymentController` ke`ECS`, dan parameter `deploymentStrategy` ke`BLUE_GREEN`. Tentukan ARNs grup target, grup target alternatif, pendengar produksi, dan pendengar uji opsional.
3. Verifikasi bahwa layanan berfungsi seperti yang diharapkan.
4. Hapus CodeDeploy pengaturan untuk layanan Amazon ECS ini karena Anda sekarang menggunakan Amazon ECS biru/hijau.

Layanan baru dengan penyeimbang beban yang ada

Pendekatan ini menggunakan blue/green strategi untuk migrasi.

Pertimbangkan hal berikut saat menggunakan pendekatan ini:

- Ada gangguan minimal. Ini hanya terjadi selama pertukaran port Elastic Load Balancing.
- Rollback mengharuskan Anda mengembalikan swap port Elastic Load Balancing.
- Risikonya rendah karena ada konfigurasi paralel. Oleh karena itu Anda dapat menguji sebelum shift lalu lintas.

1. Biarkan pendengar, grup target, dan layanan Amazon ECS untuk CodeDeploy penyiapan utuh sehingga Anda dapat dengan mudah mengembalikan ke pengaturan ini jika diperlukan.
2. Buat grup target baru dan pendengar baru (dengan port berbeda dari pendengar asli) di bawah penyeimbang beban yang ada. Kemudian, buat layanan Amazon ECS baru yang cocok dengan layanan Amazon ECS yang ada kecuali yang Anda gunakan `ECS` sebagai pengontrol penerapan,

`BLUE_GREEN` sebagai strategi penerapan, dan teruskan ARNs untuk grup target baru dan aturan pendengar baru.

3. Verifikasi pengaturan baru dengan mengirim lalu lintas HTTP secara manual ke layanan. Jika semuanya berjalan dengan baik, tukar port pendengar asli dan pendengar baru untuk mengarahkan lalu lintas ke pengaturan baru.
4. Verifikasi pengaturan baru, dan jika semuanya terus berfungsi seperti yang diharapkan, hapus CodeDeploy pengaturan.

Layanan baru dengan penyeimbang beban baru

Seperti pendekatan sebelumnya, pendekatan ini menggunakan blue/green strategi untuk migrasi. Perbedaan utamanya adalah peralihan dari CodeDeploy pengaturan ke pengaturan Amazon ECS blue/green terjadi pada lapisan proxy terbalik di atas penyeimbang beban. Contoh implementasi untuk lapisan proxy terbalik adalah Route 53 dan CloudFront.

Pendekatan ini cocok untuk pelanggan yang sudah memiliki lapisan proxy terbalik ini, dan jika semua komunikasi dengan layanan terjadi melaluinya (misalnya, tidak ada komunikasi langsung di tingkat penyeimbang beban).

Pertimbangkan hal berikut saat menggunakan pendekatan ini:

- Ini membutuhkan lapisan proxy terbalik.
- Prosedur migrasi lebih kompleks karena Anda perlu memperbarui pengontrol penyebaran dan strategi penyebaran layanan Amazon ECS yang ada.
- Ada gangguan minimal. Ini hanya terjadi selama pertukaran port Elastic Load Balancing.
- Rollback mengharuskan Anda membalikkan perubahan konfigurasi proxy.
- Risikonya rendah karena ada konfigurasi paralel. Oleh karena itu Anda dapat menguji sebelum shift lalu lintas.

1. Jangan mengubah CodeDeploy pengaturan yang ada secara utuh (penyeimbang beban, pendengar, grup target, layanan Amazon ECS, dan grup penerapan). CodeDeploy
2. Buat penyeimbang beban baru, grup target, dan pendengar yang dikonfigurasi untuk penerapan Amazon ECS. blue/green

   Konfigurasikan sumber daya yang sesuai.

- Application Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Application Load Balancer untuk penerapan blue/green
- Network Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS

3. Buat layanan baru dengan `ECS` sebagai pengontrol penerapan dan `BLUE_GREEN` sebagai strategi penyebaran, menunjuk ke sumber daya penyeimbang beban baru.
4. Verifikasi pengaturan baru dengan mengujinya melalui penyeimbang beban baru.
5. Perbarui konfigurasi proxy terbalik untuk merutekan lalu lintas ke penyeimbang beban baru.
6. Amati revisi layanan baru, dan jika semuanya terus berfungsi seperti yang diharapkan, hapus CodeDeploy pengaturan.

Langkah selanjutnya

Setelah bermigrasi ke penerapan Amazon ECS blue/green :

- Perbarui skrip dan CI/CD pipeline penerapan Anda untuk menggunakan Amazon ECS `UpdateService` API, bukan API. CodeDeploy `CreateDeployment`
- Perbarui pemantauan dan peringatan Anda untuk melacak penerapan layanan Amazon ECS alih-alih penerapan. CodeDeploy
- Pertimbangkan untuk menerapkan pengujian otomatis proses penerapan baru Anda untuk memastikannya berfungsi seperti yang diharapkan.

Migrasi dari penyebaran CodeDeploy blue/green to an Amazon ECS blue/green layanan

Dengan menggunakan blue/green penerapan Amazon ECS, Anda dapat membuat dan menguji perubahan layanan sebelum menerapkannya di lingkungan produksi.

Anda harus membuat kait siklus hidup baru untuk penerapan Amazon ECS Anda. blue/green

Prasyarat

Lakukan operasi berikut sebelum Anda memulai blue/green penerapan.

1. Ganti peran Amazon ECS CodeDeploy IAM dengan izin berikut.
   - Untuk informasi tentang izin Elastic Load Balancing, lihat. Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban

- Untuk informasi tentang izin Lambda, lihat. Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green

2. Matikan CodeDeploy otomatisasi. Untuk informasi selengkapnya, lihat Bekerja dengan grup penerapan CodeDeploy di Panduan CodeDeploy Pengguna.
3. Pastikan Anda memiliki informasi berikut dari CodeDeploy blue/green deployment. You can reuse this information for the Amazon ECS blue/green penerapan Anda:
   - Kelompok sasaran produksi
   - Pendengar produksi
   - Aturan produksi
   - Kelompok sasaran uji

   Ini adalah kelompok sasaran untuk revisi layanan hijau,
4. Pastikan grup target Application Load Balancer terkait dengan aturan listener dengan benar:
   - Jika Anda tidak menggunakan pendengar pengujian, kedua grup target (produksi dan pengujian) harus dikaitkan dengan aturan pendengar produksi.
   - Jika Anda menggunakan pendengar pengujian, satu grup target harus ditautkan ke aturan pendengar produksi dan grup target lainnya harus ditautkan ke aturan pendengar pengujian.

   Jika persyaratan ini tidak terpenuhi, penyebaran layanan akan gagal dengan kesalahan berikut: `Service deployment rolled back because of invalid networking configuration. Both targetGroup and alternateTargetGroup must be associated with the productionListenerRule or testListenerRule.`
5. Verifikasi bahwa tidak ada penerapan layanan yang sedang berlangsung untuk layanan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.
6. blue/green Penerapan Amazon ECS mengharuskan layanan Anda menggunakan salah satu fitur berikut: Konfigurasikan sumber daya yang sesuai.
   - Application Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Application Load Balancer untuk penerapan blue/green
   - Network Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS
   - Service Connect - Untuk informasi selengkapnya, lihatSumber daya Service Connect untuk penerapan biru/hijau Amazon ECS.

7. Putuskan apakah Anda ingin menjalankan fungsi Lambda untuk tahapan siklus hidup untuk tahapan dalam penerapan Amazon ECS. blue/green
   - Pra skala naik
   - Setelah skala naik
   - Uji pergeseran lalu lintas
   - Setelah uji pergeseran lalu lintas
   - Pergeseran lalu lintas produksi
   - Setelah pergeseran lalu lintas produksi

   Buat fungsi Lambda untuk setiap tahap siklus hidup. Untuk informasi selengkapnya, lihat Membuat fungsi Lambda dengan konsol di Panduan AWS Lambda Pengembang.

Untuk informasi selengkapnya tentang memperbarui pengontrol penerapan layanan, lihatPerbarui parameter layanan Amazon ECS.

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.

   Halaman detail cluster ditampilkan.
3. Dari tab Layanan, pilih layanan.

   Halaman detail layanan ditampilkan.
4. Di spanduk, pilih Perbarui jenis pengontrol penerapan.

   Halaman jenis pengontrol penyebaran Migrasi ditampilkan.
5. Perluas Baru, lalu tentukan parameter berikut.
   a. Untuk jenis pengontrol Deployment, pilih ECS.
   b. Untuk strategi Deployment, pilih Biru/hijau.
   c. Untuk waktu Panggang, masukkan waktu revisi layanan biru dan hijau berjalan.
   d. Untuk menjalankan fungsi Lambda untuk tahap siklus hidup, di bawah Deployment lifecyce hooks lakukan hal berikut untuk setiap fungsi Lambda yang unik:
      i. Pilih Tambahkan.

Ulangi untuk setiap fungsi unik yang ingin Anda jalankan.

ii. Untuk fungsi Lambda, masukkan nama fungsi.

iii. Untuk Peran, pilih peran yang Anda buat dalam prasyarat dengan izin biru/hijau.

Untuk informasi selengkapnya, lihat Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green .

iv. Untuk tahapan Siklus Hidup, pilih tahapan yang dijalankan fungsi Lambda.

v. (Opsional) Untuk detail Hook, masukkan pasangan kunci-nilai yang memberikan informasi tentang hook.

6. Perluas Load Balancing, dan konfigurasikan berikut ini:

a. Untuk Peran, pilih peran yang Anda buat dalam prasyarat dengan izin. blue/green

Untuk informasi selengkapnya, lihat Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green .

b. Untuk Listener, pilih pendengar produksi dari penerapan CodeDeploy biru/hijau Anda.

c. Untuk aturan Produksi, pilih aturan produksi dari penerapan CodeDeploy biru/hijau Anda.

d. Untuk aturan Uji, pilih aturan pengujian dari penerapan CodeDeploy biru/hijau Anda.

e. Untuk grup Target, pilih grup target produksi dari penyebaran CodeDeploy biru/hijau Anda.

f. Untuk grup target Alternatif, pilih grup target pengujian dari penerapan CodeDeploy biru/hijau Anda.

7. Pilih Perbarui.

Langkah selanjutnya

- Perbarui layanan untuk memulai penyebaran. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.
- Pantau proses penyebaran untuk memastikannya mengikuti pola biru/hijau:
  - Revisi layanan hijau dibuat dan ditingkatkan
  - Lalu lintas uji diarahkan ke revisi hijau (jika dikonfigurasi)
  - Lalu lintas produksi dialihkan ke revisi hijau
  - Setelah waktu memanggang, revisi biru dihentikan

Migrasi template AWS CloudFormation CodeDeploy blue/green deployment template to an Amazon ECS blue/green AWS CloudFormation

Migrasikan AWS CloudFormation template yang menggunakan strategi CodeDeploy blue/green deployments for Amazon ECS services to one that uses the native Amazon ECS blue/green penerapan. Migrasi mengikuti pendekatan “Gunakan kembali sumber daya Elastic Load Balancing yang sama yang digunakan CodeDeploy untuk”. Untuk informasi lebih lanjut, lihat the section called “Migrasikan penerapan CodeDeploy blue/green deployments to Amazon ECS blue/green” dokumentasi.

Template sumber

Template ini menggunakan `AWS::CodeDeployBlueGreen` transform dan `AWS::CodeDeploy::BlueGreen` hook untuk mengimplementasikan blue/green penerapan untuk layanan Amazon ECS.

Sumber

Ini adalah AWS CloudFormation template lengkap menggunakan penyebaran CodeDeploy biru/hijau. Untuk informasi selengkapnya, lihat contoh templat penerapan biru/hijau di Panduan Pengguna: AWS CloudFormation

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "Vpc": {
      "Type": "AWS::EC2::VPC::Id"
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet::Id"
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet::Id"
    }
  },
  "Transform": [
    "AWS::CodeDeployBlueGreen"
  ],
  "Hooks": {
    "CodeDeployBlueGreenHook": {
      "Type": "AWS::CodeDeploy::BlueGreen",
      "Properties": {
```

```
        "TrafficRoutingConfig": {
          "Type": "TimeBasedCanary",
          "TimeBasedCanary": {
            "StepPercentage": 15,
            "BakeTimeMins": 5
          }
        },
        "Applications": [
          {
            "Target": {
              "Type": "AWS::ECS::Service",
              "LogicalID": "ECSDemoService"
            },
            "ECSAttributes": {
              "TaskDefinitions": [
                "BlueTaskDefinition",
                "GreenTaskDefinition"
              ],
              "TaskSets": [
                "BlueTaskSet",
                "GreenTaskSet"
              ],
              "TrafficRouting": {
                "ProdTrafficRoute": {
                  "Type": "AWS::ElasticLoadBalancingV2::Listener",
                  "LogicalID": "ALBListenerProdTraffic"
                },
                "TargetGroups": [
                  "ALBTargetGroupBlue",
                  "ALBTargetGroupGreen"
                ]
              }
            }
          }
        ]
      }
    }
  },
  "Resources": {
    "ExampleSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Security group for ec2 access",
        "VpcId": {"Ref": "Vpc"},
```

```
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 8080,
            "ToPort": 8080,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    },
    "ALBTargetGroupBlue": {
      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
          "HttpCode": "200"
        },
        "Port": 80,
        "Protocol": "HTTP",
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
```

```
        "VpcId": {"Ref": "Vpc"}
      }
    },
    "ALBTargetGroupGreen": {
      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
          "HttpCode": "200"
        },
        "Port": 80,
        "Protocol": "HTTP",
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
        "VpcId": {"Ref": "Vpc"}
      }
    },
    "ExampleALB": {
      "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties": {
        "Scheme": "internet-facing",
        "SecurityGroups": [
          {"Ref": "ExampleSecurityGroup"}
        ],
        "Subnets": [
          {"Ref": "Subnet1"},
          {"Ref": "Subnet2"}
        ],
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
```

```
        ],
        "Type": "application",
        "IpAddressType": "ipv4"
      }
    },
    "ALBListenerProdTraffic": {
      "Type": "AWS::ElasticLoadBalancingV2::Listener",
      "Properties": {
        "DefaultActions": [
          {
            "Type": "forward",
            "ForwardConfig": {
              "TargetGroups": [
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
                  "Weight": 1
                }
              ]
            }
          }
        ],
        "LoadBalancerArn": {"Ref": "ExampleALB"},
        "Port": 80,
        "Protocol": "HTTP"
      }
    },
    "ALBListenerProdRule": {
      "Type": "AWS::ElasticLoadBalancingV2::ListenerRule",
      "Properties": {
        "Actions": [
          {
            "Type": "forward",
            "ForwardConfig": {
              "TargetGroups": [
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
                  "Weight": 1
                }
              ]
            }
          }
        ],
        "Conditions": [
          {
```

```
            "Field": "http-header",
            "HttpHeaderConfig": {
              "HttpHeaderName": "User-Agent",
              "Values": [
                "Mozilla"
              ]
            }
          }
        ],
        "ListenerArn": {"Ref": "ALBListenerProdTraffic"},
        "Priority": 1
      }
    },
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "",
              "Effect": "Allow",
              "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
        ]
      }
    },
    "BlueTaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
        "ContainerDefinitions": [
          {
            "Name": "DemoApp",
            "Image": "nginxdemos/hello:latest",
            "Essential": true,
            "PortMappings": [
```

```
            {
              "HostPort": 80,
              "Protocol": "tcp",
              "ContainerPort": 80
            }
          ]
        }
      ],
      "RequiresCompatibilities": [
        "FARGATE"
      ],
      "NetworkMode": "awsvpc",
      "Cpu": "256",
      "Memory": "512",
      "Family": "ecs-demo"
    }
  },
  "ECSDemoCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {}
  },
  "ECSDemoService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
      "Cluster": {"Ref": "ECSDemoCluster"},
      "DesiredCount": 1,
      "DeploymentController": {
        "Type": "EXTERNAL"
      }
    }
  },
  "BlueTaskSet": {
    "Type": "AWS::ECS::TaskSet",
    "Properties": {
      "Cluster": {"Ref": "ECSDemoCluster"},
      "LaunchType": "FARGATE",
      "NetworkConfiguration": {
        "AwsVpcConfiguration": {
          "AssignPublicIp": "ENABLED",
          "SecurityGroups": [
            {"Ref": "ExampleSecurityGroup"}
          ],
          "Subnets": [
            {"Ref": "Subnet1"},
```

```
              {"Ref": "Subnet2"}
            ]
          }
        },
        "PlatformVersion": "1.4.0",
        "Scale": {
          "Unit": "PERCENT",
          "Value": 100
        },
        "Service": {"Ref": "ECSDemoService"},
        "TaskDefinition": {"Ref": "BlueTaskDefinition"},
        "LoadBalancers": [
          {
            "ContainerName": "DemoApp",
            "ContainerPort": 80,
            "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"}
          }
        ]
      }
    },
    "PrimaryTaskSet": {
      "Type": "AWS::ECS::PrimaryTaskSet",
      "Properties": {
        "Cluster": {"Ref": "ECSDemoCluster"},
        "Service": {"Ref": "ECSDemoService"},
        "TaskSetId": {"Fn::GetAtt": ["BlueTaskSet", "Id"]}
      }
    }
  }
}
```

Langkah migrasi

Hapus CodeDeploy sumber daya -spesifik

Anda tidak lagi membutuhkan properti berikut:

- `AWS::CodeDeployBlueGreen`Transformasi
- `CodeDeployBlueGreenHook`Kaitnya
- `GreenTaskSet`Sumber daya `GreenTaskDefinition` dan sumber daya (ini akan dikelola oleh Amazon ECS)
- Sumber `PrimaryTaskSet` daya (Amazon ECS akan mengelola set tugas secara internal)

Konfigurasikan ulang pendengar penyeimbang beban

Ubah `ALBListenerProdTraffic` sumber daya untuk menggunakan tindakan maju dengan dua kelompok target:

```
{
  "DefaultActions": [
    {
      "Type": "forward",
      "ForwardConfig": {
        "TargetGroups": [
          {
            "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
            "Weight": 1
          },
          {
            "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
            "Weight": 0
          }
        ]
      }
    }
  ]
}
```

Perbarui properti penerapan

Perbarui dan tambahkan yang berikut ini:

- Ubah `DeploymentController` properti dari `EXTERNAL` ke`ECS`.
- Tambahkan `Strategy` properti dan atur ke BLUE_GREEN.
- Tambahkan `BakeTimeInMinutes` properti.

```
{
  "DeploymentConfiguration": {
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 100,
    "DeploymentCircuitBreaker": {
      "Enable": true,
      "Rollback": true
    },
    "BakeTimeInMinutes": 5,
```

```
    "Strategy": "BLUE_GREEN"
  }
}
```

- Tambahkan konfigurasi penyeimbang beban ke layanan:

```
{
  "LoadBalancers": [
    {
      "ContainerName": "DemoApp",
      "ContainerPort": 80,
      "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
      "AdvancedConfiguration": {
        "AlternateTargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
        "ProductionListenerRule": {"Ref": "ALBListenerProdRule"},
        "RoleArn": {"Fn::GetAtt": ["ECSInfrastructureRoleForLoadBalancers", "Arn"]}
      }
    }
  ]
}
```

- Tambahkan referensi definisi tugas ke layanan:

```
{
  "TaskDefinition": {"Ref": "BlueTaskDefinition"}
}
```

Buat ECSInfrastructure RolePolicyForLoadBalancers peran Amazon

Tambahkan peran IAM baru yang memungkinkan Amazon ECS mengelola sumber daya penyeimbang beban:

Ganti *user-input* dengan nilai Anda.

```
{
  "ECSInfrastructureRoleForLoadBalancers": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
```

```
            "Sid": "AllowAccessToECSForInfrastructureManagement",
            "Effect": "Allow",
            "Principal": {
              "Service": "ecs.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/AmazonECSInfrastructureRolePolicyForLoadBalancers"
      ]
    }
  }
}
```

Rekomendasi pengujian

1. Menerapkan template yang dimigrasi ke lingkungan non-produksi.
2. Verifikasi bahwa layanan diterapkan dengan benar dengan konfigurasi awal.
3. Uji penerapan dengan memperbarui definisi tugas dan mengamati proses penerapan biru/hijau.
4. Verifikasi bahwa lalu lintas bergeser dengan benar antara penerapan biru dan hijau.
5. Uji fungsionalitas rollback dengan memaksa kegagalan penerapan.

Template setelah migrasi

Template akhir

Ini adalah AWS CloudFormation template lengkap menggunakan blue/green penyebaran Amazon ECS:

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "Vpc": {
      "Type": "AWS::EC2::VPC::Id"
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet::Id"
    },
    "Subnet2": {
```

```
      "Type": "AWS::EC2::Subnet::Id"
    }
  },
  "Resources": {
    "ExampleSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Security group for ec2 access",
        "VpcId": {"Ref": "Vpc"},
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 8080,
            "ToPort": 8080,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    },
    "ALBTargetGroupBlue": {
      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
          "HttpCode": "200"
        },
        "Port": 80,
```

```
        "Protocol": "HTTP",
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
        "VpcId": {"Ref": "Vpc"}
      }
    },
    "ALBTargetGroupGreen": {
      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
          "HttpCode": "200"
        },
        "Port": 80,
        "Protocol": "HTTP",
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
        "VpcId": {"Ref": "Vpc"}
      }
    },
    "ExampleALB": {
      "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties": {
        "Scheme": "internet-facing",
        "SecurityGroups": [
          {"Ref": "ExampleSecurityGroup"}
        ],
```

```
        "Subnets": [
          {"Ref": "Subnet1"},
          {"Ref": "Subnet2"}
        ],
        "Tags": [
          {
            "Key": "Group",
            "Value": "Example"
          }
        ],
        "Type": "application",
        "IpAddressType": "ipv4"
      }
    },
    "ALBListenerProdTraffic": {
      "Type": "AWS::ElasticLoadBalancingV2::Listener",
      "Properties": {
        "DefaultActions": [
          {
            "Type": "forward",
            "ForwardConfig": {
              "TargetGroups": [
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
                  "Weight": 1
                },
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
                  "Weight": 0
                }
              ]
            }
          }
        ],
        "LoadBalancerArn": {"Ref": "ExampleALB"},
        "Port": 80,
        "Protocol": "HTTP"
      }
    },
    "ALBListenerProdRule": {
      "Type": "AWS::ElasticLoadBalancingV2::ListenerRule",
      "Properties": {
        "Actions": [
          {
```

```
            "Type": "forward",
            "ForwardConfig": {
              "TargetGroups": [
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
                  "Weight": 1
                },
                {
                  "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
                  "Weight": 0
                }
              ]
            }
          }
        ],
        "Conditions": [
          {
            "Field": "http-header",
            "HttpHeaderConfig": {
              "HttpHeaderName": "User-Agent",
              "Values": [
                "Mozilla"
              ]
            }
          }
        ],
        "ListenerArn": {"Ref": "ALBListenerProdTraffic"},
        "Priority": 1
      }
    },
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "",
              "Effect": "Allow",
              "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
```

```
          ]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
        ]
      }
    },
    "ECSInfrastructureRoleForLoadBalancers": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AllowAccessToECSForInfrastructureManagement",
              "Effect": "Allow",
              "Principal": {
                "Service": "ecs.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/AmazonECSInfrastructureRolePolicyForLoadBalancers"
        ]
      }
    },
    "BlueTaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
        "ContainerDefinitions": [
          {
            "Name": "DemoApp",
            "Image": "nginxdemos/hello:latest",
            "Essential": true,
            "PortMappings": [
              {
                "HostPort": 80,
                "Protocol": "tcp",
                "ContainerPort": 80
              }
            ]
```

```
        }
      ],
      "RequiresCompatibilities": [
        "FARGATE"
      ],
      "NetworkMode": "awsvpc",
      "Cpu": "256",
      "Memory": "512",
      "Family": "ecs-demo"
    }
  },
  "ECSDemoCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {}
  },
  "ECSDemoService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
      "Cluster": {"Ref": "ECSDemoCluster"},
      "DesiredCount": 1,
      "DeploymentController": {
        "Type": "ECS"
      },
      "DeploymentConfiguration": {
        "MaximumPercent": 200,
        "MinimumHealthyPercent": 100,
        "DeploymentCircuitBreaker": {
          "Enable": true,
          "Rollback": true
        },
        "BakeTimeInMinutes": 5,
        "Strategy": "BLUE_GREEN"
      },
      "NetworkConfiguration": {
        "AwsvpcConfiguration": {
          "AssignPublicIp": "ENABLED",
          "SecurityGroups": [
            {"Ref": "ExampleSecurityGroup"}
          ],
          "Subnets": [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
          ]
        }
```

```
        },
        "LaunchType": "FARGATE",
        "PlatformVersion": "1.4.0",
        "TaskDefinition": {"Ref": "BlueTaskDefinition"},
        "LoadBalancers": [
          {
            "ContainerName": "DemoApp",
            "ContainerPort": 80,
            "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
            "AdvancedConfiguration": {
              "AlternateTargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
              "ProductionListenerRule": {"Ref": "ALBListenerProdRule"},
              "RoleArn": {"Fn::GetAtt": ["ECSInfrastructureRoleForLoadBalancers",
 "Arn"]}
            }
          }
        ]
      }
    }
  }
}
```

Bermigrasi dari CodeDeploy biru/hijau ke penyebaran layanan pembaruan bergulir Amazon ECS

Anda dapat memigrasikan penerapan layanan dari penerapan CodeDeploy biru/hijau ke penerapan pembaruan bergulir Amazon ECS. Ini menjauhkan Anda dari CodeDeploy ketergantungan untuk menggunakan penerapan terintegrasi.

Penjadwal layanan Amazon ECS menggantikan tugas yang sedang berjalan dengan tugas baru. Jumlah tugas yang ditambahkan atau dihapus Amazon ECS dari layanan selama pembaruan bergulir dikendalikan oleh konfigurasi penyebaran layanan.

Prasyarat

Lakukan operasi berikut sebelum Anda memulai blue/green penerapan.

1. Anda tidak lagi membutuhkan peran Amazon ECS CodeDeploy IAM.
2. Matikan CodeDeploy otomatisasi. Untuk informasi selengkapnya, lihat Bekerja dengan grup penerapan CodeDeploy di Panduan CodeDeploy Pengguna.
3. Verifikasi bahwa tidak ada penerapan layanan yang sedang berlangsung untuk layanan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

Untuk informasi selengkapnya tentang memperbarui pengontrol penerapan layanan, lihatPerbarui parameter layanan Amazon ECS.

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.

   Halaman detail cluster ditampilkan.
3. Dari tab Layanan, pilih layanan.

   Halaman detail layanan ditampilkan.
4. Di spanduk, pilih Migrasi.

   Halaman konfigurasi Update deployment ditampilkan.
5. Perluas opsi Deployment, lalu tentukan parameter berikut.

   a. Untuk jenis pengontrol Deployment, pilih ECS.

   b. Untuk strategi Deployment, pilih Pembaruan bergulir.

   c. Untuk tugas yang menjalankan Min, masukkan batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Untuk informasi selengkapnya, lihat Konfigurasi penerapan.

   d. Untuk tugas yang berjalan Max, masukkan batas atas jumlah tugas dalam layanan yang diizinkan dalam `PENDING` status `RUNNING` atau selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat).
6. Perluas Load Balancing, lalu konfigurasikan yang berikut ini:

   a. Untuk Peran, pilih peran yang Anda buat dalam prasyarat dengan izin. blue/green

      Untuk informasi selengkapnya, lihat Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green .

   b. Untuk Listener, pilih pendengar produksi dari penerapan CodeDeploy biru/hijau Anda.

   c. Untuk grup Target, pilih grup target produksi dari penyebaran CodeDeploy biru/hijau Anda.
7. Pilih Perbarui.

Langkah selanjutnya

Anda harus memperbarui layanan agar perubahan diterapkan. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Perbarui strategi penyebaran Amazon ECS

Amazon ECS mendukung beberapa strategi penerapan untuk memperbarui layanan Anda. Anda dapat bermigrasi di antara strategi ini berdasarkan persyaratan aplikasi Anda. Topik ini menjelaskan cara bermigrasi antara penerapan bergulir dan penerapan. blue/green

Memahami strategi penyebaran Amazon ECS

Sebelum bermigrasi antar strategi penerapan, penting untuk memahami cara kerja setiap strategi dan perbedaan utamanya:

Penerapan bergulir

Dalam penerapan bergulir, Amazon ECS menggantikan versi aplikasi yang sedang berjalan dengan versi baru. Penjadwal layanan menggunakan parameter persentase sehat minimum dan maksimum untuk menentukan strategi penerapan.

Penyebaran bergulir lebih sederhana untuk diatur tetapi memberikan kontrol yang lebih sedikit atas proses penyebaran dan perutean lalu lintas.

Penerapan biru/hijau

Dalam blue/green penerapan, Amazon ECS membuat versi baru layanan Anda (hijau) di samping versi yang ada (biru). Ini memungkinkan Anda untuk memverifikasi versi baru sebelum merutekan lalu lintas produksi ke sana.

Penerapan biru/hijau memberikan kontrol lebih besar atas proses penyebaran, termasuk perpindahan lalu lintas, pengujian, dan kemampuan rollback.

Praktik terbaik

Ikuti praktik terbaik ini saat bermigrasi antar strategi penerapan:

- Uji di lingkungan non-produksi: Selalu uji pembaruan di lingkungan non-produksi sebelum menerapkan perubahan pada layanan produksi.
- Rencana untuk rollback: Miliki rencana rollback jika pembaruan tidak berfungsi seperti yang diharapkan.

- Monitor selama transisi: Pantau layanan Anda dengan cermat selama dan setelah migrasi untuk memastikannya terus beroperasi dengan benar.
- Perbarui dokumentasi: Perbarui dokumentasi penerapan Anda untuk mencerminkan strategi penerapan baru.
- Pertimbangkan dampak lalu lintas: Pahami bagaimana pembaruan dapat memengaruhi lalu lintas ke layanan Anda dan rencanakan dengan tepat.

Memperbarui strategi penerapan dari pembaruan bergulir ke Amazon ECS biru/hijau

Anda dapat bermigrasi dari penerapan pembaruan bergulir ke penerapan Amazon ECS blue/green saat Anda ingin membuat dan menguji perubahan layanan sebelum menerapkannya di lingkungan produksi.

Prasyarat

Sebelum memigrasikan layanan Anda dari bergulir ke blue/green penerapan, pastikan Anda memiliki yang berikut:

- Tunggu hingga penerapan saat ini selesai.
- Layanan Amazon ECS yang ada menggunakan strategi penyebaran bergulir.
- Jika Anda memiliki beberapa revisi layanan yang melayani lalu lintas, Amazon ECS mencoba mengkonsolidasikan lalu lintas ke satu revisi selama migrasi. Jika gagal, Anda mungkin perlu memperbarui layanan secara manual untuk menggunakan satu revisi sebelum bermigrasi.
- Konfigurasikan izin yang sesuai.
  - Untuk informasi tentang izin Elastic Load Balancing, lihat. Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban
  - Untuk informasi tentang izin Lambda, lihat. Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green
- Tergantung pada konfigurasi, Anda perlu melakukan salah satu dari yang berikut:
  - Jika layanan Anda menggunakan Elastic Load Balancing, perbarui layanan Anda dengan `AdvancedConfiguration` yang baru dan mulai penerapan bergulir.
  - Jika layanan Anda menggunakan Service Connect, perbarui layanan Anda dan mulai penerapan bergulir.
  - Jika layanan Anda menggunakan Elastic Load Balancing dan Service Connect, lakukan kedua langkah di atas (Anda dapat menggunakan satu UpdateService permintaan).

- Jika layanan Anda tidak menggunakan salah satu dari yang di atas, maka tidak diperlukan operasi tambahan.

- blue/green Penyebaran Amazon ECS mengharuskan layanan Anda menggunakan salah satu fitur berikut. Konfigurasikan sumber daya yang sesuai.
  - Application Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Application Load Balancer untuk penerapan blue/green
  - Network Load Balancer - Untuk informasi lebih lanjut, lihat. Sumber daya Network Load Balancer untuk penerapan biru/hijau Amazon ECS
  - Service Connect - Untuk informasi selengkapnya, lihatSumber daya Service Connect untuk penerapan biru/hijau Amazon ECS.

Prosedur

1. Buka konsol Amazon ECS dihttps://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster yang berisi layanan yang ingin Anda migrasikan.

   Halaman detail Cluster ditampilkan.
4. Pada halaman detail Cluster, pilih tab Layanan.
5. Pilih layanan, lalu pilih Perbarui.

   Halaman layanan Update ditampilkan
6. Perluas opsi Deployment, lalu lakukan hal berikut:
7. Untuk strategi Deployment, pilih Biru/hijau.
8. Konfigurasikan pengaturan blue/green penerapan:

   a. Untuk waktu Panggang, masukkan jumlah menit revisi layanan biru dan hijau akan berjalan secara bersamaan sebelum revisi biru dihentikan.

      Ini memungkinkan waktu untuk verifikasi dan pengujian.

   b. (Opsional) Konfigurasikan fungsi Lambda untuk berjalan pada tahap penerapan tertentu. Di bawah kait siklus hidup Deployment, konfigurasikan fungsi Lambda untuk tahapan berikut:

      - Pre scale up: Berjalan sebelum meningkatkan revisi layanan hijau
      - Pasca peningkatan skala: Berjalan setelah meningkatkan revisi layanan hijau

- Uji pergeseran lalu lintas: Berjalan selama perutean lalu lintas uji ke revisi layanan hijau
- Pergeseran lalu lintas pasca uji: Berjalan setelah lalu lintas uji diarahkan ke revisi layanan hijau
- Pergeseran lalu lintas produksi: Berjalan selama perutean lalu lintas produksi ke revisi layanan hijau
- Pergeseran lalu lintas pasca produksi: Berjalan setelah lalu lintas produksi diarahkan ke revisi layanan hijau

Untuk menambahkan kait siklus hidup:

a. Pilih Tambahkan.
b. Untuk fungsi Lambda, masukkan nama fungsi atau ARN.
c. Untuk Peran, pilih peran IAM yang memiliki izin untuk menjalankan fungsi Lambda.
d. Untuk tahapan Siklus Hidup, pilih tahapan saat fungsi Lambda harus dijalankan.
e. Opsional: Untuk detail Hook, masukkan pasangan nilai kunci untuk memberikan informasi tambahan ke hook.

9. Konfigurasikan pengaturan penyeimbang beban:

   a. Di bawah Load balancing, verifikasi bahwa layanan Anda dikonfigurasi untuk menggunakan penyeimbang beban.
   b. Untuk grup Target, pilih kelompok target utama untuk lingkungan produksi (biru) Anda.
   c. Untuk grup target Alternatif, pilih grup target untuk lingkungan pengujian (hijau) Anda.
   d. Untuk aturan Production listener, pilih aturan listener untuk merutekan lalu lintas produksi.
   e. Opsional: Untuk aturan Test listener, pilih aturan listener untuk merutekan lalu lintas pengujian ke lingkungan hijau Anda.
   f. Untuk Peran, pilih peran IAM yang memungkinkan Amazon ECS mengelola penyeimbang beban Anda.

10. Tinjau perubahan konfigurasi Anda, lalu pilih Perbarui.

Langkah selanjutnya

- Perbarui layanan untuk memulai penyebaran. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.
- Pantau proses penyebaran untuk memastikannya mengikuti blue/green pola:

- Revisi layanan hijau dibuat dan ditingkatkan
- Lalu lintas uji diarahkan ke revisi hijau (jika dikonfigurasi)
- Lalu lintas produksi dialihkan ke revisi hijau
- Setelah waktu memanggang, revisi biru dihentikan

Memperbarui strategi penerapan dari Amazon ECS blue/green ke pembaruan bergulir

Anda dapat memigrasikan blue/green penerapan ke penerapan pembaruan bergulir.

Ingatlah pertimbangan berikut saat bermigrasi ke penerapan bergulir:

- Penanganan lalu lintas: Dengan penerapan bergulir, tugas baru mulai menerima lalu lintas segera setelah mereka lulus pemeriksaan kesehatan. Tidak ada fase pengujian terpisah seperti blue/green penerapan.
- Efisiensi sumber daya: Penerapan bergulir biasanya menggunakan sumber daya yang lebih sedikit daripada blue/green penerapan karena mereka mengganti tugas secara bertahap daripada menciptakan lingkungan duplikat yang lengkap.
- Kompleksitas rollback: Penerapan bergulir membuat rollback lebih kompleks dibandingkan dengan penerapan. blue/green Jika Anda perlu memutar kembali, Anda harus memulai penerapan baru dengan definisi tugas sebelumnya.
- Kecepatan penyebaran: Penyebaran bergulir mungkin membutuhkan waktu lebih lama untuk diselesaikan daripada blue/green penerapan, terutama untuk layanan dengan banyak tugas.
- Konfigurasi penyeimbang beban: Konfigurasi penyeimbang beban Anda yang ada akan terus bekerja dengan penerapan bergulir, tetapi perilaku pergeseran lalu lintas akan berbeda.

Prasyarat

Sebelum memigrasikan layanan Anda dari blue/green ke penerapan bergulir, pastikan Anda memiliki yang berikut:

- Layanan Amazon ECS yang ada menggunakan strategi blue/green penyebaran
- Tidak ada penerapan yang sedang berlangsung untuk layanan (tunggu hingga penerapan saat ini selesai)
- Pemahaman yang jelas tentang bagaimana layanan Anda akan berperilaku dengan penerapan bergulir

> **Note**
>
> Anda tidak dapat memigrasikan layanan ke penerapan bergulir jika layanan tersebut memiliki penerapan yang sedang berlangsung. Tunggu hingga penerapan saat ini selesai sebelum melanjutkan.

Prosedur migrasi

Ikuti langkah-langkah berikut untuk memigrasikan layanan Amazon ECS Anda dari blue/green ke penerapan bergulir:

1. Buka konsol Amazon ECS dihttps://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster yang berisi layanan yang ingin Anda migrasikan.
4. Pada halaman detail Cluster, pilih tab Layanan.
5. Pilih layanan yang ingin Anda migrasikan, lalu pilih Perbarui.
6. Pada halaman Update Service, navigasikan ke bagian Opsi Deployment dan perluas jika perlu.
7. Untuk strategi Deployment, pilih Pembaruan bergulir.
8. Konfigurasikan pengaturan penerapan bergulir:
    a. Untuk Persentase sehat minimum, masukkan persentase minimum tugas yang harus tetap dalam `RUNNING` keadaan selama penerapan. Nilai ini ditentukan sebagai persentase dari jumlah tugas yang diinginkan untuk layanan.
    b. Untuk persen Maksimum, masukkan persentase maksimum tugas yang diizinkan di `PENDING` negara bagian `RUNNING` atau selama penerapan. Nilai ini ditentukan sebagai persentase dari jumlah tugas yang diinginkan untuk layanan.
9. Opsional: Di bawah deteksi kegagalan Deployment, konfigurasikan cara Amazon ECS mendeteksi dan menangani kegagalan penerapan:
    a. Untuk mengaktifkan pemutus sirkuit penyebaran, pilih Gunakan pemutus sirkuit penyebaran.
    b. Untuk secara otomatis memutar kembali penerapan yang gagal, pilih Rollback on failure.
10. Tinjau perubahan konfigurasi Anda, lalu pilih Perbarui untuk menyimpan perubahan Anda dan memigrasikan layanan ke penerapan bergulir.

Amazon ECS akan memperbarui konfigurasi layanan Anda untuk menggunakan strategi penerapan bergulir. Lain kali Anda memperbarui layanan Anda, itu akan menggunakan proses penyebaran bergulir.

> **Note**
>
> Saat Anda bermigrasi dari blue/green ke penerapan bergulir, Amazon ECS menangani transisi dengan:
>
> 1. Mengidentifikasi revisi layanan aktif saat ini yang melayani lalu lintas.
> 2. Mempertahankan konfigurasi penyeimbang beban yang ada tetapi mengubah cara penerapan baru ditangani.
> 3. Mempersiapkan layanan untuk penerapan bergulir di masa depan.

Langkah selanjutnya

- Perbarui layanan untuk memulai penyebaran. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Memecahkan masalah pembaruan strategi penerapan Amazon ECS

Bagian ini memberikan solusi untuk masalah umum yang mungkin Anda temui saat memigrasikan strategi penerapan.

Beberapa revisi layanan atau set tugas

Masalah berikut terkait dengan memiliki beberapa revisi layanan untuk penerapan.

Beberapa set tugas saat memperbarui pengontrol penyebaran ECS

Pesan kesalahan: `Updating the deployment controller is not supported when there are multiple tasksets in the service. Please ensure your service has only one taskset and try again.`

Solusi: Kesalahan ini terjadi ketika mencoba mengubah jenis pengontrol penerapan layanan dengan beberapa set tugas aktif. Untuk mengatasi masalah ini untuk pengontrol `CODE_DEPLOY` atau `EXTERNAL` penerapan:

1. Periksa set tugas saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].taskSets"
```

2. Tunggu hingga penerapan yang sedang berlangsung selesai.
3. Memaksa penerapan baru untuk membersihkan set tugas:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --force-new-deployment
```

4. Jika perlu, hapus set tugas tambahan secara manual:

```
aws ecs delete-task-set --cluster your-cluster-name --service your-service-name
 --task-set task-set-id
```

5. Setelah hanya satu set tugas yang tersisa, coba lagi perbarui pengontrol penerapan.

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Set tugas utama tidak ada saat memperbarui ECS pengontrol penerapan

Pesan kesalahan: `Updating the deployment controller requires a primary taskset in the service. Please ensure your service has a primary taskset and try again.`

Solusi: Kesalahan ini terjadi ketika mencoba mengubah jenis pengontrol penerapan layanan yang tidak memiliki set tugas utama. Untuk menyelesaikan masalah ini:

1. Verifikasi status layanan dan set tugas.). Jika set tugas ada dalam layanan, itu harus ditandai sebagai`ACTIVE`.

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].taskSets[*].[status,id]
```

Jika tidak ada set tugas di `ACTIVE` negara bagian, migrasi penerapan. Untuk informasi selengkapnya, lihat Pendekatan migrasi.

2. Jika layanan tidak memiliki tugas yang berjalan, gunakan setidaknya satu tugas dengan memperbarui layanan:

```
aws ecs update-service-primary-task-set --cluster your-cluster-name --
service your-service-name --primary-task-set your-taskset-id
```

Ini akan menandai tugas (sebelumnyaACTIVE) yang ditetapkan dalam layanan sebagai PRIMARY status.

3. Tunggu tugas mencapai status berjalan stabil. Anda dapat memeriksa statusnya dengan:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-
name --query "services[0].deployments"
```

4. Setelah layanan memiliki tugas utama yang ditetapkan dengan tugas yang sedang berjalan, coba lagi memperbarui pengontrol penerapan.

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Ketidakcocokan antara jenis deteksi kegagalan penerapan dan pengontrol penerapan

Masalah berikut terkait dengan ketidakcocokan antara jenis deteksi kegagalan penerapan dan pengontrol penerapan.

Pemutus sirkuit penyebaran dengan pengontrol non-ECS

Pesan kesalahan: `Deployment circuit breaker feature is only supported with ECS deployment controller. Update to ECS deployment controller and try again.`

Solusi: Kesalahan ini terjadi ketika mencoba mengaktifkan fitur pemutus sirkuit penyebaran pada layanan yang tidak menggunakan pengontrol penyebaran. ECS Pemutus sirkuit penyebaran hanya kompatibel dengan pengontrol ECS penyebaran.

1. Periksa pengontrol penerapan layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-
name --query "services[0].deploymentController"
```

2. Perbarui layanan Anda untuk menggunakan pengontrol ECS penerapan:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

3. Setelah layanan menggunakan pengontrol ECS penyebaran, aktifkan pemutus sirkuit penyebaran:

```
aws ecs update-service --cluster your-cluster-name --
service your-service-name --deployment-configuration
 "deploymentCircuitBreaker={enable=true,rollback=true}"
```

Untuk informasi selengkapnya, lihat the section called “ Bagaimana pemutus sirkuit penyebaran mendeteksi kegagalan”.

Rollback berbasis alarm dengan pengontrol non-ECS

Pesan kesalahan: `Alarm based rollback feature is only supported with ECS deployment controller. Update to ECS deployment controller and try again.`

Solusi: Kesalahan ini terjadi ketika mencoba mengkonfigurasi rollback berbasis alarm pada layanan yang tidak menggunakan pengontrol penerapan. ECS Fitur rollback berbasis alarm hanya kompatibel dengan pengontrol penerapan. ECS

1. Periksa pengontrol penerapan layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-
name --query "services[0].deploymentController"
```

2. Perbarui layanan Anda untuk menggunakan pengontrol ECS penerapan:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

3. Setelah layanan menggunakan pengontrol ECS penyebaran, konfigurasikan rollback berbasis alarm:

```
aws ecs update-service --cluster your-cluster-name --services your-
service-name --deployment-configuration "alarms={alarmNames=[your-alarm-
name],enable=true,rollback=true}"
```

Untuk informasi selengkapnya, lihat the section called “Bagaimana CloudWatch alarm mendeteksi kegagalan penerapan”.

Ketidakcocokan antara Service Connect dan pengontrol penerapan

Masalah berikut terkait dengan ketidakcocokan antara Service Connect dan pengontrol penerapan.

`EXTERNAL`pengontrol dengan Service Connect

Pesan kesalahan: `The EXTERNAL deployment controller type is not supported for services using Service Connect.`

Solusi: Kesalahan ini terjadi saat mencoba menggunakan pengontrol `EXTERNAL` penerapan dengan layanan yang mengaktifkan Service Connect. `EXTERNAL`Kontroler tidak kompatibel dengan Service Connect.

1. Periksa apakah layanan Anda mengaktifkan Service Connect:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].serviceConnectConfiguration"
```

2. Jika Anda perlu menggunakan pengontrol `EXTERNAL` penyebaran, nonaktifkan Service Connect dengan memperbarui layanan Anda:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --service-connect-configuration "{}"
```

3. Atau, jika Anda harus menggunakan Service Connect, gunakan pengontrol `ECS` penerapan sebagai gantinya:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Service Connect dengan pengontrol non-ECS

Pesan kesalahan: `Service Connect feature is only supported with ECS (rolling update) deployment controller. Update to ECS deployment controller and try again.`

Solusi: Kesalahan ini terjadi ketika mencoba mengaktifkan Service Connect pada yang tidak menggunakan pengontrol ECS penerapan. Fitur Service Connect hanya kompatibel dengan pengontrol ECS penerapan.

1. Periksa pengontrol penerapan layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].deploymentController"
```

2. Perbarui layanan Anda untuk menggunakan pengontrol penyebaran ECS:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

3. Setelah layanan menggunakan pengontrol penyebaran ECS, aktifkan Service Connect:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --service-connect-configuration "enabled=true,namespace=your-namespace"
```

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Ketidakcocokan antara jenis pengontrol dan strategi penjadwalan

Masalah berikut terkait dengan ketidakcocokan antara jenis pengontrol dan strategi penjadwalan.

`CODE_DEPLOY`controller dengan strategi `DAEMON` penjadwalan

Pesan kesalahan: `The CODE_DEPLOY deployment controller type is not supported for services using the DAEMON scheduling strategy.`

Solusi: Kesalahan ini terjadi ketika mencoba menggunakan pengontrol penyebaran CODE_DEPLOY dengan layanan yang menggunakan strategi penjadwalan. `DAEMON` `CODE_DEPLOY`Pengontrol hanya kompatibel dengan strategi `REPLICA` penjadwalan.

1. Periksa strategi penjadwalan layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].schedulingStrategy"
```

2. Jika Anda memerlukan blue/green penerapan, ubah layanan Anda untuk menggunakan strategi `REPLICA` penjadwalan:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --scheduling-strategy REPLICA
```

3. Atau, jika Anda harus menggunakan strategi `DAEMON` penjadwalan, gunakan pengontrol `ECS` penerapan sebagai gantinya:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Pengontrol EKSTERNAL dengan strategi penjadwalan DAEMON

Pesan kesalahan: `The EXTERNAL deployment controller type is not supported for services using the DAEMON scheduling strategy.`

Solusi: Kesalahan ini terjadi ketika mencoba menggunakan pengontrol penyebaran EKSTERNAL dengan layanan ECS yang menggunakan strategi penjadwalan DAEMON. Pengontrol EKSTERNAL hanya kompatibel dengan strategi penjadwalan REPLICA.

1. Periksa strategi penjadwalan layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].schedulingStrategy"
```

2. Jika Anda perlu menggunakan pengontrol `EXTERNAL` penerapan, ubah layanan Anda untuk menggunakan strategi `REPLICA` penjadwalan:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --scheduling-strategy REPLICA
```

3. Atau, jika Anda harus menggunakan strategi `DAEMON` penjadwalan, gunakan pengontrol `ECS` penerapan sebagai gantinya:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --deployment-controller type=ECS
```

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Registri layanan dengan tipe peluncuran eksternal

Pesan kesalahan: `Service registries are not supported for external launch type.`

Solusi: Kesalahan ini terjadi ketika mencoba mengonfigurasi penemuan layanan (pendaftar layanan) untuk layanan yang menggunakan jenis peluncuran. `EXTERNAL` Penemuan layanan tidak kompatibel dengan jenis `EXTERNAL` peluncuran.

1. Periksa jenis peluncuran layanan Anda saat ini:

```
aws ecs describe-services --cluster your-cluster-name --services your-service-
name --query "services[0].launchType"
```

2. Jika Anda memerlukan penemuan layanan, ubah layanan Anda untuk menggunakan jenis `EC2` atau `FARGATE` peluncuran:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --launch-type FARGATE
```

3. Atau, jika Anda harus menggunakan jenis `EXTERNAL` peluncuran, hapus konfigurasi registri layanan:

```
aws ecs update-service --cluster your-cluster-name --service your-service-name
 --service-registries "[]"
```

Untuk informasi selengkapnya, lihat the section called “Pengontrol dan strategi penyebaran layanan”.

Kembalikan pembaruan pengontrol penerapan

Jika Anda memutuskan ingin kembali ke pengontrol penerapan sebelumnya, Anda dapat melakukan salah satu hal berikut:

- Jika Anda menggunakan AWS CloudFormation, Anda dapat menggunakan template sebelumnya untuk membuat tumpukan baru. Untuk informasi selengkapnya, lihat Membuat tumpukan dari Panduan AWS CloudFormation Pengguna.
- Jika Anda menggunakan konsol Amazon ECS, atau AWS CLI, Anda dapat memperbarui layanan. Untuk informasi selengkapnya, lihat the section called “Memperbarui layanan”.

  Jika Anda menggunakan perintah update-service, gunakan `--deployment-controller` opsi dan atur ke controller deployment sebelumnya.

Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS

Penyebaran layanan memberikan pandangan komprehensif tentang penerapan Anda. Penyebaran layanan memberikan informasi berikut tentang layanan:

- Konfigurasi beban kerja yang saat ini diterapkan (revisi layanan sumber)
- Konfigurasi beban kerja yang sedang digunakan (revisi layanan target)
- Status penyebaran
- Jumlah tugas gagal yang terdeteksi oleh pemutusan sirkuit
- CloudWatch Alarm yang ada di alarm
- Ketika penyebaran layanan dimulai dan selesai
- Rincian rollback jika terjadi

Untuk informasi tentang properti penyebaran layanan, lihatProperti yang disertakan dalam penyebaran layanan Amazon ECS.

Penyebaran layanan hanya-baca dan masing-masing memiliki ID unik.

Ada tiga tahap penyebaran layanan:

Stage	Definisi	Negara terkait
Tertunda	Penyebaran layanan telah dibuat, tetapi belum dimulai	MENUNGGU
Berkelanjutan	Penyebaran layanan sedang berlangsung	• IN_PROGRESS • STOP_REQUEST • ROLLBACK_REQUEST • ROLLBACK_IN_PROGRESS
Selesai	Penyebaran layanan telah selesai (berhasil atau tidak berhasil)	• SUKSES • DIHENTIKAN • ROLLBACK_SUCCESSFUL • ROLLBACK_FAILED

Anda menggunakan penerapan layanan untuk memahami siklus hidup layanan Anda dan untuk menentukan apakah ada tindakan yang perlu Anda lakukan. Misalnya, jika rollback terjadi, Anda mungkin perlu menyelidiki penyebaran layanan dan melihat peristiwa layanan.

Anda dapat melihat riwayat 90 hari terbaru untuk penerapan yang dibuat pada atau setelah 25 Oktober 2024 dengan menggunakan konsol, API, dan. AWS CLI

Anda dapat menghentikan penerapan yang belum selesai. Untuk informasi selengkapnya, lihat Menghentikan penyebaran layanan Amazon ECS.

Siklus hidup penyebaran layanan

Amazon ECS membuat penyebaran layanan baru secara otomatis ketika salah satu tindakan berikut terjadi:

- Seorang pengguna membuat layanan.
- Pengguna memperbarui layanan dan menggunakan opsi force new deployment.
- Pengguna memperbarui satu atau beberapa properti layanan yang memerlukan penerapan.

Saat penerapan sedang berlangsung, Amazon ECS memperbarui properti penyebaran layanan berikut untuk mencerminkan kemajuan penerapan layanan:

- Negara
- Jumlah tugas yang sedang berjalan

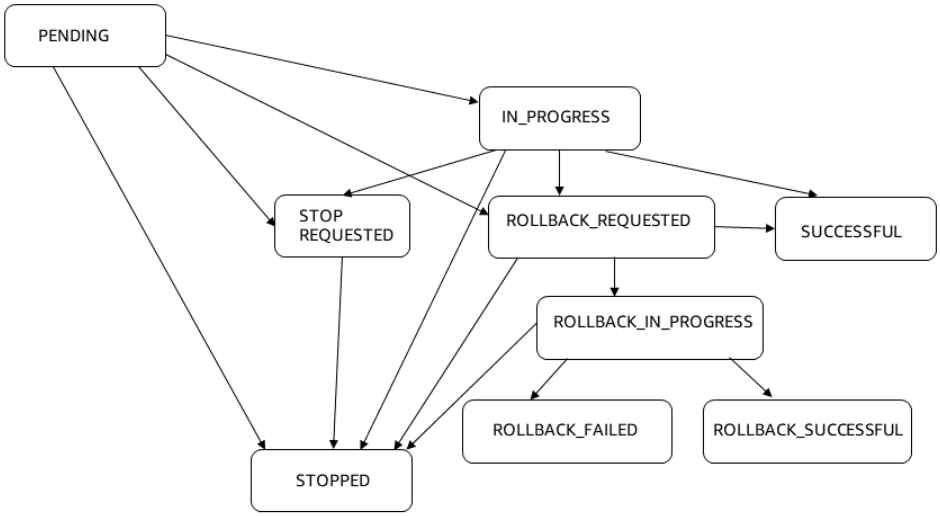
  Jumlah tugas yang berjalan yang ditunjukkan dalam revisi layanan mungkin tidak sama dengan jumlah sebenarnya dari tugas yang sedang berjalan. Nomor ini mewakili jumlah tugas yang berjalan saat penerapan selesai. Misalnya, jika Anda meluncurkan tugas independen dari penyebaran layanan, tugas tersebut tidak termasuk dalam jumlah tugas yang sedang berjalan untuk revisi layanan.
- Deteksi kegagalan pemutus sirkuit:
  - Jumlah tugas yang gagal untuk memulai
- CloudWatch deteksi kegagalan alarm
  - Alarm yang aktif
- Informasi rollback:
  - Waktu mulai
  - Alasan rollback
  - ARN dari revisi layanan yang digunakan untuk rollback
- Alasan status

Amazon ECS menghapus penyebaran layanan saat Anda menghapus layanan.

Status penyebaran layanan

Penyebaran layanan dimulai dalam `PENDING` status.

Ilustrasi berikut menunjukkan status penyebaran layanan yang dapat terjadi setelah `PENDING` status:`IN_PROGRESS`,,,`ROLLBACK_REQUESTED`,`SUCCESSFUL`,`STOP_REQUESTED`, `ROLLBACK_IN_PROGRESSS` `ROLLBACK_FAILEDROLLBACK_SUCCESSFUL`, dan. `STOPPED`



Informasi berikut memberikan rincian tentang status penyebaran layanan:

- `PENDING`- Penyebaran layanan telah dibuat, tetapi belum dimulai.

  Negara dapat pindah ke`IN_PROGRESS`,`ROLLBACK_REQUESTED`,`STOP_REQUESTED`, atau`STOPPED`.
- `IN_PROGRESS`- Penyebaran layanan sedang berlangsung.

  Negara dapat pindah ke`SUCCESSFUL`,`STOP_REQUESTED`,`ROLLBACK_REQUESTED`,`ROLLBACK_IN_PROGRESS`, dan`STOPPED`.
- `STOP_REQUESTED`- Status penyebaran layanan berpindah ke `STOP_REQUESTED` saat salah satu hal berikut terjadi:
  - Pengguna memulai penyebaran layanan baru.
  - Opsi rollback tidak digunakan untuk mekanisme deteksi kegagalan (pemutus sirkuit atau berbasis alarm) dan layanan tidak mencapai keadaan. `SUCCESSFUL`

  Negara bergerak ke`STOPPED`.
- `ROLLBACK_REQUESTED`- Status penyebaran layanan berpindah ke `ROLLBACK_REQUESTED` saat pengguna meminta rollback melalui konsol, API, atau CLI.

Negara dapat pindah keSUCCESSFUL,ROLLBACK_IN_PROGRESS, danSTOPPED.

- SUCCESSFUL- Status penyebaran layanan bergerak ke SUCCESSFUL saat penyebaran layanan berhasil diselesaikan.
- ROLLBACK_IN_PROGRESS- Status penyebaran layanan bergerak ke ROLLBACK_IN_PROGRESS saat opsi rollback digunakan untuk mekanisme deteksi kegagalan (pemutus sirkuit atau berbasis alarm) dan layanan gagal.

  Negara bergerak keROLLBACK_SUCCESSFUL, atauROLLBACK_FAILED.

Properti yang disertakan dalam penyebaran layanan Amazon ECS

Properti berikut disertakan dalam penyebaran layanan.

Properti	Deskripsi
Penyebaran layanan ARN	ARN dari penyebaran layanan.
Layanan ARN	ARN layanan untuk penyebaran layanan ini.
ARN klaster	ARN untuk cluster yang menjadi tuan rumah layanan.
Waktu pembuatan penyebaran layanan	Waktu penyebaran layanan dibuat.
Waktu mulai penyebaran layanan	Waktu penyebaran layanan dimulai.
Waktu penyelesaian penyebaran layanan	Waktu penyebaran layanan selesai.
Waktu penyebaran layanan dihentikan	Waktu penyebaran layanan berhenti.
Waktu pembaruan penyebaran layanan	Waktu penyebaran layanan terakhir diperbarui.

Properti	Deskripsi
Revisi layanan sumber	Revisi layanan yang sedang berjalan. Untuk informasi tentang properti yang disertakan, lihatProperti yang termasuk dalam revisi layanan Amazon ECS.
Konfigurasi deployment	Parameter penyebaran termasuk konfigurasi pemutus sirkuit, alarm yang menentuka n. • Batas bawah pada jumlah tugas yang harus dijalanka n untuk layanan selama penerapan atau ketika instance kontainer terkuras, sebagai persen dari jumlah tugas yang diinginkan untuk layanan. • Batas atas jumlah tugas yang harus dijalankan untuk layanan selama penerapan atau ketika instance kontainer terkuras, sebagai persen dari jumlah tugas yang diinginkan untuk layanan. • Konfigurasi pemutus sirkuit. • Alarm yang digunakan untuk deteksi kegagalan.

Properti	Deskripsi
Revisi layanan target	Revisi layanan untuk menyebarkan. Setelah penerapan berhasil diselesaikan, revisi layanan target adalah revisi layanan yang berjalan.
Status penyebaran layanan	Status penyebaran layanan. Nilai yang valid adalah PENDING, SUCCESS, STOP_REQUESTED, STOP_IN_PROGRESS, IN_PROGRESS, ROLLBACK_IN_PROGRESS, ROLLBACK_SUCCESSFUL, dan ROLLBACK_FAILED.
Informasi status penyebaran layanan	Informasi tentang mengapa penyebaran layanan dalam status saat ini. Misalnya, pemutus sirkuit mendeteksi kegagalan.
Informasi rollback	Opsi rollback yang digunakan penyebaran layanan saat penerapan gagal.
Opsi pemutus sirkuit penyebaran layanan	Pemutus sirkuit yang menentukan penyebaran layanan gagal.
CloudWatch alarm untuk penyebaran layanan	CloudWatch Alarm yang menentukan kapan penerapan layanan gagal.

Izin diperlukan untuk melihat penerapan layanan Amazon ECS

Saat Anda mengikuti praktik terbaik pemberian hak istimewa paling sedikit, Anda perlu menambahkan izin tambahan untuk melihat penerapan layanan di konsol.

Anda memerlukan akses ke tindakan berikut:

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

Anda memerlukan akses ke sumber daya berikut:

- Layanan
- Penyebaran layanan
- Revisi layanan

Kebijakan contoh berikut berisi izin yang diperlukan, dan membatasi tindakan ke layanan tertentu.

Ganti`account`,`cluster-name`, dan `service-name` dengan nilai-nilai Anda.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "ecs:ListServiceDeployments",
              "ecs:DescribeServiceDeployments",
              "ecs:DescribeServiceRevisions"
            ],
            "Resource": [
             "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
             "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/
service-name/*",
             "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-
name/*"
            ]
        }
    ]
}
```

Melihat penyebaran layanan Amazon ECS

Anda dapat melihat riwayat 90 hari terbaru untuk penerapan yang dibuat pada atau setelah 25 Oktober 2024. Penyebaran layanan dapat berada di salah satu status berikut:

- Sedang berlangsung
- Tertunda
- Selesai

Anda dapat menggunakan informasi ini untuk menentukan apakah Anda perlu memperbarui cara layanan digunakan, atau revisi layanan. Untuk informasi tentang properti yang disertakan, lihatProperti yang disertakan dalam penyebaran layanan Amazon ECS.

Sebelum Anda mulai, konfigurasikan izin yang diperlukan untuk melihat penerapan layanan. Untuk informasi selengkapnya, lihat Izin diperlukan untuk melihat penerapan layanan Amazon ECS.

Amazon ECS Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan ditampilkan.
4. Pada halaman detail layanan, pilih Deployment.
5. Pilih penyebaran layanan untuk dilihat.

Untuk melihat penerapan layanan untuk jenis penerapan ini	Lakukan hal berikut	
Penyebaran yang sedang berlangsung	Di bawah Deployment yang sedang berlangsung, pilih ID Deployment.	
Penerapan terakhir	Di bawah Deployment terakhir, pilih ID Deploymen t.	

Untuk melihat penerapan layanan untuk jenis penerapan ini	Lakukan hal berikut	
Penerapan selesai	Di bawah Penerapan layanan, pilih ID Deployment.	

Halaman detail penyebaran layanan muncul.

6. (Opsional) Bandingkan revisi layanan untuk melihat perbedaannya.

   Di bawah Revisi layanan, pilih Bandingkan revisi, lalu pilih 2 revisi untuk membandingkan.

   Revisi layanan ditampilkan side-by-side dengan perbedaan yang disorot.

AWS CLI

1. Jalankan `list-service-deployments` untuk mengambil ARN penyebaran layanan.

   Ganti variabel dengan nilai Anda.

   ```
   aws ecs list-service-deployments --cluster cluster-name --service service-name
   ```

   Perhatikan serviceDeploymentArn untuk penerapan yang ingin Anda lihat.

   ```
   {
       "serviceDeployments": [
           {
               "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
   deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
               "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/
   sd-example",
               "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
               "targetServiceRevisionArn": "arn:aws:ecs:us-
   west-2:123456789012:service-revision/example/sd-example/4980306466373577095",
               "status": "SUCCESSFUL"
           }
       ]
   ```

```
}
```

2. Jalankan `describe-service-deployments`. Gunakan `serviceDeploymentArn` yang dikembalikan dari`list-service-deployments`.

   Ganti variabel dengan nilai Anda.

```
aws ecs describe-service-deployments --service-deployment-arns
 arn:aws:ecs:region:123456789012:service-deployment/cluster-name/service-
name/NCWGC2ZR-taawPAYrIaU5
```

Langkah selanjutnya

Anda dapat melihat detail untuk revisi layanan dalam penerapan. Untuk informasi selengkapnya, lihat Melihat detail revisi layanan Amazon ECS

Revisi layanan Amazon ECS

Revisi layanan berisi catatan konfigurasi beban kerja yang coba diterapkan Amazon ECS. Setiap kali Anda membuat atau menerapkan layanan, Amazon ECS secara otomatis membuat dan menangkap konfigurasi yang Anda coba terapkan dalam revisi layanan.

Revisi layanan bersifat read-only dan memiliki pengidentifikasi unik. Untuk informasi tentang properti yang disertakan, lihatProperti yang termasuk dalam revisi layanan Amazon ECS.

Revisi layanan memberikan manfaat sebagai berikut:

- Selama penerapan layanan, Anda dapat membandingkan revisi layanan (sumber) yang saat ini diterapkan dengan yang sedang digunakan (target).
- Saat Anda menggunakan opsi rollback untuk penerapan layanan, Amazon ECS secara otomatis mengembalikan penerapan layanan ke revisi layanan terakhir yang berhasil diterapkan.
- Revisi layanan berisi catatan konfigurasi beban kerja dalam satu sumber daya.

Siklus hidup revisi layanan

Amazon ECS secara otomatis membuat revisi layanan baru saat Anda membuat layanan, atau memperbarui properti layanan yang memulai penerapan.

Amazon ECS tidak membuat revisi layanan baru untuk operasi rollback. Amazon ECS menggunakan revisi layanan terakhir yang berhasil untuk rollback.

Revisi layanan tidak dapat diubah.

Amazon ECS menghapus revisi layanan saat Anda menghapus layanan.

Anda dapat melihat revisi layanan yang dibuat pada atau setelah 25 Oktober 2024 dengan menggunakan konsol, API, dan CLI.

Properti yang termasuk dalam revisi layanan Amazon ECS

Properti berikut termasuk dalam revisi layanan.

Sumber Daya	Deskripsi
Layanan ARN	ARN yang mengidentifikasi layanan.
ARN klaster	ARN untuk cluster yang menjadi tuan rumah layanan.
Definisi tugas ARN	ARN dari definisi tugas yang digunakan untuk tugas layanan.
Registrasi layanan	Rincian untuk pendaftar layanan yang digunakan untuk penemuan layanan. • Nama kontainer Nilai nama kontainer yang akan digunakan untuk layanan penemuan layanan Anda. • Port kontainer Nilai port yang akan digunakan untuk layanan

Sumber Daya	Deskripsi	
	penemuan layanan Anda. Nilai ini juga ditentukan dalam definisi tugas. • Pelabuhan Nilai port yang digunakan jika layanan pencari layanan Anda menetapkan catatan SRV. • Registri ARN Amazon Resource Name (ARN) registri layanan.	
Penyedia kapasitas	Rincian strategi penyedia kapasitas. • Nama penyedia kapasitas Nama penyedia kapasitas yang digunakan untuk layanan tersebut. • Berat penyedia kapasitas Persentase relatif dari jumlah total tugas yang diluncurkan yang harus menggunakan penyedia kapasitas yang ditentukan. • Basis penyedia kapasitas Jumlah tugas, minimal, untuk dijalankan pada penyedia kapasitas yang ditentukan.	

Sumber Daya	Deskripsi
Image kontainer	Detail tentang gambar kontainer. • Nama kontainer • Intisari gambar • Gambar kontainer
Jaringan	Konfigurasi jaringan untuk layanan. • Apakah ada alamat IP publik • Subnet tempat tugas dijalankan • Kelompok keamanan yang mengontrol lalu lintas layanan
Jenis peluncuran	Jenis peluncuran yang digunakan untuk layanan ini.

Properti khusus fargate	Ketika jenis peluncurannya adalah Fargate, ini adalah informasi tentang versi Fargate.
	• Versi platform dan keluarga platform
	• Penyimpanan sementara
	  • Jumlah penyimpanan sementara untuk dialokasikan untuk penyebaran.
	  • Kunci KMS yang digunakan untuk enkripsi penyimpanan.

Sumber Daya	Deskripsi	
Volume Amazon EBS yang dikonfigurasi saat penerapan	Konfigurasi untuk volume yang ditentukan dalam definisi tugas sebagai volume yang dikonfigurasi pada waktu peluncuran. • Dienkripsi Menunjukkan apakah volume dienkripsi. • Jenis sistem file Jenis sistem file Linux untuk volume. • IOPS Jumlah I/O operasi per detik (IOPS). • Kunci KMS • Peran IAM ARN dari peran IAM untuk dikaitkan dengan volume ini. • Size Ukuran volume di GiB. • ID Snapshot ID snapshot volume. • Spesifikasi tag Konfigurasi tag untuk volume. • Throughput	

Sumber Daya	Deskripsi
	Throughput disediakan untuk volume. • Tipe volume Jenis volume.
Layanan Connect	Konfigurasi Service Connect. • Indikasi apakah Service Connect sedang digunakan • Namespace • Daftar layanan yang saling berhubungan • Konfigurasi logging Service Connect
Penyeimbang beban layanan	Penyeimbang beban yang merutekan lalu lintas layanan. • Nama • Nama kontainer • Port kontainer • Kelompok sasaran ARN
Pemantauan Runtime	Menunjukkan jika Runtime Monitoring aktif.
Tanggal pembuatan	Tanggal revisi layanan dibuat.
Kisi VPC	Konfigurasi VPC Lattice untuk revisi layanan.

Melihat detail revisi layanan Amazon ECS

Anda dapat melihat informasi tentang jenis revisi layanan berikut yang dibuat pada atau setelah 25 Oktober 2024:

- Sumber -Konfigurasi beban kerja yang saat ini digunakan
- Target - Konfigurasi beban kerja sedang digunakan

Amazon ECS Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan ditampilkan.
4. Pada halaman detail layanan, pilih Deployment.
5. Pilih revisi layanan untuk dilihat.

Untuk melihat revisi layanan untuk jenis penerapan ini	Lakukan hal berikut	
Penerapan yang sedang berlangsung	Di bawah Penerapan yang sedang berlangsung, lakukan hal berikut: • Untuk melihat revisi layanan sumber, di bawah Revisi layanan, pilih ID revisi layanan untuk jenis Revisi Sumber. • Untuk melihat revisi layanan target, di bawah Revisi layanan, pilih ID revisi layanan untuk jenis Revisi Target.	

Untuk melihat revisi layanan untuk jenis penerapan ini	Lakukan hal berikut
Penerapan terakhir	Di bawah Penerapan terakhir, pilih revisi layanan target.
Penerapan selesai	Di bawah Penerapan Layanan, lakukan hal berikut: • Di bawah Revisi layanan target, pilih ID.

AWS CLI

1. Jalankan `describe-service-deployments` untuk mengambil ARN revisi layanan.

   Ganti variabel dengan nilai Anda.

   ```
   aws ecs describe-service-deployments --service-deployment-arns
    arn:aws:ecs:region:account-id:service/cluster-name/service-name/NCWGC2ZR-
   taawPAYrIaU5
   ```

   Perhatikan `arn` untuk `sourceServiceRevisions` atau`targetServiceRevisions`.

   ```
   {
       "serviceDeployments": [
           {
               "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
   deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
               "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/
   sd-example",
               "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
               "updatedAt": "2024-09-10T16:49:35.572000+00:00",
               "sourceServiceRevision": {
                   "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
   example/sd-example/4980306466373578954",
                   "requestedTaskCount": 0,
                   "runningTaskCount": 0,
   ```

```
                "pendingTaskCount": 0
            },
            "targetServiceRevision": {
                "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
example/sd-example/4980306466373577095",
                "requestedTaskCount": 0,
                "runningTaskCount": 0,
                "pendingTaskCount": 0
            },
            "status": "IN_PROGRESS",
            "deploymentConfiguration": {
                "deploymentCircuitBreaker": {
                    "enable": false,
                    "rollback": false
                },
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            }
        }
    ],
    "failures": []
}
```

2. Jalankan `describe-service-revisions`. Gunakan `arn` yang dikembalikan dari`describe-service-deployments`.

   Ganti variabel dengan nilai Anda.

   ```
   aws ecs describe-service-revisions --service-revision-arns
    arn:aws:ecs:region:123456789012:service-revision/cluster-name/service-
   name/4980306466373577095
   ```

Menyeimbangkan layanan Amazon ECS di seluruh Availability Zone

Untuk membantu aplikasi Anda mencapai ketersediaan tinggi, kami sarankan untuk mengonfigurasi layanan multi-tugas Anda agar berjalan di beberapa Availability Zone. Untuk layanan yang menentukan strategi penempatan pertama mereka sebagai penyebaran Availability Zone, AWS lakukan upaya terbaik untuk mendistribusikan tugas layanan secara merata di seluruh Availability Zone yang tersedia.

Namun, mungkin ada kalanya jumlah tugas yang berjalan di satu Availability Zone tidak sama dengan Availability Zone lainnya, seperti setelah gangguan Availability Zone. Untuk mengatasi ketidakseimbangan tugas ini, Anda dapat mengaktifkan fitur penyeimbangan kembali Availability Zone.

Dengan penyeimbangan kembali Availability Zone, Amazon ECS terus memantau distribusi tugas di seluruh Availability Zone untuk setiap layanan Anda. Saat Amazon ECS mendeteksi distribusi tugas yang tidak merata, Amazon secara otomatis mengambil tindakan untuk menyeimbangkan kembali beban kerja di seluruh Availability Zone. Ini melibatkan peluncuran tugas baru di Availability Zones dengan tugas paling sedikit dan mengakhiri tugas di Availability Zone yang kelebihan beban.

Redistribusi ini memastikan tidak ada Availability Zone tunggal yang menjadi titik kegagalan, membantu menjaga ketersediaan keseluruhan aplikasi kontainer Anda. Proses penyeimbangan ulang otomatis menghilangkan kebutuhan akan intervensi manual, mempercepat waktu untuk pemulihan setelah suatu peristiwa.

Berikut ini adalah ikhtisar proses penyeimbangan kembali Availability Zone:

1. Amazon ECS mulai memantau layanan setelah mencapai kondisi tunak, dan melihat jumlah tugas yang berjalan di setiap Availability Zone.
2. Amazon ECS melakukan operasi berikut ketika mendeteksi ketidakseimbangan dalam jumlah tugas yang berjalan di setiap Availability Zone:
   - Mengirim acara layanan yang menunjukkan bahwa penyeimbangan kembali Availability Zone dimulai.
   - Memulai tugas di Availability Zones dengan jumlah tugas yang berjalan paling sedikit
   - Menghentikan tugas di Availability Zones dengan jumlah tugas yang berjalan terbesar.
   - Penjadwal menunggu tugas yang baru dimulai `HEALTHY` dan `RUNNING` sebelum menghentikan tugas di Availability Zone yang terlalu berskala.
   - Mengirim acara layanan dengan hasil penyeimbangan kembali Availability Zone.

Bagaimana Amazon ECS mendeteksi distribusi tugas yang tidak merata

Amazon ECS menentukan ketidakseimbangan dalam jumlah tugas yang berjalan di setiap Availability Zone dengan membagi jumlah tugas yang diinginkan layanan dengan jumlah Availability Zone yang dikonfigurasi. Jika jumlah tugas yang diinginkan tidak dibagi secara merata, Amazon ECS mendistribusikan sisa tugas secara merata di seluruh Availability Zone yang dikonfigurasi. Setiap Availability Zone harus memiliki setidaknya satu tugas.

Misalnya, pertimbangkan layanan Amazon ECS dengan hitungan dua tugas yang diinginkan yang dikonfigurasi untuk dua Availability Zone. Dalam skenario ini, jumlah tugas yang diinginkan terbagi secara merata. Distribusi yang seimbang akan menjadi satu tugas per Availability Zone. Jika ada dua tugas di Availability Zone 1 dan zero task di Availability Zone 2, Amazon ECS akan memulai rebalancing dengan memulai tugas di Availability Zone 2 sebelum menghentikan tugas di Availability Zone 1.

Sekarang, pertimbangkan layanan Amazon ECS dengan hitungan tiga tugas yang diinginkan yang dikonfigurasi untuk dua Availability Zone. Dalam skenario ini, jumlah tugas yang diinginkan tidak membagi secara merata. Distribusi yang seimbang akan menjadi satu tugas di Availability Zone 1 dan dua tugas di Availability Zone 2 karena setiap Availability Zone memiliki setidaknya satu tugas dan tugas sisanya ditempatkan di Availability Zone 2.

Pertimbangkan layanan Amazon ECS yang memiliki jumlah lima tugas yang diinginkan yang dikonfigurasi untuk tiga Availability Zone. Dalam skenario ini, jumlah tugas yang diinginkan tidak membagi secara merata. Distribusi yang seimbang akan menjadi satu tugas di Availability Zone 1 dan dua tugas masing-masing di Availability Zones 2 dan 3. Setelah memperhitungkan setiap Availability Zone yang masing-masing memiliki satu tugas, dua tugas yang tersisa didistribusikan secara merata di seluruh Availability Zone.

Pertimbangan untuk mengonfigurasi penyeimbangan kembali Availability Zone

Pertimbangkan hal berikut ketika Anda ingin mengonfigurasi penyeimbangan kembali Availability Zone:

- Penyeimbangan kembali Availability Zone mendukung Fargate dan tipe peluncuran. EC2 Untuk Fargate, Amazon ECS akan secara otomatis mendistribusikan kembali tugas di seluruh Availability Zone yang tersedia untuk menjaga keseimbangan. Untuk jenis EC2 peluncuran, Amazon ECS menyeimbangkan kembali tugas di seluruh instans kontainer yang ada dengan upaya terbaik, dengan menghormati strategi dan batasan penempatan yang Anda tentukan. Namun, Amazon ECS tidak dapat meluncurkan instans baru di Availability Zone yang kurang dimanfaatkan sebagai bagian dari proses penyeimbangan kembali, sehingga membatasi penyeimbangan kembali ke instance container yang ada.
- Penyeimbangan ulang Availability Zone berfungsi dalam konfigurasi berikut:
  - Layanan yang menggunakan `Replica` strategi
  - Layanan yang menentukan penyebaran Availability Zone sebagai strategi penempatan tugas pertama, atau tidak menentukan strategi penempatan.

- Anda tidak dapat menggunakan penyeimbangan kembali Availability Zone dengan layanan yang memenuhi salah satu kriteria berikut:
  - Menggunakan `Daemon` strategi
  - Menggunakan tipe `EXTERNAL` peluncuran (ECS Anywhere)
  - Menggunakan 100% untuk `maximumPercent` nilainya
  - Menggunakan Classic Load Balancer
  - Menggunakan `attribute:ecs.availability-zone` sebagai kendala penempatan tugas

Strategi penempatan dan kendala penempatan dengan penyeimbangan kembali Availability Zone

Strategi penempatan menentukan cara Amazon ECS memilih instans kontainer dan Availability Zone untuk penghentian penempatan tugas. Kendala penempatan tugas adalah aturan yang menentukan apakah tugas diizinkan untuk dijalankan pada instance kontainer tertentu.

Untuk jenis EC2 peluncuran, Anda dapat menggunakan strategi penempatan dan batasan penempatan bersama dengan penyeimbangan kembali Availability Zone. Namun, agar penyeimbangan kembali Availability Zone berfungsi, strategi penempatan spread Availability Zone harus menjadi strategi pertama yang ditentukan.

Penyeimbangan kembali Availability Zone kompatibel dengan berbagai kombinasi strategi penempatan. Misalnya, Anda dapat membuat strategi yang pertama-tama mendistribusikan tugas secara merata di seluruh Availability Zone, lalu bin pack tugas berdasarkan memori dalam setiap Availability Zone. Dalam hal ini, penyeimbangan kembali Availability Zone berfungsi karena strategi penyebaran Availability Zone ditentukan terlebih dahulu.

Penting untuk dicatat bahwa penyeimbangan kembali Availability Zone tidak akan berfungsi jika strategi pertama dalam array strategi penempatan bukan komponen penyebaran Availability Zone. Persyaratan ini memastikan bahwa fokus utama distribusi tugas adalah menjaga keseimbangan di seluruh Availability Zone, yang sangat penting untuk ketersediaan tinggi.

Untuk informasi selengkapnya tentang strategi dan kendala penempatan tugas, lihat. Bagaimana Amazon ECS menempatkan tugas pada instans kontainer

Strategi penempatan tugas dan kendala tidak didukung untuk tugas yang menggunakan tipe peluncuran Fargate. Fargate akan mencoba yang terbaik untuk menyebarkan tugas di seluruh Availability Zone yang dapat diakses. Jika penyedia kapasitas mencakup Fargate dan Fargate Spot, perilaku spread bersifat independen untuk setiap penyedia kapasitas.

Contoh strategi berikut mendistribusikan tugas secara merata di seluruh Availability Zones, dan kemudian bin pack tugas berdasarkan memori dalam setiap Availability Zone. Penyeimbangan ulang Availability Zone kompatibel dengan layanan karena `spread` strateginya adalah yang pertama.

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Aktifkan penyeimbangan kembali Availability Zone

Anda perlu mengaktifkan penyeimbangan kembali Availability Zone untuk layanan baru dan yang sudah ada.

Anda dapat mengaktifkan dan menonaktifkan penyeimbangan kembali Availability Zone menggunakan konsol, APIs, atau. AWS CLI

Jenis layanan	API	Konsol	CLI
Ada	UpdateService	Memperbarui layanan Amazon ECS	layanan pembaruan-
Baru	CreateService	Membuat penyebaran pembaruan bergulir Amazon ECS	membuat-layanan

Contoh berikut menunjukkan cara mengaktifkan penyeimbangan kembali layanan saat membuat layanan baru:

```
aws ecs create-service \
    --cluster my-cluster \
    --service-name my-service \
    --task-definition my-task-definition:1 \
```

```
    --desired-count 6 \
    --availability-zone-rebalancing ENABLED
```

Lacak penyeimbangan kembali Zona Ketersediaan Amazon ECS

Anda dapat memverifikasi apakah penyeimbangan ulang Availability Zone diaktifkan untuk layanan di konsol atau dengan menelepon. `describe-services` Contoh berikut dapat digunakan untuk melihat status dengan CLI.

Responsnya akan berupa `ENABLED` atau`DISABLED`.

```
aws ecs describe-services \
    --services service-name \
    --cluster cluster-name \
    --query services[0].availabilityZoneRebalancing
```

Acara layanan

Amazon ECS mengirimkan peristiwa tindakan layanan untuk membantu Anda memahami siklus hidup penyeimbangan ulang Availability Zone.

Peristiwa	Skenario	Tipe	Pelajari selengkapnya
`SERVICE_REBALANCING_STARTED`	Amazon ECS memulai operasi penyeimbangan kembali Availability Zone	INFO	service (*service-name*) tidak AZ seimbang dengan *number-tasks* tugas di dalam*Availability Zone 1* , *number-tasks* di*Availability Zone 2* , dan *number-tasks* di*Availability Zone 3* . AZ Rebalancing sedang berlangsung.

Peristiwa	Skenario	Tipe	Pelajari selengkapnya
SERVICE_REBALANCING_COMPLETED	Operasi penyeimbangan kembali Availability Zone selesai	INFO	service (*service-name*) adalah AZ seimbang dengan *number-tasks* tugas di*Availability Zone 1* , *number-tasks* tugas di*Availability Zone 2* , dan *number-tasks* tugas di*Availability Zone 3* .
TASKS_STARTED	Amazon ECS berhasil memulai tugas sebagai bagian dari operasi penyeimbangan kembali Availability Zone	INFO	*service-name* telah memulai *number-tasks* tugas *Availability Zone* ke AZ Rebalance:*task-ids*.
TASKS_STOPPED	Amazon ECS berhasil menghentikan tugas sebagai bagian dari operasi penyeimbangan kembali Availability Zone	INFO	*service-name* telah berhenti *number-tasks* menjalankan tugas *Availability Zone* karena penyeimbangan kembali AZ:. *task-id*

Peristiwa	Skenario	Tipe	Pelajari selengkapnya
SERVICE_TASK_PLACEMENT_FAILURE	Amazon ECS gagal memulai tugas sebagai bagian dari operasi penyeimbangan kembali Availability Zone	ERROR	Untuk jenis EC2 peluncuran, lihat service (*service-name*) tidak dapat menempatkan tugas *Availability Zone* karena tidak ada instance kontainer yang memenuhi semua persyaratannya. Untuk jenis peluncuran Fargate, lihat service (*service-name*) tidak dapat menempatkan tugas di*Availability Zone*.
TASKSET_SCALE_IN_FAILURE_BY_TASK_PROTECTION	Operasi penyeimbangan kembali Availability Zone diblokir karena perlindungan tugas sedang digunakan.	INFO	service (*service-name*) tidak dapat melakukan AZ Rebalance karena *task-set-name* tidak dapat menskalakan karena*reason*.

Peristiwa	Skenario	Tipe	Pelajari selengkapnya
SERVICE_REBALANCING_STOPPED	Operasi penyeimbangan kembali Availability Zone berhenti. Amazon ECS mengirimkan acara tambahan yang memberikan informasi lebih lanjut.	INFO	service (*service-name*) menghentikan AZ Rebalancing.

Peristiwa perubahan status tugas

Amazon ECS mengirimkan peristiwa perubahan status tugas (START) untuk setiap tugas yang dimulai sebagai bagian dari proses penyeimbangan kembali.

Amazon ECS mengirimkan peristiwa perubahan status tugas (STOPPED) untuk setiap tugas yang dihentikan sebagai bagian dari proses penyeimbangan kembali. Alasannya diatur ke`Availability-zone rebalancing initiated by (deployment ecs-svc/`*deployment-id*`)`.

Untuk informasi lebih lanjut tentang acara, lihatAcara perubahan status tugas Amazon ECS.

Pemecahan masalah penyeimbangan kembali layanan

Jika Anda mengalami masalah dengan penyeimbangan kembali layanan, pertimbangkan langkah-langkah pemecahan masalah berikut:

Penyeimbangan kembali tidak dimulai

Verifikasi bahwa:

- Penyeimbangan kembali layanan diaktifkan untuk layanan Anda
- Layanan Anda menggunakan konfigurasi yang didukung (lihatthe section called “Pertimbangan untuk mengonfigurasi penyeimbangan kembali Availability Zone”)
- Layanan Anda telah mencapai kondisi mapan

Kegagalan penempatan tugas selama penyeimbangan kembali

Jika Anda melihat SERVICE_TASK_PLACEMENT_FAILURE acara:

- Untuk jenis EC2 peluncuran: Periksa apakah Anda memiliki instance kontainer yang tersedia di Zona Ketersediaan target
- Untuk jenis peluncuran Fargate: Periksa apakah ada kendala sumber daya atau kuota layanan yang membatasi penempatan tugas
- Tinjau batasan penempatan tugas Anda untuk memastikan mereka tidak mencegah distribusi tugas yang tepat

Penyeimbangan kembali berhenti secara tak terduga

Jika Anda melihat `SERVICE_REBALANCING_STOPPED` acara:

- Periksa perlindungan tugas yang mungkin menghalangi operasi
- Cari penerapan layanan bersamaan yang dapat mengganggu penyeimbangan kembali
- Tinjau peristiwa layanan untuk informasi tambahan tentang mengapa penyeimbangan kembali dihentikan

Praktik terbaik untuk penyeimbangan kembali layanan

Ikuti praktik terbaik ini untuk mendapatkan hasil maksimal dari penyeimbangan kembali layanan:

- Memantau operasi penyeimbangan kembali - Siapkan CloudWatch alarm untuk memantau peristiwa layanan yang terkait dengan penyeimbangan kembali untuk mengidentifikasi masalah apa pun dengan cepat.
- Pertimbangkan dampak kinerja - Sadarilah bahwa operasi penyeimbangan kembali dapat meningkatkan penggunaan sumber daya sementara saat tugas baru dimulai sebelum tugas lama dihentikan.
- Gunakan perlindungan tugas secara strategis - Jika Anda memiliki tugas penting yang tidak boleh dihentikan selama penyeimbangan kembali, pertimbangkan untuk menggunakan perlindungan tugas.
- Rencanakan EC2 kapasitas - Untuk jenis EC2 peluncuran, pastikan Anda memiliki instans kontainer yang cukup di semua Availability Zone untuk mendukung penyeimbangan kembali yang efektif.
- Uji perilaku penyeimbangan kembali - Sebelum mengandalkan penyeimbangan kembali dalam produksi, uji bagaimana layanan Anda berperilaku selama operasi penyeimbangan kembali di lingkungan non-produksi.

Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS

Layanan Anda secara opsional dapat dikonfigurasi untuk menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas dalam layanan Anda.

> **Note**
>
> Bila Anda menggunakan set tugas, semua tugas dalam set harus dikonfigurasi untuk menggunakan Elastic Load Balancing atau untuk tidak menggunakan Elastic Load Balancing.

Layanan Amazon ECS yang dihosting AWS Fargate mendukung Application Load Balancers, Network Load Balancers, dan Gateway Load Balancers. Gunakan tabel berikut untuk mempelajari jenis penyeimbang beban apa yang akan digunakan.

Jenis Load Balancer	Gunakan dalam kasus ini	
Penyeimbang Beban Aplikasi	Lalu lintas rute HTTP/HTTPS (atau lapisan 7). Application Load Balancers menawarkan beberapa fitur yang membuatnya menarik untuk digunakan dengan layanan Amazon ECS: • Setiap layanan dapat melayani lalu lintas dari beberapa penyeimbang beban dan mengekspo s beberapa port yang diseimbangkan beban dengan menentukan beberapa grup target.	

Jenis Load Balancer	Gunakan dalam kasus ini	
	• Mereka didukung oleh tugas yang dihosting di Fargate dan EC2 instance. • Application Load Balancers memungkinkan kontainer untuk menggunakan pemetaan port host dinamis (sehingga beberapa tugas dari layanan yang sama diperbolehkan per instance kontainer). • Application Load Balancers mendukung routing berbasis jalur dan aturan prioritas (sehingga beberapa layanan dapat menggunakan port listener yang sama pada satu Application Load Balancer).	
Penyeimbang Beban Jaringan	Rute lalu lintas TCP atau UDP (atau lapisan 4).	
Penyeimbang Beban Gateway	Rute lalu lintas TCP atau UDP (atau lapisan 4). Gunakan peralatan virtual, seperti firewall, sistem deteksi dan pencegahan intrusi, dan sistem inspeksi paket dalam.	

Kami menyarankan Anda menggunakan Application Load Balancer untuk layanan Amazon ECS Anda sehingga Anda dapat memanfaatkan fitur-fitur terbaru ini, kecuali layanan Anda memerlukan fitur yang hanya tersedia dengan Network Load Balancers atau Gateway Load Balancer. Untuk

informasi selengkapnya tentang Elastic Load Balancing dan perbedaan antara jenis load balancer, lihat Panduan Pengguna Elastic Load Balancing.

Dengan penyeimbang beban, Anda hanya membayar apa yang Anda gunakan. Untuk informasi selengkapnya, lihat Harga Elastic Load Balancing.

Optimalkan parameter pemeriksaan kesehatan penyeimbang beban untuk Amazon ECS

Load balancer merutekan permintaan hanya ke target sehat di Availability Zones untuk penyeimbang beban. Setiap target didaftarkan ke kelompok sasaran. Penyeimbang beban memeriksa kesehatan setiap target, menggunakan pengaturan pemeriksaan kesehatan kelompok sasaran. Setelah Anda mendaftarkan target, itu harus melewati satu pemeriksaan kesehatan agar dianggap sehat. Amazon ECS memonitor penyeimbang beban. Penyeimbang beban secara berkala mengirimkan pemeriksaan kesehatan ke wadah Amazon ECS. Agen Amazon ECS memantau, dan menunggu penyeimbang beban melaporkan kesehatan kontainer. Ia melakukan ini sebelum menganggap wadah berada dalam status sehat.

Dua parameter pemeriksaan kesehatan Elastic Load Balancing memengaruhi kecepatan penerapan:

- Interval pemeriksaan kesehatan: Menentukan perkiraan jumlah waktu, dalam hitungan detik, antara pemeriksaan kesehatan wadah individu. Secara default, penyeimbang beban memeriksa setiap 30 detik.

  Parameter ini dinamai:

  - `HealthCheckIntervalSeconds`di Elastic Load Balancing API
  - Interval di EC2 konsol Amazon
- Jumlah ambang batas yang sehat: Menentukan jumlah keberhasilan pemeriksaan kesehatan berturut-turut yang diperlukan sebelum mempertimbangkan wadah yang tidak sehat sehat. Secara default, penyeimbang beban memerlukan lima pemeriksaan kesehatan yang lewat sebelum melaporkan bahwa wadah target sehat.

  Parameter ini dinamai:

  - `HealthyThresholdCount`di Elastic Load Balancing API
  - Ambang batas yang sehat di EC2 konsol Amazon

Penting: Untuk target yang baru terdaftar, hanya satu pemeriksaan kesehatan yang berhasil diperlukan untuk mempertimbangkan target yang sehat, terlepas dari pengaturan jumlah ambang

batas yang sehat. Jumlah ambang batas yang sehat hanya berlaku ketika target beralih dari keadaan tidak sehat kembali ke keadaan sehat.

Dengan pengaturan default, jika target menjadi tidak sehat dan kemudian pulih, total waktu untuk menentukan kesehatan wadah adalah dua menit dan 30 detik (`30 seconds * 5 = 150 seconds`).

Anda dapat mempercepat proses pemeriksaan kesehatan jika layanan Anda dimulai dan stabil dalam waktu kurang dari 10 detik. Untuk mempercepat proses, kurangi interval pemeriksaan kesehatan dan jumlah ambang batas yang sehat.

- `HealthCheckIntervalSeconds`(Nama Elastic Load Balancing API) atau Interval (nama EC2 konsol Amazon): 5
- `HealthyThresholdCount`(Nama Elastic Load Balancing API) atau ambang Healthy (nama EC2 konsol Amazon): 2

Dengan pengaturan ini, proses pemeriksaan kesehatan membutuhkan waktu 10 detik dibandingkan dengan default dua menit dan 30 detik.

Untuk informasi selengkapnya tentang parameter pemeriksaan kesehatan Elastic Load Balancing, lihat Pemeriksaan Kesehatan untuk grup target Anda di Panduan Pengguna Elastic Load Balancing.

Optimalkan parameter pengurasan koneksi penyeimbang beban untuk Amazon ECS

Untuk memungkinkan pengoptimalan, klien mempertahankan koneksi tetap hidup ke layanan kontainer. Ini memungkinkan permintaan berikutnya dari klien tersebut untuk menggunakan kembali koneksi yang ada. Saat Anda ingin menghentikan lalu lintas ke kontainer, Anda memberi tahu penyeimbang beban. Penyeimbang beban secara berkala memeriksa untuk melihat apakah klien menutup koneksi tetap hidup. Agen Amazon ECS memantau penyeimbang beban, dan menunggu penyeimbang beban melaporkan bahwa koneksi tetap hidup ditutup (target dalam keadaan). `UNUSED`

Jumlah waktu yang menunggu penyeimbang beban untuk memindahkan target ke `UNUSED` status adalah penundaan deregistrasi. Anda dapat mengonfigurasi parameter penyeimbang beban berikut untuk mempercepat penerapan Anda.

- `deregistration_delay.timeout_seconds`: 300 (default)

Ketika Anda memiliki layanan dengan waktu respons di bawah 1 detik, atur parameter ke nilai berikut agar penyeimbang beban hanya menunggu 5 detik sebelum memutuskan koneksi antara klien dan layanan back-end:

- `deregistration_delay.timeout_seconds`: 5

> **Note**
>
> Jangan atur nilainya menjadi 5 detik saat Anda memiliki layanan dengan permintaan yang berumur panjang, seperti unggahan file yang lambat atau koneksi streaming.

Responsif SIGTERM

Amazon ECS pertama kali mengirimkan sinyal SIGTERM ke tugas untuk memberi tahu aplikasi harus diselesaikan dan dimatikan. Kemudian, Amazon ECS mengirimkan pesan SIGKILL. Ketika aplikasi mengabaikan SIGTERM, layanan Amazon ECS harus menunggu untuk mengirim sinyal SIGKILL untuk menghentikan proses.

Jumlah waktu yang Amazon ECS menunggu untuk mengirim pesan SIGKILL ditentukan oleh opsi agen Amazon ECS berikut:

- `ECS_CONTAINER_STOP_TIMEOUT`: 30 (default)

  Untuk informasi selengkapnya tentang parameter agen kontainer, lihat Agen Kontainer Amazon ECS di GitHub.

Untuk mempercepat masa tunggu, atur parameter agen Amazon ECS ke nilai berikut:

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

  Jika aplikasi Anda membutuhkan waktu lebih dari 1 detik, kalikan nilainya dengan 2 dan gunakan angka itu sebagai nilainya.

Dalam hal ini, Amazon ECS menunggu 2 detik hingga wadah dimatikan, dan kemudian Amazon ECS mengirimkan pesan SIGKILL ketika aplikasi tidak berhenti.

Anda juga dapat memodifikasi kode aplikasi untuk menjebak sinyal SIGTERM dan bereaksi terhadapnya. Berikut ini adalah contoh di JavaScript:

```
process.on('SIGTERM', function() {
  server.close();
})
```

Kode ini menyebabkan server HTTP berhenti mendengarkan permintaan baru, menyelesaikan menjawab permintaan dalam penerbangan, dan kemudian proses Node.js berakhir karena loop peristiwa tidak ada hubungannya. Mengingat hal ini, jika dibutuhkan proses hanya 500 ms untuk menyelesaikan permintaan dalam penerbangannya, itu berakhir lebih awal tanpa harus menunggu batas waktu berhenti dan dikirimi SIGKILL.

Menggunakan Application Load Balancer untuk Amazon ECS

Application Load Balancer membuat keputusan routing di layer aplikasi (HTTP/HTTPS), mendukung routing berbasis jalur, dan dapat merutekan permintaan ke satu atau beberapa port pada setiap instance container di cluster Anda. Aplikasi Load Balancer mendukung pemetaan port host dinamis. Misalnya, jika definisi kontainer tugas Anda menentukan port 80 untuk port kontainer NGINX, dan port 0 untuk port host, maka port host dipilih secara dinamis dari rentang port sementara instance kontainer (seperti 32768 hingga 61000 pada AMI Amazon ECS terbaru yang dioptimalkan). Ketika tugas diluncurkan, kontainer NGINX terdaftar dengan Application Load Balancer sebagai ID instance dan kombinasi port, dan lalu lintas didistribusikan ke ID instance dan port yang sesuai dengan container tersebut. Pemetaan dinamis ini mengizinkan Anda memiliki banyak tugas dari satu layanan pada instans kontainer yang sama. Untuk informasi selengkapnya, lihat Panduan Pengguna untuk Penyeimbang Beban Aplikasi.

Untuk informasi tentang praktik terbaik untuk menyetel parameter guna mempercepat penerapan Anda, lihat:

- Optimalkan parameter pemeriksaan kesehatan penyeimbang beban untuk Amazon ECS
- Optimalkan parameter pengurasan koneksi penyeimbang beban untuk Amazon ECS

Pertimbangkan hal berikut saat menggunakan Application Load Balancers dengan Amazon ECS:

- Amazon ECS memerlukan peran IAM terkait layanan yang menyediakan izin yang diperlukan untuk mendaftarkan dan membatalkan pendaftaran target dengan penyeimbang beban Anda saat tugas dibuat dan dihentikan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Grup target harus memiliki jenis alamat IP yang disetel ke IPv4.

Meskipun tugas mendukung IPv6, integrasi layanan dengan IPv6 Elastic Load Balancing tidak didukung.

- Untuk layanan dengan tugas menggunakan mode `awsvpc` jaringan, saat Anda membuat grup target untuk layanan Anda, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon.
- Jika layanan Anda memerlukan akses ke beberapa port yang seimbang beban, seperti port 80 dan port 443 untuk suatu HTTP/HTTPS layanan, Anda dapat mengonfigurasi dua pendengar. Satu listener bertanggung jawab untuk HTTPS yang meneruskan permintaan ke layanan, dan listener lainnya bertanggung jawab untuk mengarahkan permintaan HTTP ke port HTTPS yang sesuai. Untuk informasi selengkapnya, lihat Membuat pendengar ke Application Load Balancer di Panduan Pengguna untuk Application Load Balancers.
- Konfigurasi subnet penyeimbang beban Anda harus menyertakan semua Availability Zone tempat instans kontainer Anda berada.
- Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika tugas layanan gagal dalam kriteria pemeriksaan kesehatan penyeimbang beban, tugas dihentikan dan dimulai ulang. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas berjalan yang diinginkan.
- Jika Anda mengalami masalah dengan layanan yang diaktifkan penyeimbang beban, lihat Memecahkan masalah penyeimbang beban layanan di Amazon ECS.
- Saat menggunakan tipe `instance` target, tugas dan penyeimbang beban Anda harus dalam VPC yang sama. Saat menggunakan tipe `ip` target, konektivitas lintas-VPC didukung.
- Gunakan grup target unik untuk setiap layanan.

Menggunakan grup target yang sama untuk beberapa layanan dapat menyebabkan masalah selama penerapan layanan.

Untuk selengkapnya tentang cara membuat Application Load Balancer, lihat Membuat Application Load Balancer di Application Load Balancers

Menggunakan Network Load Balancer untuk Amazon ECS

Network Load Balancer membuat keputusan routing pada layer transport (TCP/SSL). Hal itu dapat menangani jutaan permintaan per detik. Setelah penyeimbang beban menerima koneksi, penyeimbang akan memilih target dari grup target untuk aturan default menggunakan algoritme perutean hash alur. Penyeimbang mencoba untuk membuka koneksi TCP ke target yang dipilih pada port yang ditentukan dalam konfigurasi listener. Ini meneruskan permintaan tanpa memodifikasi header. Network Load Balancers mendukung pemetaan port host dinamis. Misalnya, jika definisi kontainer tugas Anda menentukan port 80 untuk port kontainer NGINX, dan port 0 untuk port host, maka port host dipilih secara dinamis dari rentang port sementara instance kontainer (seperti 32768 hingga 61000 pada AMI Amazon ECS terbaru yang dioptimalkan). Ketika tugas diluncurkan, kontainer NGINX terdaftar dengan Network Load Balancer sebagai ID instance dan kombinasi port, dan lalu lintas didistribusikan ke ID instance dan port yang sesuai dengan container tersebut. Pemetaan dinamis ini mengizinkan Anda memiliki banyak tugas dari satu layanan pada instans kontainer yang sama. Untuk informasi selengkapnya, lihat Panduan Pengguna untuk Network Load Balancer.

Untuk informasi tentang praktik terbaik untuk menyetel parameter guna mempercepat penerapan Anda, lihat:

- Optimalkan parameter pemeriksaan kesehatan penyeimbang beban untuk Amazon ECS
- Optimalkan parameter pengurasan koneksi penyeimbang beban untuk Amazon ECS

Pertimbangkan hal berikut saat menggunakan Network Load Balancer dengan Amazon ECS:

- Amazon ECS memerlukan peran IAM terkait layanan yang menyediakan izin yang diperlukan untuk mendaftarkan dan membatalkan pendaftaran target dengan penyeimbang beban Anda saat tugas dibuat dan dihentikan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Anda tidak dapat melampirkan lebih dari lima grup target ke layanan.

- Untuk layanan dengan tugas menggunakan mode `awsvpc` jaringan, saat Anda membuat grup target untuk layanan Anda, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon.
- Konfigurasi subnet penyeimbang beban Anda harus menyertakan semua Availability Zone tempat instans kontainer Anda berada.
- Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika tugas layanan gagal dalam kriteria pemeriksaan kesehatan penyeimbang beban, tugas dihentikan dan dimulai ulang. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas berjalan yang diinginkan.
- Saat Anda menggunakan Load Balancer Gateway yang dikonfigurasi dengan alamat IP sebagai target dan Pelestarian IP Klien tidak aktif, permintaan akan terlihat berasal dari alamat IP pribadi Gateway Load Balancers. Ini berarti bahwa layanan di balik Load Balancer Gateway secara efektif terbuka untuk dunia segera setelah Anda mengizinkan permintaan masuk dan pemeriksaan kesehatan di grup keamanan target.
- Untuk tugas Fargate, Anda harus menggunakan versi platform `1.4.0` (Linux) atau `1.0.0` (Windows).
- Jika Anda mengalami masalah dengan layanan yang diaktifkan penyeimbang beban, lihat Memecahkan masalah penyeimbang beban layanan di Amazon ECS.
- Saat menggunakan tipe `instance` target, tugas dan penyeimbang beban Anda harus dalam VPC yang sama. Saat menggunakan tipe `ip` target, konektivitas lintas-VPC didukung.
- Pelestarian alamat IP klien Network Load Balancer kompatibel dengan target Fargate.
- Gunakan grup target unik untuk setiap layanan.

  Menggunakan grup target yang sama untuk beberapa layanan dapat menyebabkan masalah selama penerapan layanan.

Untuk selengkapnya tentang cara membuat Network Load Balancer, lihat Membuat Network Load Balancer di Network Load Balancer

> **Important**
>
> Jika definisi tugas layanan Anda menggunakan mode `awsvpc` jaringan (yang diperlukan untuk jenis peluncuran Fargate), Anda harus memilih `ip` sebagai jenis target, bukan. `instance` Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon.
> Anda tidak dapat mendaftarkan instans berdasarkan ID instans jika mereka memiliki jenis instance berikut: C1,,,,, CC1 CC2, G1 CG1 CG2, G2 CR1,,, M1, M2 HI1 HS1, M3, dan T1. Anda dapat mendaftarkan instans tipe ini berdasarkan alamat IP.

Menggunakan Load Balancer Gateway untuk Amazon ECS

Penyeimbang Beban Gateway beroperasi pada lapisan ketiga model Open Systems Interconnection (OSI), yaitu lapisan jaringan. Penyeimbang Beban Gateway mendengarkan semua paket IP di semua port dan meneruskan lalu lintas ke grup target yang ditentukan dalam aturan listener. Ini mempertahankan kelengketan aliran ke alat target tertentu menggunakan 5-tuple (untuk aliran) atau 3-tuple (untuk TCP/UDP aliran non-TCP/UDP). Misalnya, jika definisi kontainer tugas Anda menentukan port 80 untuk port kontainer NGINX, dan port 0 untuk port host, maka port host dipilih secara dinamis dari rentang port sementara instance kontainer (seperti 32768 hingga 61000 pada AMI Amazon ECS terbaru yang dioptimalkan). Saat tugas diluncurkan, kontainer NGINX terdaftar dengan Load Balancer Gateway sebagai ID instance dan kombinasi port, dan lalu lintas didistribusikan ke ID instance dan port yang sesuai dengan container tersebut. Pemetaan dinamis ini mengizinkan Anda memiliki banyak tugas dari satu layanan pada instans kontainer yang sama. Untuk informasi selengkapnya, lihat Apa itu Load Balancer Gateway di Gateway Load Balancers.

Untuk informasi tentang praktik terbaik untuk menyetel parameter guna mempercepat penerapan Anda, lihat:

- Optimalkan parameter pemeriksaan kesehatan penyeimbang beban untuk Amazon ECS
- Optimalkan parameter pengurasan koneksi penyeimbang beban untuk Amazon ECS

Pertimbangkan hal berikut saat menggunakan Gateway Load Balancer dengan Amazon ECS:

- Amazon ECS memerlukan peran IAM terkait layanan yang menyediakan izin yang diperlukan untuk mendaftarkan dan membatalkan pendaftaran target dengan penyeimbang beban Anda saat tugas dibuat dan dihentikan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.
- Untuk layanan dengan tugas menggunakan mode `awsvpc` jaringan, saat Anda membuat grup target untuk layanan Anda, Anda harus memilih `ip` sebagai jenis target, bukan`instance`. Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon.
- Untuk layanan dengan tugas yang menggunakan mode jaringan selain`awsvpc`, Gateway Load Balancers tidak didukung.
- Konfigurasi subnet penyeimbang beban Anda harus menyertakan semua Availability Zone tempat instans kontainer Anda berada.
- Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.
- Jika tugas layanan gagal dalam kriteria pemeriksaan kesehatan penyeimbang beban, tugas dihentikan dan dimulai ulang. Proses ini berlanjut hingga layanan Anda mencapai jumlah tugas berjalan yang diinginkan.
- Saat Anda menggunakan Load Balancer Gateway yang dikonfigurasi dengan alamat IP sebagai target, permintaan akan terlihat berasal dari alamat IP pribadi Gateway Load Balancers. Ini berarti bahwa layanan di balik Load Balancer Gateway secara efektif terbuka untuk dunia segera setelah Anda mengizinkan permintaan masuk dan pemeriksaan kesehatan di grup keamanan target.
- Untuk tugas Fargate, Anda harus menggunakan versi platform `1.4.0` (Linux) atau `1.0.0` (Windows).
- Jika Anda mengalami masalah dengan layanan yang diaktifkan penyeimbang beban, lihat Memecahkan masalah penyeimbang beban layanan di Amazon ECS.
- Saat menggunakan tipe `instance` target, tugas dan penyeimbang beban Anda harus dalam VPC yang sama. Saat menggunakan tipe `ip` target, konektivitas lintas-VPC didukung.

- Gunakan grup target unik untuk setiap layanan.

  Menggunakan grup target yang sama untuk beberapa layanan dapat menyebabkan masalah selama penerapan layanan.

Untuk informasi tentang cara membuat Load Balancer Gateway, lihat Memulai Gateway Load Balancers di Gateway Load Balancers

> **Important**
>
> Jika definisi tugas layanan Anda menggunakan mode `awsvpc` jaringan (yang diperlukan untuk jenis peluncuran Fargate), Anda harus memilih `ip` sebagai jenis target, bukan. `instance` Ini karena tugas yang menggunakan mode `awsvpc` jaringan dikaitkan dengan elastic network interface, bukan EC2 instance Amazon.
> Anda tidak dapat mendaftarkan instans berdasarkan ID instans jika mereka memiliki jenis instance berikut: C1,,,,, CC1 CC2, G1 CG1 CG2, G2 CR1,,, M1, M2 HI1 HS1, M3, dan T1. Anda dapat mendaftarkan instans tipe ini berdasarkan alamat IP.

Mendaftarkan beberapa grup sasaran dengan layanan Amazon ECS

Layanan Amazon ECS Anda dapat melayani lalu lintas dari beberapa penyeimbang beban dan mengekspos beberapa port seimbang beban saat Anda menentukan beberapa grup target dalam definisi layanan.

Untuk membuat layanan yang menentukan beberapa grup target, Anda harus membuat layanan menggunakan Amazon ECS API, SDK AWS CLI, atau templat. AWS CloudFormation Setelah layanan dibuat, Anda dapat melihat layanan dan grup target yang terdaftar dengan AWS Management Console. Anda harus menggunakan `UpdateService` untuk memodifikasi konfigurasi penyeimbang beban dari layanan yang ada.

Beberapa grup target dapat ditentukan dalam penentuan layanan menggunakan format berikut. Untuk sintaksis lengkap penentuan layanan, lihat Templat definisi layanan.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
```

```
      "containerName":"container_name",
      "containerPort":container_port
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"container_name",
      "containerPort":container_port
   }
]
```

Pertimbangan

Hal-hal berikut harus dipertimbangkan saat Anda menentukan beberapa grup target dalam penentuan layanan.

- Untuk layanan yang menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan, Anda tidak dapat melampirkan lebih dari lima grup target ke layanan.
- Penentuan beberapa grup target dalam penentuan layanan hanya didukung dalam kondisi berikut:
  - Layanan harus menggunakan Application Load Balancer atau Network Load Balancer.
  - Layanan harus menggunakan tipe pengendali deployment (`ECS`) pembaruan bergulir.
- Menentukan beberapa grup target didukung untuk layanan yang berisi tugas menggunakan jenis Fargate EC2 dan peluncuran.
- Saat membuat layanan yang menentukan beberapa grup target, peran terkait layanan Amazon ECS harus dibuat. Peran dibuat dengan menghilangkan parameter `role` dalam permintaan API, atau properti `Role` di AWS CloudFormation. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

Contoh penentuan layanan

Berikut adalah beberapa contoh kasus penggunaan untuk menentukan beberapa grup target dalam penentuan layanan. Untuk sintaksis penentuan layanan lengkap, lihat Templat definisi layanan.

Memiliki penyeimbang beban terpisah untuk lalu lintas internal dan eksternal

Dalam kasus penggunaan berikut, layanan menggunakan dua penyeimbang beban terpisah, satu untuk lalu lintas internal dan yang kedua untuk lalu lintas yang berhadapan dengan internet, untuk kontainer dan port yang sama.

```
"loadBalancers":[
   //Internal ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"nginx",
      "containerPort":8080
   },
   //Internet-facing ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"nginx",
      "containerPort":8080
   }
]
```

Mengekspos beberapa port dari wadah yang sama

Dalam kasus penggunaan berikut, layanan menggunakan satu penyeimbang beban, akan tetapi mengekspos beberapa port dari kontainer yang sama. Misalnya, kontainer Jenkins mungkin mengekspos port 8080 untuk antarmuka web Jenkins dan port 50000 untuk API.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"jenkins",
      "containerPort":8080
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"jenkins",
      "containerPort":50000
   }
]
```

Mengekspos port dari beberapa kontainer

Dalam kasus penggunaan berikut, layanan menggunakan satu penyeimbang beban dan dua grup target untuk mengekspos port dari kontainer terpisah.

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"webserver",
      "containerPort":80
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"database",
      "containerPort":3306
   }
]
```

Secara otomatis menskalakan layanan Amazon ECS Anda

Penskalaan otomatis adalah kemampuan untuk menambah atau mengurangi jumlah tugas yang diinginkan dalam layanan Amazon ECS Anda secara otomatis. Amazon ECS memanfaatkan layanan Application Auto Scaling untuk menyediakan fungsionalitas ini. Untuk informasi selengkapnya, lihat Panduan Pengguna Application Auto Scaling.

Amazon ECS menerbitkan CloudWatch metrik dengan penggunaan CPU dan memori rata-rata layanan Anda. Untuk informasi selengkapnya, lihat Metrik pemanfaatan layanan Amazon ECS. Anda dapat menggunakan metrik ini dan CloudWatch metrik lainnya untuk meningkatkan skala layanan Anda (menambahkan lebih banyak tugas) untuk menangani permintaan tinggi pada waktu puncak, dan untuk meningkatkan skala dalam layanan Anda (menjalankan lebih sedikit tugas) untuk mengurangi biaya selama periode pemanfaatan rendah.

Amazon ECS Service Auto Scaling mendukung jenis penskalaan otomatis berikut:

- Gunakan metrik target untuk menskalakan layanan Amazon ECSMeningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan nilai target untuk metrik tertentu. Hal ini

mirip dengan cara termostat mempertahankan suhu di rumah Anda. Anda mengatur suhu dan termostat melakukan sisanya.

- Gunakan kenaikan yang telah ditentukan berdasarkan CloudWatch alarm untuk menskalakan layanan Amazon ECSMeningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan serangkaian penyesuaian penskalaan, yang dikenal sebagai penyesuaian langkah, yang bervariasi berdasarkan ukuran pelanggaran alarm.
- Gunakan tindakan terjadwal untuk menskalakan layanan Amazon ECSMeningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan tanggal dan waktu.
- Gunakan pola historis untuk menskalakan layanan Amazon ECS dengan penskalaan prediktifMeningkatkan atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan analitik data beban historis untuk mendeteksi pola harian atau mingguan dalam arus lalu lintas.

Pertimbangan

Saat menggunakan kebijakan penskalaan, pertimbangkan hal berikut:

- Amazon ECS mengirimkan metrik dalam interval 1 menit ke. CloudWatch Metrik tidak tersedia sampai kluster dan layanan mengirimkan metrik CloudWatch, dan Anda tidak dapat membuat CloudWatch alarm untuk metrik yang tidak ada.
- Kebijakan penskalaan mendukung periode pendinginan. Ini adalah jumlah detik untuk menunggu hingga aktivitas penskalaan sebelumnya berlaku.
  - Dengan kebijakan penskalaan, tujuannya adalah untuk terus menerus (tetapi tidak berlebihan) menskalakan naik. Setelah Service Auto Scaling berhasil diskalakan menggunakan kebijakan penskalaan, maka mulai menghitung waktu cooldown. Kebijakan penskalaan tidak akan meningkatkan kapasitas yang diinginkan lagi kecuali jika skala yang lebih besar dimulai atau periode cooldown berakhir. Selama periode pendinginan penskalaan keluar berlaku, kapasitas yang ditambahkan dengan cara memulai aktivitas penskalaan keluar dihitung sebagai bagian dari kapasitas yang diinginkan untuk aktivitas penskalaan keluar berikutnya.
  - Untuk kejadian penskalaan kedalam, tujuannya adalah untuk melakukan penskalaan kedalam secara konservatif guna melindungi ketersediaan aplikasi Anda, sehingga aktivitas penskalaan kedalam diblokir hingga periode pendinginan berakhir. Namun, jika alarm lain memulai aktivitas scale-out selama periode cooldown scale-in, Service Auto Scaling segera menskalakan target. Dalam hal ini, periode pendinginan penskalaan kedalam berhenti dan tidak selesai.

- Penjadwal layanan menghormati jumlah yang diinginkan setiap saat, tetapi selama Anda memiliki kebijakan penskalaan aktif dan alarm pada layanan, Service Auto Scaling dapat mengubah hitungan yang diinginkan yang ditetapkan secara manual oleh Anda.
- Jika jumlah yang diinginkan layanan ditetapkan di bawah nilai kapasitas minimumnya, dan alarm memulai aktivitas scale-out, Service Auto Scaling menskalakan jumlah yang diinginkan hingga nilai kapasitas minimum dan kemudian melanjutkan skala sesuai kebutuhan, berdasarkan kebijakan penskalaan yang terkait dengan alarm. Namun, kegiatan penskalaan kedalam tidak menyesuaikan jumlah yang diinginkan, karena sudah di bawah nilai kapasitas minimal.
- Jika jumlah yang diinginkan layanan ditetapkan di atas nilai kapasitas maksimumnya, dan alarm memulai skala dalam aktivitas, Auto Scaling Service menskalakan hitungan yang diinginkan ke nilai kapasitas maksimum dan kemudian melanjutkan penskalaan sesuai kebutuhan, berdasarkan kebijakan penskalaan yang terkait dengan alarm. Namun, aktivitas penskalaan keluar tidak menyesuaikan jumlah yang diinginkan, karena sudah di atas nilai kapasitas maksimal.
- Selama aktivitas penskalaan, jumlah tugas yang sebenarnya berjalan dalam layanan adalah nilai yang digunakan Service Auto Scaling sebagai titik awalnya, sebagai lawan dari hitungan yang diinginkan. Inilah yang seharusnya menjadi kapasitas pemrosesan. Ini mencegah penskalaan berlebihan (runaway) yang mungkin tidak terpenuhi, misalnya, jika tidak ada sumber daya instance kontainer yang cukup untuk menempatkan tugas tambahan. Jika nantinya kapasitas instans kontainer tersedia, maka aktivitas penskalaan yang tertunda kemungkinan akan berhasil, dan kemudian aktivitas penskalaan lebih lanjut dapat diteruskan setelah periode pendinginan.
- Jika Anda ingin jumlah tugas Anda menjadi nol ketika tidak ada pekerjaan yang harus dilakukan, tetapkan kapasitas minimum 0. Dengan kebijakan penskalaan pelacakan target, ketika kapasitas aktual adalah 0 dan metrik menunjukkan bahwa ada permintaan beban kerja, Service Auto Scaling menunggu satu titik data dikirim sebelum penskalaan keluar. Dalam hal ini, penskalaan otomatis layanan akan menskalakan keluar berdasarkan jumlah minimal yang memungkinkan sebagai titik awal, dan kemudian melanjutkan penskalaan berdasarkan jumlah tugas berjalan yang aktual.
- Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Perilaku ini tidak berlaku untuk layanan Amazon ECS menggunakan pengontrol penerapan eksternal. Untuk informasi selengkapnya, lihat penskalaan otomatis dan deployment layanan.
- Anda memiliki beberapa opsi Application Auto Scaling untuk tugas Amazon ECS. Pelacakan target adalah mode termudah untuk digunakan. Dengan itu, yang perlu Anda lakukan adalah menetapkan nilai target untuk metrik, seperti pemanfaatan rata-rata CPU. Kemudian, auto scaler secara otomatis mengelola jumlah tugas yang diperlukan untuk mencapai nilai itu. Dengan

penskalaan langkah, Anda dapat bereaksi lebih cepat terhadap perubahan permintaan, karena Anda menentukan ambang batas tertentu untuk metrik penskalaan Anda, dan berapa banyak tugas yang harus ditambahkan atau dihapus saat ambang batas dilintasi. Dan, yang lebih penting, Anda dapat bereaksi sangat cepat terhadap perubahan permintaan dengan meminimalkan jumlah waktu alarm ambang batas dilanggar.

Untuk informasi selengkapnya tentang praktik terbaik untuk penskalaan otomatis servis, lihatMengoptimalkan penskalaan otomatis layanan Amazon ECS.

penskalaan otomatis dan deployment layanan

Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Perilaku ini tidak berlaku untuk layanan Amazon ECS menggunakan pengontrol penerapan eksternal. Jika Anda ingin menangguhkan proses penskalaan keluar saat deployment dalam progres, lakukan langkah-langkah berikut.

1. Panggil describe-scalable-targetsperintah, tentukan ID sumber daya layanan yang terkait dengan target yang dapat diskalakan di Application Auto Scaling (Contoh:). `service/default/sample-webapp` Catat outputnya. Anda akan membutuhkannya saat memanggil perintah berikutnya.
2. Panggil register-scalable-targetperintah, tentukan ID sumber daya, namespace, dan dimensi yang dapat diskalakan. Tentukan `true`, baik untuk `DynamicScalingInSuspended` maupun `DynamicScalingOutSuspended`.
3. Setelah penerapan selesai, Anda dapat memanggil register-scalable-targetperintah untuk melanjutkan penskalaan.

Untuk informasi lebih lanjut, lihat Menangguhkan dan melanjutkan penskalaan untuk Application Auto Scaling.

Gunakan metrik target untuk menskalakan layanan Amazon ECS

Dengan kebijakan penskalaan pelacakan target, Anda memilih metrik dan menetapkan nilai target. Auto Scaling Amazon ECS Service membuat dan mengelola CloudWatch alarm yang mengontrol kebijakan penskalaan dan menghitung penyesuaian penskalaan berdasarkan metrik dan nilai target. Kebijakan penskalaan menambahkan atau menghapus tugas layanan sebagaimana diperlukan untuk menjaga metrik pada, atau mendekati nilai target yang ditentukan. Selain menjaga agar metrik tetap

mendekati nilai target, kebijakan penskalaan pelacakan target juga menyesuaikan dengan fluktuasi metrik akibat pola beban yang berfluktuasi, serta meminimalkan fluktuasi yang cepat dalam jumlah tugas yang berjalan di layanan Anda.

Kebijakan pelacakan target menghilangkan kebutuhan untuk menentukan CloudWatch alarm dan penyesuaian penskalaan secara manual. Amazon ECS menangani ini secara otomatis berdasarkan target yang Anda tetapkan.

Pertimbangkan hal berikut saat menggunakan kebijakan pelacakan target:

- Kebijakan penskalaan pelacakan target mengasumsikan bahwa penskalaan ke luar harus dilakukan saat metrik yang ditentukan berada di atas nilai target. Anda tidak dapat menggunakan kebijakan penskalaan pelacakan target untuk menskalakan keluar jika metrik yang ditentukan berada di bawah nilai target.
- Kebijakan penskalaan pelacakan target tidak melakukan penskalaan saat metrik yang ditentukan tidak memiliki data yang mencukupi. Kebijakan penskalaan pelacakan target tidak melakukan penskalaan ke dalam karena data yang tidak mencukupi tidak ditafsirkan sebagai pemanfaatan yang rendah.
- Anda mungkin melihat kesenjangan antara nilai target dan titik data metrik yang aktual. Ini karena Service Auto Scaling selalu bertindak konservatif dengan membulatkan ke atas atau ke bawah ketika menentukan berapa banyak kapasitas untuk menambah atau menghapus. Hal ini mencegahnya menambahkan kapasitas yang tidak mencukupi atau membuang terlalu banyak kapasitas.
- Untuk memastikan ketersediaan aplikasi, layanan menskalakan keluar secara proporsional ke dalam metrik secepat mungkin, namun penskalaan kedalam meningkat secara bertahap.
- Application Auto Scaling menonaktifkan proses scale-in saat penerapan Amazon ECS sedang berlangsung. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Perilaku ini tidak berlaku untuk layanan Amazon ECS menggunakan pengontrol penerapan eksternal. Untuk informasi selengkapnya, lihat penskalaan otomatis dan deployment layanan.
- Anda dapat memiliki beberapa kebijakan penskalaan pelacakan target untuk layanan Amazon ECS, asalkan masing-masing menggunakan metrik yang berbeda. Tujuan Service Auto Scaling adalah untuk selalu memprioritaskan ketersediaan, sehingga perilakunya berbeda tergantung pada apakah kebijakan pelacakan target siap untuk skala atau skala. Ini akan meningkatkan skala layanan jika ada kebijakan pelacakan target yang siap untuk diskalakan, tetapi akan menskalakan hanya jika semua kebijakan pelacakan target (dengan bagian penskalaan dihidupkan) siap untuk diskalakan.

- Jangan mengedit atau menghapus CloudWatch alarm yang dikelola Service Auto Scaling untuk kebijakan penskalaan pelacakan target. Service Auto Scaling akan menghapus alarm secara otomatis saat Anda menghapus kebijakan penskalaan.
- `ALBRequestCountPerTarget`Metrik untuk kebijakan penskalaan pelacakan target tidak didukung untuk jenis blue/green penerapan.

Untuk informasi lebih lanjut tentang kebijakan penskalaan pelacakan target, lihat Kebijakan penskalaan pelacakan target dalam Panduan Pengguna Application Auto Scaling.

Membuat kebijakan penskalaan pelacakan target untuk penskalaan otomatis layanan Amazon ECS

Buat kebijakan penskalaan pelacakan target agar Amazon ECS meningkatkan atau mengurangi jumlah tugas yang diinginkan di layanan Anda secara otomatis. Pelacakan target bekerja dari nilai metrik target.

Konsol

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS.
2. Tentukan metrik yang akan digunakan untuk kebijakan. Metrik berikut tersedia:
   - ECSServiceRata-rata CPUUtilization — Pemanfaatan CPU rata-rata yang harus digunakan layanan.
   - ECSServiceAverageMemoryUtilization— Pemanfaatan memori rata-rata yang harus digunakan layanan.
   - ALBRequestCountPerTarget— Jumlah rata-rata permintaan per menit yang harus diterima tugas yang idealnya diterima.
3. Buka konsol di https://console.aws.amazon.com/ecs/v2.
4. Pada halaman Clusters, pilih cluster.
5. Pada halaman detail cluster, di bagian Layanan, dan kemudian pilih layanan.

   Halaman detail layanan muncul.
6. Pilih Atur jumlah tugas.
7. Di bawah jumlah tugas layanan Amazon ECS, pilih Gunakan penskalaan otomatis.

   Bagian Penghitungan tugas muncul.

a. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

b. Untuk Maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

c. Pilih Simpan.

Halaman kebijakan muncul.

8. Pilih Buat kebijakan penskalaan.

Halaman Buat kebijakan akan muncul.

9. Untuk Jenis kebijakan penskalaan, pilih Pelacakan target.
10. Untuk nama Kebijakan, masukkan nama kebijakan.
11. Untuk jenis Metrik, pilih metrik Anda dari daftar opsi.
12. Untuk pemanfaatan Target, masukkan nilai target untuk persentase tugas yang harus dipertahankan Amazon ECS. Service auto scaling meningkatkan kapasitas Anda hingga pemanfaatan rata-rata berada pada target pemanfaatan, atau hingga mencapai jumlah maksimum tugas yang Anda tentukan.
13. Di bawah Pengaturan Tambahan, lakukan hal berikut

    a. Untuk periode cooldown Scale-in, masukkan jumlah waktu dalam hitungan detik setelah aktivitas scale-in selesai sebelum aktivitas scale-in lainnya dapat dimulai.

    b. Untuk periode cooldown Scale-out, masukkan jumlah waktu dalam hitungan detik untuk menunggu aktivitas scale-out sebelumnya diterapkan.

    c. Untuk hanya membuat kebijakan penskalaan, pilih Nonaktifkan penskalaan.

14. Pilih Buat kebijakan penskalaan.

AWS CLI

1. Daftarkan layanan Amazon ECS Anda sebagai target yang dapat diskalakan menggunakan perintah. register-scalable-target
2. Buat kebijakan penskalaan menggunakan put-scaling-policyperintah.

Gunakan kenaikan yang telah ditentukan berdasarkan CloudWatch alarm untuk menskalakan layanan Amazon ECS

Dengan kebijakan penskalaan langkah, Anda membuat dan mengelola CloudWatch alarm yang menjalankan proses penskalaan. Saat alarm dilanggar, Amazon ECS memulai kebijakan penskalaan yang terkait dengan alarm tersebut. Kebijakan penskalaan langkah menskalakan tugas menggunakan serangkaian penyesuaian, yang dikenal sebagai penyesuaian langkah. Ukuran penyesuaian bervariasi berdasarkan besarnya pelanggaran alarm.

- Jika pelanggaran melebihi ambang pertama, Amazon ECS menerapkan penyesuaian langkah pertama.
- Jika pelanggaran melebihi ambang kedua, Amazon ECS menerapkan penyesuaian langkah kedua, dan seterusnya.

Kami sangat menyarankan agar Anda menggunakan kebijakan penskalaan pelacakan target untuk menskalakan metrik seperti pemanfaatan CPU rata-rata atau jumlah permintaan rata-rata per target. Metrik yang menurun ketika kapasitas meningkat dan meningkat ketika kapasitas menurun dapat digunakan untuk skala proporsional atau dalam jumlah tugas menggunakan pelacakan target. Ini membantu memastikan bahwa Amazon ECS mengikuti kurva permintaan untuk aplikasi Anda dengan cermat.

Buat kebijakan penskalaan langkah untuk penskalaan otomatis layanan Amazon ECS

Buat kebijakan penskalaan langkah agar Amazon ECS menambah atau mengurangi jumlah tugas yang diinginkan dalam layanan Anda secara otomatis. Penskalaan langkah berjalan berdasarkan serangkaian penyesuaian penskalaan, yang dikenal sebagai penyesuaian langkah, yang bervariasi berdasarkan ukuran pelanggaran alarm.

Konsol

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS.
2. Tentukan metrik yang akan digunakan untuk kebijakan. Metrik berikut tersedia:
   - ECSServiceRata-rata CPUUtilization — Pemanfaatan CPU rata-rata yang harus digunakan layanan.

- ECSServiceAverageMemoryUtilization— Pemanfaatan memori rata-rata yang harus digunakan layanan.
- ALBRequestCountPerTarget— Jumlah rata-rata permintaan per menit yang harus diterima tugas yang idealnya diterima.

3. Buat CloudWatch alarm untuk metrik. Untuk informasi selengkapnya, lihat Membuat CloudWatch alarm berdasarkan ambang batas statis di Panduan CloudWatch Pengguna Amazon.
4. Buka konsol di https://console.aws.amazon.com/ecs/v2.
5. Pada halaman Clusters, pilih cluster.
6. Pada halaman detail cluster, di bagian Layanan, dan kemudian pilih layanan.

   Halaman detail layanan muncul.
7. Pilih Atur jumlah tugas.
8. Di bawah jumlah tugas layanan Amazon ECS, pilih Gunakan penskalaan otomatis.

   Bagian Penghitungan tugas muncul.

   a. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.
   b. Untuk Maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.
   c. Pilih Simpan.

      Halaman kebijakan muncul.
9. Pilih Buat kebijakan penskalaan.

   Halaman Buat kebijakan akan muncul.
10. Untuk jenis kebijakan Penskalaan, pilih Penskalaan Langkah.
11. Konfigurasikan properti scaling-out. Di bawah Langkah-langkah untuk menambahkan tugas lakukan hal berikut:

    a. Untuk nama Kebijakan, masukkan nama kebijakan.
    b. Untuk nama CloudWatch alarm, pilih CloudWatch alarm.
    c. Untuk jenis agregasi metrik, pilih cara membandingkan metrik yang dipilih dengan ambang batas yang ditentukan.

d. Untuk tipe Djustment A, pilih apakah penyesuaian didasarkan pada perubahan jumlah tugas, atau perubahan persentase tugas.

e. Untuk Tindakan yang harus diambil, masukkan nilai untuk tindakan apa yang harus diambil.

Pilih Tambahkan langkah untuk menambahkan tindakan tambahan.

12. Konfigurasikan properti penskalaan. Di bawah Langkah-langkah untuk menghapus tugas, lakukan hal berikut:

a. Untuk nama Kebijakan, masukkan nama kebijakan.

b. Untuk nama CloudWatch alarm, pilih CloudWatch alarm.

c. Untuk jenis agregasi metrik, pilih cara membandingkan metrik yang dipilih dengan ambang batas yang ditentukan.

d. Untuk jenis Penyesuaian, pilih apakah penyesuaian didasarkan pada perubahan jumlah tugas, atau perubahan persentase tugas.

e. Untuk Tindakan yang harus diambil, masukkan nilai untuk tindakan apa yang harus diambil.

Pilih Tambahkan langkah untuk menambahkan tindakan tambahan.

13. Untuk periode Cooldown, masukkan jumlah waktu, dalam detik, untuk menunggu aktivitas penskalaan sebelumnya diterapkan. Untuk kebijakan add, ini adalah waktu setelah aktivitas scale-out yang kebijakan penskalaan memblokir aktivitas scale-in dan membatasi berapa banyak tugas yang dapat diskalakan pada suatu waktu. Untuk kebijakan penghapusan, ini adalah waktu setelah aktivitas penskalaan yang harus diteruskan sebelum aktivitas penskalaan lainnya dapat dimulai.

14. Pilih Buat kebijakan penskalaan.

AWS CLI

1. Daftarkan layanan Amazon ECS Anda sebagai target yang dapat diskalakan menggunakan perintah. register-scalable-target

2. Buat kebijakan penskalaan menggunakan put-scaling-policyperintah.

Gunakan tindakan terjadwal untuk menskalakan layanan Amazon ECS

Dengan penskalaan terjadwal, Anda dapat mengatur penskalaan otomatis untuk aplikasi berdasarkan perubahan beban yang dapat diprediksi dengan membuat tindakan terjadwal yang menambah atau

mengurangi jumlah tugas pada waktu tertentu. Ini memungkinkan Anda menskalakan aplikasi secara proaktif agar sesuai dengan perubahan beban yang dapat diprediksi.

Tindakan penskalaan terjadwal ini memungkinkan Anda mengoptimalkan biaya dan kinerja. Aplikasi Anda memiliki jumlah tugas yang cukup untuk menangani puncak lalu lintas pertengahan minggu, tetapi tidak menyediakan jumlah tugas secara berlebihan di waktu lain.

Anda dapat menggunakan kebijakan penskalaan dan penskalaan terjadwal bersama-sama untuk mendapatkan manfaat dari pendekatan proaktif dan reaktif untuk penskalaan. Setelah tindakan penskalaan terjadwal berjalan, kebijakan penskalaan dapat terus membuat keputusan tentang apakah akan menskalakan jumlah tugas lebih lanjut. Ini membantu Anda memastikan bahwa Anda memiliki cukup banyak tugas untuk menangani beban aplikasi Anda. Sementara skala aplikasi Anda sesuai dengan permintaan, kapasitas saat ini harus termasuk dalam jumlah tugas minimum dan maksimum yang ditetapkan oleh tindakan terjadwal Anda.

Anda dapat mengonfigurasi penskalaan jadwal menggunakan. AWS CLI Untuk informasi selengkapnya tentang penskalaan terjadwal, lihat Penskalaan Terjadwal di Panduan Pengguna Application Auto Scaling.

Buat tindakan terjadwal untuk penskalaan otomatis layanan Amazon ECS

Buat tindakan terjadwal agar Amazon ECS menambah atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan tanggal dan waktu.

Konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan muncul.
4. Pilih Service auto scaling.

   Halaman penskalaan otomatis layanan muncul.
5. Jika Anda belum mengonfigurasi penskalaan otomatis layanan, pilih Setel jumlah tugas.

   Bagian penghitungan tugas layanan Amazon ECS muncul.

   Di bawah jumlah tugas layanan Amazon ECS, pilih Gunakan penskalaan otomatis layanan untuk menyesuaikan jumlah tugas yang diinginkan layanan Anda.

Bagian Penghitungan tugas muncul.

a. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

b. Untuk Maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

c. Pilih Pilih Simpan.

Halaman kebijakan muncul.

6. Pilih Tindakan terjadwal, lalu pilih Buat.

Halaman tindakan Buat Terjadwal muncul.

7. Untuk nama Action, masukkan nama unik.

8. Untuk zona waktu, pilih zona waktu.

Semua zona waktu yang tercantum berasal dari database Zona Waktu IANA. Untuk informasi selengkapnya, lihat Daftar zona waktu database tz.

9. Untuk Waktu mulai, masukkan Tanggal dan Waktu tindakan dimulai.

Jika Anda memilih jadwal berulang, waktu mulai menentukan kapan tindakan terjadwal pertama dalam seri berulang berjalan.

10. Untuk Recurrence, pilih salah satu opsi yang tersedia.

- Untuk menskalakan jadwal berulang, pilih seberapa sering Amazon ECS menjalankan tindakan terjadwal.
  - Jika Anda memilih opsi yang dimulai dengan Rate, ekspresi cron dibuat untuk Anda.
  - Jika Anda memilih Cron, masukkan ekspresi cron yang menentukan kapan harus melakukan tindakan.
- Untuk skala hanya sekali, pilih Sekali.

11. Di bawah Penyesuaian tugas, lakukan hal berikut:

- Untuk Minimum, masukkan jumlah tugas minimum yang harus dijalankan oleh layanan.
- Untuk Maksimum, masukkan jumlah tugas maksimum yang harus dijalankan layanan.

12. Pilih Buat tindakan terjadwal.

CLI

Gunakan AWS CLI sebagai berikut untuk mengonfigurasi kebijakan penskalaan terjadwal untuk layanan Anda. Ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Contoh: Untuk skala satu kali saja

Gunakan put-scheduled-actionperintah berikut dengan `--start-time "YYYY-MM-DDThh:mm:ssZ"` dan dan salah satu atau keduanya `--MaxCapacity` opsi `--MinCapacity` dan.

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-one-time-schedule \
  --start-time 2021-01-30T12:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

Contoh: Untuk menjadwalkan penskalaan pada jadwal berulang

Gunakan perintah berikut put-scheduled-action. Ganti *user input* dengan nilai-nilai Anda.

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-recurring-action \
  --schedule "rate(5 hours)" \
  --start-time 2021-01-30T12:00:00 \
  --end-time 2021-01-31T22:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

Jadwal pengulangan yang ditentukan berjalan berdasarkan zona waktu UTC. Untuk menentukan zona waktu yang berbeda, sertakan `--time-zone` opsi dan nama zona waktu IANA, seperti pada contoh berikut.

```
--time-zone "America/New_York"
```

Untuk informasi selengkapnya, lihat Daftar zona waktu database tz.

Gunakan pola historis untuk menskalakan layanan Amazon ECS dengan penskalaan prediktif

Penskalaan prediktif melihat data beban masa lalu lintas dari arus lalu lintas untuk menganalisis pola harian atau mingguan. Kemudian menggunakan analisis ini untuk mengantisipasi kebutuhan masa depan dan secara proaktif meningkatkan tugas dalam layanan Anda sesuai kebutuhan.

Penskalaan otomatis prediktif paling berguna dalam situasi berikut.

- Lalu lintas siklus - Peningkatan penggunaan sumber daya selama jam kerja reguler, dan penurunan penggunaan sumber daya selama malam hari dan akhir pekan.
- Pola on-and-off beban kerja berulang - Contohnya termasuk pemrosesan batch, pengujian, atau analisis data berkala.
- Aplikasi dengan waktu inisialisasi yang lama - Ini dapat memengaruhi kinerja aplikasi selama peristiwa penskalaan yang menyebabkan latensi yang nyata.

Jika aplikasi Anda membutuhkan waktu lama untuk diinisialisasi dan lalu lintas meningkat dalam pola reguler, Anda harus mempertimbangkan untuk menggunakan penskalaan prediktif. Ini membantu Anda menskalakan lebih cepat dengan secara proaktif meningkatkan jumlah tugas untuk beban yang diperkirakan, alih-alih menggunakan kebijakan penskalaan dinamis, seperti Pelacakan Target atau Penskalaan Langkah saja. Dengan membantu Anda menghindari kemungkinan penyediaan berlebihan jumlah tugas, penskalaan prediktif juga berpotensi menghemat uang Anda.

Misalnya, pertimbangkan aplikasi yang memiliki penggunaan tinggi selama jam kerja dan penggunaan rendah dalam semalam. Pada awal setiap hari kerja, penskalaan prediktif dapat meningkatkan tugas sebelum masuknya lalu lintas pertama. Ini membantu aplikasi Anda mempertahankan ketersediaan dan kinerja tinggi saat beralih dari periode pemanfaatan yang lebih rendah ke periode pemanfaatan yang lebih tinggi. Anda tidak perlu menunggu penskalaan dinamis untuk bereaksi terhadap perubahan lalu lintas. Anda juga tidak perlu menghabiskan waktu meninjau pola pemuatan aplikasi Anda dan mencoba menjadwalkan jumlah tugas yang tepat menggunakan penskalaan terjadwal.

Penskalaan prediktif adalah kemampuan tingkat layanan yang menskalakan tugas layanan Anda secara independen dari penskalaan kapasitas komputasi yang mendasarinya (misalnya, atau Fargate). EC2 Untuk Fargate, AWS mengelola dan secara otomatis menskalakan kapasitas yang mendasarinya berdasarkan persyaratan tugas. Untuk EC2 kapasitas, Anda dapat menggunakan penyedia kapasitas grup Auto Scaling untuk menskalakan EC2 instans dasar secara otomatis berdasarkan persyaratan penskalaan tugas Anda.

Daftar Isi

- Cara kerja penskalaan prediktif di Amazon ECS
- Membuat kebijakan penskalaan prediktif untuk penskalaan otomatis layanan Amazon ECS
- Evaluasi kebijakan penskalaan prediktif Anda untuk Amazon ECS
- Menggunakan tindakan terjadwal untuk mengganti nilai perkiraan untuk Amazon ECS
- Kebijakan penskalaan prediktif lanjutan menggunakan metrik khusus untuk Amazon ECS

Cara kerja penskalaan prediktif di Amazon ECS

Di sini Anda dapat mempelajari tentang pertimbangan untuk menggunakan penskalaan prediktif, cara kerjanya, dan apa batasannya.

Pertimbangan untuk menggunakan penskalaan prediktif

- Anda ingin memastikan penskalaan prediktif cocok untuk beban kerja Anda. Anda dapat memeriksanya dengan mengonfigurasi kebijakan penskalaan dalam mode perkiraan saja dan melihat apa yang direkomendasikan konsol. Anda harus mengevaluasi perkiraan dan rekomendasi sebelum mulai menggunakan penskalaan prediktif.
- Sebelum penskalaan prediktif dapat memulai peramalan, dibutuhkan setidaknya 24 jam data historis. Semakin banyak data historis yang tersedia, semakin efektif ramalannya, dengan dua minggu ideal. Anda juga harus menunggu 24 jam sebelum penskalaan prediktif dapat menghasilkan perkiraan baru saat Anda menghapus layanan Amazon ECS dan membuat yang baru. Salah satu cara untuk mempercepat ini adalah dengan menggunakan metrik khusus untuk menggabungkan metrik di seluruh layanan Amazon ECS lama dan baru.
- Pilih metrik pemuatan yang secara akurat mewakili beban penuh pada aplikasi Anda dan merupakan aspek aplikasi Anda yang paling penting untuk diskalakan.
- Penskalaan dinamis dengan penskalaan prediktif membantu Anda mengikuti permintaan aplikasi dengan cermat, sehingga Anda dapat menskalakan selama jeda dan meningkatkan skala selama peningkatan lalu lintas yang tidak terduga. Ketika beberapa kebijakan penskalaan aktif, setiap kebijakan menentukan jumlah tugas yang diinginkan secara independen, dan jumlah tugas yang diinginkan diatur secara maksimal.
- Anda dapat menggunakan penskalaan prediktif di samping kebijakan penskalaan dinamis Anda, seperti pelacakan target atau penskalaan langkah, sehingga aplikasi Anda menskalakan berdasarkan pola real-time dan historis. Dengan sendirinya, penskalaan prediktif tidak menskalakan tugas Anda.

- Jika Anda menggunakan peran khusus saat memanggil `register-scalable-target` API, Anda mungkin mendapatkan kesalahan yang mengatakan kebijakan penskalaan prediktif hanya dapat berfungsi dengan SLR diaktifkan. Dalam hal ini Anda harus menelepon `register-scalable-target` lagi tetapi tanpa role-arn. Gunakan SLR saat mendaftarkan target yang dapat diskalakan dan panggil API. `put-scaling-policy`

Cara kerja penskalaan prediktif

Anda menggunakan penskalaan prediktif dengan membuat kebijakan penskalaan prediktif yang menentukan CloudWatch metrik untuk dipantau dan dianalisis. Penskalaan prediktif harus memiliki setidaknya 24 jam data untuk mulai memperkirakan nilai masa depan.

Setelah Anda membuat kebijakan, penskalaan prediktif mulai menganalisis data metrik hingga 14 hari terakhir untuk mengidentifikasi pola. Analisis ini digunakan untuk menghasilkan 48 jam perkiraan persyaratan per jam berikutnya. CloudWatch Data terbaru digunakan untuk memperbarui perkiraan setiap enam jam. Saat data baru masuk, penskalaan prediktif terus meningkatkan akurasi prakiraan masa depan.

Saat Anda pertama kali mengaktifkan penskalaan prediktif, penskalaan ini berjalan dalam mode perkiraan saja. Ini menghasilkan perkiraan dalam mode ini, tetapi tidak menskalakan layanan Amazon ECS Anda berdasarkan perkiraan tersebut. Ini berarti Anda dapat mengevaluasi keakuratan dan kesesuaian ramalan. Anda melihat data perkiraan dengan menggunakan operasi `GetPredictiveScalingForecast` API atau AWS Management Console.

Saat Anda memutuskan untuk mulai menggunakan penskalaan prediktif, alihkan kebijakan penskalaan ke mode prakiraan dan skala. Berikut ini terjadi saat dalam mode ini.

Layanan Amazon ECS Anda diskalakan pada awal setiap jam berdasarkan perkiraan untuk jam itu, secara default. Anda dapat memilih untuk memulai lebih awal dengan menggunakan `SchedulingBufferTime` properti dalam operasi `PutScalingPolicy` API. Ini membuat tugas baru diluncurkan di depan permintaan yang diperkirakan dan memberi mereka waktu untuk boot dan siap menangani lalu lintas.

Batas tugas maksimum

Saat Anda mendaftarkan layanan Amazon ECS untuk penskalaan, Anda menentukan jumlah tugas maksimum yang dapat diluncurkan per layanan. Secara default, ketika kebijakan penskalaan ditetapkan, mereka tidak dapat meningkatkan jumlah tugas yang lebih tinggi dari batas maksimumnya.

Atau, Anda dapat mengizinkan jumlah tugas maksimum layanan ditingkatkan secara otomatis jika perkiraan mendekati atau melebihi jumlah tugas maksimum layanan Amazon ECS.

> **Warning**
>
> Berhati-hatilah saat memungkinkan jumlah tugas maksimum ditingkatkan secara otomatis. Hal ini dapat menyebabkan lebih banyak tugas yang diluncurkan daripada yang dimaksudkan, jika peningkatan jumlah tugas maksimum tidak dipantau dan dikelola. Peningkatan jumlah tugas maksimum kemudian menjadi jumlah tugas maksimum normal baru untuk layanan Amazon ECS hingga Anda memperbaruinya secara manual. Jumlah tugas maksimum tidak secara otomatis berkurang kembali ke maksimum semula.

Wilayah yang didukung

- Timur AS (N. Virginia)
- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Tiongkok (Beijing)
- Tiongkok (Ningxia)
- Eropa (Frankfurt)
- Eropa (Irlandia)

- Eropa (London)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Stockholm)
- Timur Tengah (Bahrain)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (AS-Timur)
- AWS GovCloud (AS-Barat)

Membuat kebijakan penskalaan prediktif untuk penskalaan otomatis layanan Amazon ECS

Buat kebijakan penskalaan prediktif agar Amazon ECS menambah atau mengurangi jumlah tugas yang dijalankan layanan Anda berdasarkan data historis.

> **Note**
>
> Layanan baru perlu menyediakan setidaknya 24 jam data sebelum perkiraan dapat dihasilkan.

Konsol

1. Selain izin IAM standar untuk membuat dan memperbarui layanan, Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS.
2. Tentukan metrik yang akan digunakan untuk kebijakan. Metrik berikut tersedia:

   - ECSServiceRata-rata CPUUtilization — Pemanfaatan CPU rata-rata yang harus digunakan layanan.
   - ECSServiceAverageMemoryUtilization— Pemanfaatan memori rata-rata yang harus digunakan layanan.
   - ALBRequestCountPerTarget— Jumlah rata-rata permintaan per menit yang harus diterima tugas yang idealnya diterima.

   Anda dapat menggunakan metrik khusus. Anda perlu menentukan nilai-nilai berikut:

- Load - metrik yang secara akurat mewakili beban penuh pada aplikasi Anda dan merupakan aspek aplikasi Anda yang paling penting untuk diskalakan.

- Metrik penskalaan - prediktor terbaik untuk seberapa banyak pemanfaatan yang ideal untuk aplikasi Anda.

3. Buka konsol di https://console.aws.amazon.com/ecs/v2.
4. Pada halaman Clusters, pilih cluster.
5. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan muncul.
6. Pilih Service auto scaling dan kemudian pilih Atur jumlah tugas.
7. Di bawah jumlah tugas layanan Amazon ECS, pilih Gunakan penskalaan otomatis.

   Bagian Penghitungan tugas muncul.

   a. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.
   b. Untuk Maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.
   c. Pilih Simpan.

      Halaman kebijakan muncul.
8. Pilih Buat kebijakan penskalaan.

   Halaman Buat kebijakan akan muncul.
9. Untuk jenis kebijakan Penskalaan, pilih Penskalaan Prediktif.
10. Untuk nama Kebijakan, masukkan nama kebijakan.
11. Untuk pasangan Metrik, pilih metrik Anda dari daftar opsi.

    Jika Anda memilih jumlah permintaan Application Load Balancer per target, maka pilih grup target di grup Target. Jumlah permintaan Application Load Balancer per target hanya didukung jika Anda telah melampirkan grup target Application Load Balancer untuk layanan Anda.

    Jika Anda memilih Pasangan metrik kustom, pilih metrik individual dari daftar untuk Metrik beban dan metrik Penskalaan.

12. Untuk pemanfaatan Target, masukkan nilai target untuk persentase tugas yang harus dipertahankan Amazon ECS. Service auto scaling meningkatkan kapasitas Anda hingga pemanfaatan rata-rata berada pada target pemanfaatan, atau hingga mencapai jumlah maksimum tugas yang Anda tentukan.
13. Pilih Buat kebijakan penskalaan.

AWS CLI

Gunakan AWS CLI sebagai berikut untuk mengonfigurasi kebijakan penskalaan prediktif untuk layanan Amazon ECS Anda. Ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Untuk informasi selengkapnya tentang CloudWatch metrik yang dapat Anda tentukan, lihat PredictiveScalingMetricSpecificationdi Referensi API Amazon EC2 Auto Scaling.

Contoh 1: Kebijakan penskalaan prediktif dengan memori yang telah ditentukan sebelumnya.

Berikut ini adalah contoh kebijakan dengan konfigurasi memori yang telah ditentukan.

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceMemoryUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

Contoh berikut mengilustrasikan pembuatan kebijakan dengan menjalankan put-scaling-policyperintah dengan file konfigurasi yang ditentukan.

```
aws application-autoscaling put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
```

```
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

Jika berhasil, perintah ini mengembalikan ARN kebijakan.

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
    "Alarms": []
}
```

Contoh 2: Kebijakan penskalaan prediktif dengan CPU yang telah ditentukan sebelumnya.

Berikut ini adalah contoh kebijakan dengan konfigurasi CPU yang telah ditentukan.

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 0.00000004,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceCPUUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

Contoh berikut mengilustrasikan pembuatan kebijakan dengan menjalankan put-scaling-policyperintah dengan file konfigurasi yang ditentukan.

```
aws aas put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
```

```
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

Jika berhasil, perintah ini mengembalikan ARN kebijakan.

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
    "Alarms": []
}
```

Evaluasi kebijakan penskalaan prediktif Anda untuk Amazon ECS

Sebelum Anda menggunakan kebijakan penskalaan prediktif untuk menskalakan layanan Anda, tinjau rekomendasi dan data lain untuk kebijakan Anda di konsol Amazon ECS. Ini penting karena Anda tidak ingin kebijakan penskalaan prediktif untuk menskalakan kapasitas aktual Anda sampai Anda tahu bahwa prediksinya akurat.

Jika layanan ini baru, biarkan 24 jam untuk membuat perkiraan pertama.

Saat AWS membuat perkiraan, ia menggunakan data historis. Jika layanan Anda belum memiliki banyak data historis terbaru, penskalaan prediktif mungkin mengisi ulang perkiraan sementara dengan agregat yang dibuat dari agregat historis yang tersedia saat ini. Prakiraan diisi kembali hingga dua minggu sebelum tanggal pembuatan kebijakan.

Lihat rekomendasi penskalaan prediktif Anda

Untuk analisis yang efektif, penskalaan otomatis servis harus memiliki setidaknya dua kebijakan penskalaan prediktif untuk dibandingkan. (Namun, Anda masih dapat meninjau temuan untuk satu kebijakan.) Saat membuat beberapa kebijakan, Anda dapat mengevaluasi kebijakan yang menggunakan satu metrik terhadap kebijakan yang menggunakan metrik berbeda. Anda juga dapat mengevaluasi dampak dari nilai target dan kombinasi metrik yang berbeda. Setelah kebijakan penskalaan prediktif dibuat, Amazon ECS segera mulai mengevaluasi kebijakan mana yang akan melakukan pekerjaan yang lebih baik untuk menskalakan grup Anda.

Untuk melihat rekomendasi Anda di konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

Halaman detail layanan muncul.

4. Pilih Service auto scaling.
5. Pilih kebijakan penskalaan prediktif, lalu pilih Tindakan, Penskalaan Prediktif, Lihat rekomendasi.

   Anda dapat melihat detail tentang kebijakan bersama dengan rekomendasi kami. Rekomendasi memberi tahu Anda apakah kebijakan penskalaan prediktif melakukan pekerjaan yang lebih baik daripada tidak menggunakannya.

   Jika Anda tidak yakin apakah kebijakan penskalaan prediktif sesuai untuk grup Anda, tinjau kolom Dampak ketersediaan dan dampak Biaya untuk memilih kebijakan yang tepat. Informasi untuk setiap kolom memberi tahu Anda apa dampak kebijakan tersebut.

   - Dampak ketersediaan: Menjelaskan apakah kebijakan akan menghindari dampak negatif terhadap ketersediaan dengan menyediakan tugas yang cukup untuk menangani beban kerja, dibandingkan dengan tidak menggunakan kebijakan.
   - Dampak biaya: Menjelaskan apakah kebijakan akan menghindari dampak negatif pada biaya Anda dengan tidak menyediakan tugas yang berlebihan, dibandingkan dengan tidak menggunakan kebijakan. Dengan terlalu banyak penyediaan, layanan Anda kurang dimanfaatkan atau tidak digunakan, yang hanya menambah dampak biaya.

   Jika Anda memiliki beberapa kebijakan, maka tag prediksi terbaik akan ditampilkan di samping nama kebijakan yang memberikan manfaat ketersediaan terbanyak dengan biaya lebih rendah. Lebih banyak bobot diberikan untuk dampak ketersediaan.
6. (Opsional) Untuk memilih periode waktu yang diinginkan untuk hasil rekomendasi, pilih nilai yang Anda inginkan dari dropdown periode Evaluasi: 2 hari, 1 minggu, atau 2 minggu. Secara default, periode evaluasi adalah dua minggu terakhir. Periode evaluasi yang lebih lama memberikan lebih banyak poin data ke hasil rekomendasi. Namun, menambahkan lebih banyak titik data mungkin tidak meningkatkan hasil jika pola beban Anda telah berubah, seperti setelah periode permintaan yang luar biasa. Dalam hal ini, Anda bisa mendapatkan rekomendasi yang lebih fokus dengan melihat data yang lebih baru.

> **Note**
>
> Rekomendasi dibuat hanya untuk kebijakan yang hanya dalam mode Forecast. Fitur rekomendasi bekerja lebih baik ketika kebijakan berada dalam mode Forecast only selama

periode evaluasi. Jika Anda memulai kebijakan dalam mode Forecast dan skala dan beralih ke mode Forecast only nanti, temuan untuk kebijakan tersebut cenderung bias. Ini karena kebijakan telah berkontribusi terhadap kapasitas aktual.

Tinjau grafik pemantauan penskalaan prediktif

Di konsol, Anda dapat meninjau perkiraan hari, minggu, atau bulan sebelumnya untuk memvisualisasikan seberapa baik kinerja kebijakan dari waktu ke waktu. Anda juga dapat menggunakan informasi ini untuk mengevaluasi keakuratan prediksi saat memutuskan apakah akan membiarkan kebijakan menskalakan jumlah tugas Anda yang sebenarnya.

Untuk meninjau grafik pemantauan penskalaan prediktif di konsol Amazon ECS

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, pilih layanan.

   Halaman detail layanan muncul.
4. Pilih Service auto scaling.
5. Pilih kebijakan penskalaan prediktif, lalu pilih Tindakan, Penskalaan Prediktif, Lihat Grafik.
6. Di bagian Pemantauan, Anda dapat melihat perkiraan masa lalu dan masa depan kebijakan Anda untuk beban dan kapasitas terhadap nilai aktual. Grafik beban menunjukkan perkiraan beban dan nilai aktual untuk metrik beban yang Anda pilih. Grafik Kapasitas menunjukkan jumlah tugas yang diprediksi oleh kebijakan. Ini juga mencakup jumlah tugas aktual yang diluncurkan. Garis vertikal memisahkan nilai historis dari perkiraan masa depan. Grafik ini tersedia segera setelah kebijakan dibuat.
7. (Opsional) Untuk mengubah jumlah data historis yang ditampilkan dalam bagan, pilih nilai pilihan Anda dari dropdown Periode evaluasi di bagian atas halaman. Periode evaluasi tidak mengubah data di halaman ini dengan cara apa pun. Itu hanya mengubah jumlah data historis yang ditampilkan.

Bandingkan data dalam grafik Load

Setiap garis horizontal mewakili serangkaian titik data berbeda yang dilaporkan dalam interval satu jam:

1. Beban yang diamati aktual menggunakan statistik SUM untuk metrik beban yang Anda pilih untuk menunjukkan total beban per jam di masa lalu.
2. Beban yang diprediksi oleh kebijakan menunjukkan prediksi beban per jam. Prediksi ini didasarkan pada dua minggu sebelumnya dari pengamatan beban aktual.

Bandingkan data dalam grafik Kapasitas

Setiap garis horizontal mewakili serangkaian titik data berbeda yang dilaporkan dalam interval satu jam:

1. Jumlah tugas yang diamati secara aktual menunjukkan kapasitas aktual layanan Amazon ECS Anda di masa lalu, yang bergantung pada kebijakan penskalaan Anda yang lain dan ukuran grup minimum yang berlaku untuk periode waktu yang dipilih.
2. Kapasitas yang diprediksi oleh kebijakan menunjukkan kapasitas dasar yang dapat Anda harapkan pada awal setiap jam ketika kebijakan dalam mode Forecast dan skala.
3. Jumlah tugas yang diperlukan yang disimpulkan menunjukkan jumlah tugas ideal dalam layanan Anda untuk mempertahankan metrik penskalaan pada nilai target yang Anda pilih.
4. Jumlah tugas minimum menunjukkan jumlah tugas minimum dalam layanan Anda.
5. Kapasitas maksimum menunjukkan jumlah tugas maksimum dalam layanan Anda.

Untuk tujuan menghitung kapasitas yang diperlukan yang disimpulkan, kita mulai dengan mengasumsikan bahwa setiap tugas sama-sama digunakan pada nilai target yang ditentukan. Dalam praktiknya, jumlah tugas tidak digunakan secara merata. Dengan mengasumsikan bahwa pemanfaatan tersebar secara seragam antar tugas, bagaimanapun, kita dapat membuat perkiraan kemungkinan jumlah kapasitas yang dibutuhkan. Persyaratan untuk jumlah tugas kemudian dihitung berbanding terbalik dengan metrik penskalaan yang Anda gunakan untuk kebijakan penskalaan prediktif Anda. Dengan kata lain, ketika jumlah tugas meningkat, metrik penskalaan menurun pada tingkat yang sama. Misalnya, jika jumlah tugas berlipat ganda, metrik penskalaan harus berkurang setengahnya.

Rumus untuk kapasitas yang dibutuhkan yang disimpulkan:

```
sum of (actualServiceUnits*scalingMetricValue)/(targetUtilization)
```

Sebagai contoh, kita mengambil `actualServiceUnits` (`10`) dan `scalingMetricValue` (`30`) untuk satu jam tertentu. Kami kemudian mengambil `targetUtilization` yang Anda tentukan

dalam kebijakan penskalaan prediktif Anda (60) dan menghitung kapasitas yang diperlukan yang disimpulkan untuk jam yang sama. Ini mengembalikan nilai5. Ini berarti bahwa lima adalah jumlah kapasitas yang disimpulkan yang diperlukan untuk mempertahankan kapasitas dalam proporsi terbalik langsung dengan nilai target metrik penskalaan.

> **Note**
>
> Berbagai tuas tersedia bagi Anda untuk menyesuaikan dan meningkatkan penghematan biaya dan ketersediaan aplikasi Anda.
>
> - Anda menggunakan penskalaan prediktif untuk kapasitas dasar dan penskalaan dinamis untuk menangani kapasitas tambahan. Penskalaan dinamis bekerja secara independen dari penskalaan prediktif, penskalaan masuk dan keluar berdasarkan pemanfaatan saat ini. Pertama, Amazon ECS menghitung jumlah tugas yang disarankan untuk setiap kebijakan penskalaan yang tidak dijadwalkan. Kemudian, skala berdasarkan kebijakan yang menyediakan jumlah tugas terbesar.
> - Agar penskalaan masuk terjadi saat beban berkurang, layanan Anda harus selalu memiliki setidaknya satu kebijakan penskalaan dinamis dengan bagian penskalaan diaktifkan.
> - Anda dapat meningkatkan kinerja penskalaan dengan memastikan bahwa kapasitas minimum dan maksimum Anda tidak terlalu membatasi. Kebijakan dengan jumlah tugas yang disarankan yang tidak termasuk dalam rentang kapasitas minimum dan maksimum akan dicegah dari penskalaan masuk dan keluar.

Pantau metrik penskalaan prediktif untuk Amazon ECS dengan CloudWatch

Anda dapat menggunakan Amazon CloudWatch untuk memantau data Anda untuk penskalaan prediktif. Kebijakan penskalaan prediktif mengumpulkan data yang digunakan untuk memperkirakan beban masa depan Anda. Data yang dikumpulkan secara otomatis disimpan secara CloudWatch berkala dan dapat digunakan untuk memvisualisasikan seberapa baik kinerja kebijakan dari waktu ke waktu. Anda juga dapat membuat CloudWatch alarm untuk memberi tahu Anda ketika indikator kinerja berubah melampaui batas yang Anda tetapkan.

Visualisasikan data perkiraan historis

Data perkiraan beban untuk kebijakan penskalaan prediktif dapat dilihat CloudWatch dan dapat berguna saat memvisualisasikan prakiraan terhadap CloudWatch metrik lain dalam satu grafik. Anda juga dapat melihat tren dari waktu ke waktu dengan melihat rentang waktu yang lebih luas. Anda

dapat mengakses metrik historis hingga 15 bulan untuk mendapatkan perspektif yang lebih baik tentang kinerja kebijakan Anda.

Untuk melihat data perkiraan historis menggunakan CloudWatch konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Metrik, lalu Semua metrik.
3. Pilih namespace metrik Application Auto Scaling.
4. Pilih Prakiraan Beban Penskalaan Prediktif.
5. Di bidang pencarian, masukkan nama kebijakan penskalaan prediktif atau nama grup layanan Amazon ECS, lalu tekan Enter untuk memfilter hasilnya.
6. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk mengubah nama grafik, pilih ikon pensil. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom. Untuk informasi selengkapnya, lihat Membuat grafik metrik di Panduan CloudWatch Pengguna Amazon.
7. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individual, lalu pilih statistik yang berbeda. Meskipun Anda dapat memilih statistik apa pun untuk setiap metrik, tidak semua statistik berguna untuk PredictiveScalingLoadForecastmetrik. Misalnya, statistik Rata-rata, Minimum, dan Maksimum berguna, tetapi statistik Jumlah tidak.
8. Untuk menambahkan metrik lain ke grafik, di bawah Browse, pilih Semua, temukan metrik tertentu, lalu pilih kotak centang di sebelahnya. Anda dapat menambahkan hingga 10 metrik.
9. (Opsional) Untuk menambahkan grafik ke CloudWatch dasbor, pilih Tindakan, Tambahkan ke dasbor.

Buat metrik akurasi menggunakan matematika metrik

Dengan matematika metrik, Anda dapat menanyakan beberapa CloudWatch metrik dan menggunakan ekspresi matematika untuk membuat deret waktu baru berdasarkan metrik ini. Anda dapat memvisualisasikan deret waktu yang dihasilkan di CloudWatch konsol dan menambahkannya ke dasbor. Untuk informasi selengkapnya tentang matematika metrik, lihat Menggunakan matematika metrik di Panduan CloudWatch Pengguna Amazon.

Dengan menggunakan matematika metrik, Anda dapat membuat grafik data yang dihasilkan oleh penskalaan otomatis servis untuk penskalaan prediktif dengan berbagai cara. Ini membantu Anda memantau kinerja kebijakan dari waktu ke waktu, dan membantu Anda memahami apakah kombinasi metrik Anda dapat ditingkatkan.

Misalnya, Anda dapat menggunakan ekspresi matematika metrik untuk memantau kesalahan persentase absolut rata-rata (MAPE). Metrik MAPE membantu memantau perbedaan antara nilai yang diperkirakan dan nilai aktual yang diamati selama jendela perkiraan tertentu. Perubahan nilai MAPE dapat menunjukkan apakah kinerja kebijakan menurun seiring waktu karena sifat aplikasi Anda berubah. Peningkatan MAPE menandakan kesenjangan yang lebih luas antara nilai yang diperkirakan dan nilai aktual.

Contoh: Ekspresi matematika metrik

Untuk memulai dengan jenis grafik ini, Anda dapat membuat ekspresi matematika metrik seperti yang ditunjukkan pada contoh berikut.

Alih-alih metrik tunggal, ada array struktur kueri data metrik untuk`MetricDataQueries`. Setiap item `MetricDataQueries` mendapat metrik atau melakukan ekspresi matematika. Item pertama,`e1`, adalah ekspresi matematika. Ekspresi yang ditunjuk menetapkan `ReturnData` parameter ke`true`, yang pada akhirnya menghasilkan deret waktu tunggal. Untuk semua metrik lainnya, `ReturnData` nilainya adalah`false`.

Dalam contoh, ekspresi yang ditunjuk menggunakan nilai aktual dan yang diperkirakan sebagai input dan mengembalikan metrik baru (MAPE). `m1`adalah CloudWatch metrik yang berisi nilai beban aktual (dengan asumsi pemanfaatan CPU adalah metrik beban yang awalnya ditentukan untuk kebijakan bernama`my-predictive-scaling-policy`). `m2`adalah CloudWatch metrik yang berisi nilai beban yang diperkirakan. Sintaks matematika untuk metrik MAPE adalah sebagai berikut:

Rata-rata (abs ((Actual - Forecast)/(Aktual)))

Visualisasikan metrik akurasi Anda dan atur alarm

Untuk memvisualisasikan data metrik akurasi, pilih tab Metrik di konsol. CloudWatch Anda dapat membuat grafik data dari sana. Untuk informasi selengkapnya, lihat Menambahkan ekspresi matematika ke CloudWatch grafik di Panduan CloudWatch Pengguna Amazon.

Anda juga dapat mengatur alarm pada metrik yang Anda pantau dari bagian Metrik. Saat berada di tab Graphed metrics, pilih ikon Create alarm di bawah kolom Actions. Ikon Create alarm direpresentasikan sebagai bel kecil. Untuk informasi selengkapnya dan opsi notifikasi, lihat Membuat CloudWatch alarm berdasarkan ekspresi matematika metrik dan Memberi tahu pengguna tentang perubahan alarm di Panduan CloudWatch Pengguna Amazon.

Atau, Anda dapat menggunakan GetMetricDatadan PutMetricAlarmmelakukan perhitungan menggunakan matematika metrik dan membuat alarm berdasarkan output.

Menggunakan tindakan terjadwal untuk mengganti nilai perkiraan untuk Amazon ECS

Terkadang, Anda mungkin memiliki informasi tambahan tentang persyaratan aplikasi future Anda yang tidak dapat diperhitungkan oleh perhitungan perkiraan. Misalnya, perhitungan perkiraan mungkin meremehkan tugas yang diperlukan untuk acara pemasaran yang akan datang. Anda dapat menggunakan tindakan terjadwal untuk mengganti perkiraan sementara selama periode waktu mendatang. Tindakan terjadwal dapat berjalan secara berulang, atau pada tanggal dan waktu tertentu ketika ada fluktuasi permintaan satu kali.

Misalnya, Anda dapat membuat tindakan terjadwal dengan jumlah tugas yang lebih tinggi daripada yang diperkirakan. Saat runtime, Amazon ECS memperbarui jumlah tugas minimum dalam layanan Anda. Karena penskalaan prediktif mengoptimalkan jumlah tugas, tindakan terjadwal dengan jumlah tugas minimum yang lebih tinggi dari nilai perkiraan akan dihormati. Ini mencegah jumlah tugas menjadi kurang dari yang diharapkan. Untuk berhenti mengesampingkan perkiraan, gunakan tindakan terjadwal kedua untuk mengembalikan jumlah tugas minimum ke pengaturan aslinya.

Prosedur berikut menguraikan langkah-langkah untuk mengesampingkan perkiraan selama periode waktu mendatang.

Topik

- Langkah 1: (Opsional) Analisis data deret waktu
- Langkah 2: Buat dua tindakan terjadwal

> **Important**
>
> Topik ini mengasumsikan bahwa Anda mencoba mengesampingkan perkiraan untuk skala ke kapasitas yang lebih tinggi daripada yang diperkirakan. Jika Anda perlu mengurangi jumlah tugas untuk sementara waktu tanpa gangguan dari kebijakan penskalaan prediktif, gunakan mode hanya perkiraan. Sementara dalam mode perkiraan saja, penskalaan prediktif akan terus menghasilkan perkiraan, tetapi tidak akan secara otomatis meningkatkan jumlah tugas. Anda kemudian dapat memantau pemanfaatan sumber daya dan secara manual mengurangi jumlah tugas sesuai kebutuhan.

Langkah 1: (Opsional) Analisis data deret waktu

Mulailah dengan menganalisis data deret waktu perkiraan. Ini adalah langkah opsional, tetapi akan sangat membantu jika Anda ingin memahami detail perkiraan.

1. Ambil ramalan

   Setelah perkiraan dibuat, Anda dapat menanyakan periode waktu tertentu dalam perkiraan. Tujuan dari kueri ini adalah untuk mendapatkan tampilan lengkap dari data deret waktu untuk periode waktu tertentu.

   Kueri Anda dapat mencakup hingga dua hari data perkiraan masa depan. Jika Anda telah menggunakan penskalaan prediktif untuk sementara waktu, Anda juga dapat mengakses data perkiraan sebelumnya. Namun, durasi waktu maksimum antara waktu mulai dan akhir adalah 30 hari.

   Untuk mendapatkan perkiraan menggunakan get-predictive-scaling-forecast AWS CLI perintah, berikan parameter berikut dalam perintah:

   - Masukkan nama nama cluster dalam `resource-id` parameter.
   - Masukkan nama kebijakan dalam `--policy-name` parameter.
   - Masukkan waktu mulai dalam `--start-time` parameter untuk mengembalikan hanya data perkiraan setelah atau pada waktu yang ditentukan.
   - Masukkan waktu akhir dalam `--end-time` parameter untuk mengembalikan hanya data perkiraan sebelum waktu yang ditentukan.

```
aws application-autoscaling get-predictive-scaling-forecast \
    --service-namespace ecs \
    --resource-id service/MyCluster/test \
    --policy-name cpu40-predictive-scaling-policy \
    --scalable-dimension ecs:service:DesiredCount \
    --start-time "2021-05-19T17:00:00Z" \
    --end-time "2021-05-19T23:00:00Z"
```

   Jika berhasil, perintah mengembalikan data yang mirip dengan contoh berikut.

```
{
    "LoadForecast": [
        {
            "Timestamps": [
                "2021-05-19T17:00:00+00:00",
                "2021-05-19T18:00:00+00:00",
                "2021-05-19T19:00:00+00:00",
                "2021-05-19T20:00:00+00:00",
```

```
                "2021-05-19T21:00:00+00:00",
                "2021-05-19T22:00:00+00:00",
                "2021-05-19T23:00:00+00:00"
            ],
            "Values": [
                153.0655799339254,
                128.8288551285919,
                107.1179447150675,
                197.3601844551528,
                626.4039934516954,
                596.9441277518481,
                677.9675713779869
            ],
            "MetricSpecification": {
                "TargetValue": 40.0,
                "PredefinedMetricPairSpecification": {
                    "PredefinedMetricType": "ASGCPUUtilization"
                }
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

Tanggapan tersebut mencakup dua prakiraan: `LoadForecast` dan`CapacityForecast`. `LoadForecast`menunjukkan perkiraan beban per jam. `CapacityForecast`menunjukkan nilai perkiraan untuk kapasitas yang dibutuhkan setiap jam untuk menangani beban yang diperkirakan sambil mempertahankan 40,0 (`TargetValue`pemanfaatan CPU rata-rata 40%).

2. Identifikasi periode waktu target

   Identifikasi jam atau jam ketika fluktuasi permintaan satu kali harus terjadi. Ingat bahwa tanggal dan waktu yang ditunjukkan dalam perkiraan ada di UTC.

Langkah 2: Buat dua tindakan terjadwal

Selanjutnya, buat dua tindakan terjadwal untuk periode waktu tertentu ketika aplikasi Anda akan memiliki beban yang lebih tinggi dari perkiraan. Misalnya, jika Anda memiliki acara pemasaran yang akan mengarahkan lalu lintas ke situs Anda untuk jangka waktu terbatas, Anda dapat menjadwalkan tindakan satu kali untuk memperbarui kapasitas minimum saat dimulai. Kemudian, jadwalkan tindakan lain untuk mengembalikan kapasitas minimum ke pengaturan asli saat acara berakhir.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail cluster, di bagian Layanan, dan kemudian pilih layanan.

   Halaman detail layanan muncul.
4. Pilih Service Auto Scaling.

   Halaman kebijakan muncul.
5. Pilih Tindakan terjadwal, lalu pilih Buat.

   Halaman tindakan Create Schedule muncul.
6. Untuk nama Action, masukkan nama unik.
7. Untuk zona waktu, pilih zona waktu.

   Semua zona waktu yang tercantum berasal dari database Zona Waktu IANA. Untuk informasi selengkapnya, lihat Daftar zona waktu database tz.
8. Untuk Waktu mulai, masukkan Tanggal dan Waktu tindakan dimulai.
9. Untuk Perulangan, pilih Sekali.

10. Di bawah Penyesuaian tugas, Untuk Minimum, masukkan nilai kurang dari atau sama dengan jumlah tugas maksimum..
11. Pilih Buat tindakan terjadwal.

    Halaman kebijakan muncul.
12. Konfigurasikan tindakan terjadwal kedua untuk mengembalikan jumlah tugas minimum ke pengaturan asli di akhir acara. Penskalaan prediktif dapat menskalakan jumlah tugas hanya jika nilai yang Anda tetapkan untuk Minimum lebih rendah dari nilai perkiraan.

Untuk membuat dua tindakan terjadwal untuk acara satu kali ()AWS CLI

Untuk menggunakan AWS CLI untuk membuat tindakan terjadwal, gunakan perintah put-scheduled-update-group-action.

Sebagai contoh, mari kita tentukan jadwal yang mempertahankan kapasitas minimum tiga instance pada 19 Mei pukul 17:00 selama delapan jam. Perintah berikut menunjukkan bagaimana menerapkan skenario ini.

Perintah put-scheduled-update-group-action pertama menginstruksikan Amazon EC2 Auto Scaling untuk memperbarui kapasitas minimum grup Auto Scaling yang ditentukan pada pukul 17:00 UTC pada 19 Mei 2021.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-
start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-
capacity 3
```

Perintah kedua menginstruksikan Amazon EC2 Auto Scaling untuk mengatur kapasitas minimum grup menjadi satu pada 1:00 UTC pada 20 Mei 2021.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end
 \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-
capacity 1
```

Setelah Anda menambahkan tindakan terjadwal ini ke grup Auto Scaling, Amazon Auto EC2 Scaling melakukan hal berikut:

- Pada pukul 17:00 UTC pada 19 Mei 2021, aksi terjadwal pertama berjalan. Jika grup saat ini memiliki kurang dari tiga instance, grup tersebut menskalakan menjadi tiga instance. Selama waktu ini dan selama delapan jam ke depan, Amazon EC2 Auto Scaling dapat terus ditingkatkan jika kapasitas yang diprediksi lebih tinggi dari kapasitas aktual atau jika ada kebijakan penskalaan dinamis yang berlaku.
- Pukul 1:00 UTC pada 20 Mei 2021, aksi terjadwal kedua berjalan. Ini mengembalikan kapasitas minimum ke pengaturan aslinya di akhir acara.

Penskalaan berdasarkan jadwal berulang

Untuk mengganti perkiraan untuk periode waktu yang sama setiap minggu, buat dua tindakan terjadwal dan berikan logika waktu dan tanggal menggunakan ekspresi cron.

Format ekspresi cron terdiri dari lima bidang yang dipisahkan oleh spasi: [Minute] [Hour] [Day_of_month] [Month_of_year] [day_of_week]. Bidang dapat berisi nilai apa pun yang diizinkan, termasuk karakter khusus.

Misalnya, ekspresi cron berikut menjalankan aksi setiap hari Selasa pukul 6:30 pagi. Tanda bintang digunakan sebagai wildcard untuk mencocokkan semua nilai untuk bidang.

```
30 6 * * 2
```

Lihat juga

Untuk informasi selengkapnya tentang cara mengelola tindakan terjadwal, lihatGunakan tindakan terjadwal untuk menskalakan layanan Amazon ECS.

Kebijakan penskalaan prediktif lanjutan menggunakan metrik khusus untuk Amazon ECS

Anda dapat menggunakan metrik yang telah ditentukan atau kustom dalam kebijakan penskalaan prediktif. Metrik kustom berguna ketika metrik yang telah ditentukan sebelumnya, seperti CPU, memori, dll) tidak cukup untuk menggambarkan beban aplikasi Anda secara memadai.

Saat membuat kebijakan penskalaan prediktif dengan metrik khusus, Anda dapat menentukan metrik lain CloudWatch yang disediakan oleh. AWS Atau, Anda dapat menentukan metrik yang Anda tentukan dan publikasikan sendiri. Anda juga dapat menggunakan matematika metrik untuk menggabungkan dan mengubah metrik yang ada menjadi deret waktu baru yang AWS tidak dilacak secara otomatis. Contohnya adalah menggabungkan nilai dalam data Anda dengan menghitung jumlah atau rata-rata baru yang disebut agregasi. Data yang dihasilkan disebut agregat.

Bagian berikut berisi praktik terbaik dan contoh bagaimana membangun struktur JSON untuk kebijakan tersebut.

Prasyarat

Untuk menambahkan metrik kustom ke kebijakan penskalaan prediktif, Anda harus memiliki izin. `cloudwatch:GetMetricData`

Untuk menentukan metrik Anda sendiri, bukan metrik yang AWS disediakan, Anda harus terlebih dahulu mempublikasikan metrik Anda. CloudWatch Untuk informasi selengkapnya, lihat Menerbitkan metrik kustom di Panduan CloudWatch Pengguna Amazon.

Jika Anda mempublikasikan metrik Anda sendiri, pastikan untuk mempublikasikan titik data pada frekuensi minimum lima menit. Titik data diambil dari CloudWatch berdasarkan lamanya periode yang dibutuhkannya. Misalnya, spesifikasi metrik beban menggunakan metrik per jam untuk mengukur beban pada aplikasi Anda. CloudWatch menggunakan data metrik yang Anda publikasikan untuk memberikan nilai data tunggal untuk periode satu jam dengan menggabungkan semua titik data dengan stempel waktu yang termasuk dalam setiap periode satu jam.

Praktik terbaik

Praktik terbaik berikut dapat membantu Anda menggunakan metrik kustom secara lebih efektif:

- Metrik yang paling berguna untuk spesifikasi metrik beban adalah metrik yang mewakili beban pada grup Auto Scaling secara keseluruhan.
- Metrik yang paling berguna untuk spesifikasi metrik penskalaan untuk diskalakan adalah throughput rata-rata atau pemanfaatan per metrik tugas.
- Pemanfaatan target harus sesuai dengan jenis metrik penskalaan. Untuk konfigurasi kebijakan yang menggunakan pemanfaatan CPU, ini adalah persentase target, misalnya.
- Jika rekomendasi ini tidak diikuti, nilai future yang diperkirakan dari deret waktu mungkin akan salah. Untuk memvalidasi bahwa data sudah benar, Anda dapat melihat nilai yang diperkirakan di konsol. Atau, setelah Anda membuat kebijakan penskalaan prediktif, periksa `LoadForecast` objek yang ditampilkan oleh panggilan ke API. GetPredictiveScalingForecast
- Kami sangat menyarankan Anda mengonfigurasi penskalaan prediktif dalam mode hanya perkiraan sehingga Anda dapat mengevaluasi perkiraan sebelum penskalaan prediktif mulai aktif penskalaan.

Batasan

- Anda dapat menanyakan titik data hingga 10 metrik dalam satu spesifikasi metrik.
- Untuk tujuan batas ini, satu ekspresi dihitung sebagai satu metrik.

Memecahkan masalah kebijakan penskalaan prediktif dengan metrik khusus

Jika terjadi masalah saat menggunakan metrik kustom, sebaiknya Anda melakukan hal berikut:

- Jika Anda mengalami masalah dalam blue/green penerapan saat menggunakan ekspresi penelusuran, pastikan Anda membuat ekspresi penelusuran yang mencari kecocokan sebagian dan bukan kecocokan persis. Anda juga harus memeriksa apakah kueri hanya menemukan grup Auto Scaling yang berjalan di aplikasi tertentu. Untuk informasi selengkapnya tentang sintaks ekspresi penelusuran, lihat sintaks ekspresi CloudWatch penelusuran di CloudWatch Panduan Pengguna Amazon.
- put-scaling-policyPerintah memvalidasi ekspresi saat Anda membuat kebijakan penskalaan. Namun, ada kemungkinan bahwa perintah ini mungkin gagal mengidentifikasi penyebab pasti dari kesalahan yang terdeteksi. Untuk memperbaiki masalah, pecahkan masalah kesalahan yang Anda terima dalam respons dari permintaan ke perintah. get-metric-data Anda juga dapat memecahkan masalah ekspresi dari konsol. CloudWatch
- Anda harus menentukan `false` `ReturnData` jika `MetricDataQueries` menentukan fungsi SEARCH () sendiri tanpa fungsi matematika seperti SUM (). Ini karena ekspresi pencarian mungkin mengembalikan beberapa deret waktu, dan spesifikasi metrik berdasarkan ekspresi hanya dapat mengembalikan satu deret waktu.
- Semua metrik yang terlibat dalam ekspresi pencarian harus memiliki resolusi yang sama.

Membangun JSON untuk metrik kustom penskalaan prediktif dengan Amazon ECS

Bagian berikut berisi contoh cara mengonfigurasi penskalaan prediktif ke data kueri dari. CloudWatch Ada dua metode berbeda untuk mengonfigurasi opsi ini, dan metode yang Anda pilih memengaruhi format mana yang Anda gunakan untuk membuat JSON untuk kebijakan penskalaan prediktif Anda. Saat Anda menggunakan matematika metrik, format JSON bervariasi lebih lanjut berdasarkan matematika metrik yang dilakukan.

1. Untuk membuat kebijakan yang mendapatkan data langsung dari CloudWatch metrik lain yang disediakan oleh AWS atau metrik yang Anda publikasikan CloudWatch, lihat. Contoh kebijakan penskalaan prediktif dengan metrik pemuatan dan penskalaan khusus menggunakan AWS CLI

Contoh kebijakan penskalaan prediktif dengan metrik pemuatan dan penskalaan khusus menggunakan AWS CLI

Untuk membuat kebijakan penskalaan prediktif dengan metrik pemuatan dan penskalaan khusus dengan AWS CLI, simpan argumen `--predictive-scaling-configuration` dalam file JSON bernama. `config.json`

Anda mulai menambahkan metrik khusus dengan mengganti nilai yang dapat diganti dalam contoh berikut dengan metrik dan pemanfaatan target Anda.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
```

```
                {
                  "Name": "MyOptionalMetricDimensionName",
                  "Value": "MyOptionalMetricDimensionValue"
                }
              ]
            },
            "Stat": "Sum"
          }
        }
      ]
    }
  }
]
}
```

Untuk informasi selengkapnya, lihat MetricDataQuerydi Referensi API EC2 Auto Scaling Amazon.

> **Note**
>
> Berikut adalah beberapa sumber daya tambahan yang dapat membantu Anda menemukan nama metrik, ruang nama, dimensi, dan statistik untuk CloudWatch metrik:
>
> - Untuk informasi tentang metrik yang tersedia untuk AWS layanan, lihat AWS layanan yang memublikasikan CloudWatch metrik di CloudWatch Panduan Pengguna Amazon.
> - Untuk mendapatkan nama metrik, namespace, dan dimensi yang tepat (jika ada) untuk CloudWatch metrik dengan metrik AWS CLI, lihat daftar-metrik.

Untuk membuat kebijakan ini, jalankan put-scaling-policyperintah menggunakan file JSON sebagai input, seperti yang ditunjukkan dalam contoh berikut.

```
aws application-autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

Jika berhasil, perintah ini mengembalikan Amazon Resource Name (ARN) kebijakan.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
```

```
  "Alarms": []
}
```

Gunakan ekspresi matematika metrik

Bagian berikut memberikan informasi tentang penggunaan matematika metrik dengan kebijakan penskalaan prediktif dalam kebijakan Anda.

Memahami matematika metrik

Jika yang ingin Anda lakukan hanyalah mengumpulkan data metrik yang ada, matematika CloudWatch metrik menghemat upaya dan biaya penerbitan metrik lain. CloudWatch Anda dapat menggunakan metrik apa pun yang AWS menyediakan, dan Anda juga dapat menggunakan metrik yang Anda tetapkan sebagai bagian dari aplikasi Anda.

Untuk informasi selengkapnya, lihat Menggunakan matematika metrik di Panduan CloudWatch Pengguna Amazon.

Jika Anda memilih untuk menggunakan ekspresi matematika metrik dalam kebijakan penskalaan prediktif Anda, pertimbangkan poin-poin berikut:

- Operasi matematika metrik menggunakan titik data dari kombinasi unik nama metrik, namespace, dan keys/value pasangan dimensi metrik.
- Anda dapat menggunakan operator aritmatika (+ - */^), fungsi statistik (seperti AVG atau SUM), atau fungsi lain yang mendukung. CloudWatch
- Anda dapat menggunakan metrik dan hasil ekspresi matematika lainnya dalam rumus ekspresi matematika.
- Ekspresi matematika metrik Anda dapat terdiri dari agregasi yang berbeda. Namun, ini adalah praktik terbaik untuk hasil agregasi akhir yang digunakan `Average` untuk metrik penskalaan dan `Sum` untuk metrik beban.
- Setiap ekspresi yang digunakan dalam spesifikasi metrik pada akhirnya harus mengembalikan satu deret waktu.

Untuk menggunakan matematika metrik, lakukan hal berikut:

- Pilih satu atau beberapa CloudWatch metrik. Kemudian, buat ekspresi. Untuk informasi selengkapnya, lihat Menggunakan matematika metrik di Panduan CloudWatch Pengguna Amazon.

- Verifikasi bahwa ekspresi matematika metrik valid dengan menggunakan CloudWatch konsol atau CloudWatch GetMetricDataAPI.

Contoh kebijakan penskalaan prediktif yang menggabungkan metrik menggunakan metrik matematika ()AWS CLI

Terkadang, alih-alih menentukan metrik secara langsung, Anda mungkin perlu terlebih dahulu memproses datanya dengan cara tertentu. Misalnya, Anda mungkin memiliki aplikasi yang menarik pekerjaan dari antrean Amazon SQS, dan Anda mungkin ingin menggunakan jumlah item dalam antrian sebagai kriteria untuk penskalaan prediktif. Jumlah pesan dalam antrian tidak hanya menentukan jumlah instance yang Anda butuhkan. Oleh karena itu, lebih banyak pekerjaan diperlukan untuk membuat metrik yang dapat digunakan untuk menghitung backlog per instance.

Berikut ini adalah contoh kebijakan penskalaan prediktif untuk skenario ini. Ini menentukan metrik penskalaan dan pemuatan yang didasarkan pada metrik Amazon SQS`ApproximateNumberOfMessagesVisible`, yang merupakan jumlah pesan yang tersedia untuk diambil dari antrian. Ini juga menggunakan `GroupInServiceInstances` metrik EC2 Auto Scaling Amazon dan ekspresi matematika untuk menghitung backlog per instance untuk metrik penskalaan.

```
aws application-autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-
policy \
  --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
  --service-namespace ecs \
  --resource-id service/MyCluster/test \
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
```

```
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
                    "Value": "my-asg"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
```

```
                {
                  "Name": "QueueName",
                  "Value": "my-queue"
                }
              ],
            },
            "Stat": "Sum"
          },
          "ReturnData": true
        }
      ]
    }
  }
]
}
```

Contoh mengembalikan ARN kebijakan.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

Interkoneksi layanan Amazon ECS

Aplikasi yang berjalan dalam tugas Amazon ECS seringkali perlu menerima koneksi dari internet atau untuk terhubung ke aplikasi lain yang berjalan di layanan Amazon ECS. Jika Anda membutuhkan koneksi eksternal dari internet, sebaiknya gunakan Elastic Load Balancing. Untuk informasi selengkapnya tentang penyeimbangan beban terintegrasi, lihatthe section called “Gunakan load balancing untuk mendistribusikan lalu lintas layanan”.

Jika Anda memerlukan aplikasi untuk terhubung ke aplikasi lain yang berjalan di layanan Amazon ECS, Amazon ECS menyediakan cara-cara berikut untuk melakukan ini tanpa penyeimbang beban:

- Amazon ECS Service Connect

  Kami merekomendasikan Service Connect, yang menyediakan konfigurasi Amazon ECS untuk penemuan layanan, konektivitas, dan pemantauan lalu lintas. Dengan Service Connect, aplikasi Anda dapat menggunakan nama pendek dan port standar untuk terhubung ke layanan Amazon ECS di cluster yang sama, kluster lain, termasuk di VPCs dalam yang sama. Wilayah AWS

Saat Anda menggunakan Service Connect, Amazon ECS mengelola semua bagian penemuan layanan: membuat nama yang dapat ditemukan, mengelola entri secara dinamis untuk setiap tugas saat tugas dimulai dan dihentikan, menjalankan agen di setiap tugas yang dikonfigurasi untuk menemukan nama. Aplikasi Anda dapat mencari nama dengan menggunakan fungsionalitas standar untuk nama DNS dan membuat koneksi. Jika aplikasi Anda sudah melakukan ini, Anda tidak perlu memodifikasi aplikasi Anda untuk menggunakan Service Connect.

Anda menyediakan konfigurasi lengkap di dalam setiap layanan dan definisi tugas. Amazon ECS mengelola perubahan pada konfigurasi ini di setiap penyebaran layanan, untuk memastikan bahwa semua tugas dalam penerapan berperilaku dengan cara yang sama. Misalnya, masalah umum dengan DNS sebagai penemuan layanan adalah mengendalikan migrasi. Jika Anda mengubah nama DNS untuk menunjuk ke alamat IP pengganti baru, mungkin diperlukan waktu TTL maksimum sebelum semua klien mulai menggunakan layanan baru. Dengan Service Connect, penyebaran klien memperbarui konfigurasi dengan mengganti tugas klien. Anda dapat mengonfigurasi pemutus sirkuit penyebaran dan konfigurasi penerapan lainnya untuk memengaruhi perubahan Service Connect dengan cara yang sama seperti penerapan lainnya.

Untuk informasi selengkapnya, lihat Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek.

- Penemuan layanan Amazon ECS

  Pendekatan lain untuk service-to-service komunikasi adalah komunikasi langsung menggunakan penemuan layanan. Dalam pendekatan ini, Anda dapat menggunakan integrasi penemuan AWS Cloud Map layanan dengan Amazon ECS. Menggunakan penemuan layanan, Amazon ECS menyinkronkan daftar tugas yang diluncurkan ke AWS Cloud Map, yang mempertahankan nama host DNS yang menyelesaikan ke alamat IP internal dari satu atau lebih tugas dari layanan tertentu. Layanan lain di Amazon VPC dapat menggunakan nama host DNS ini untuk mengirim lalu lintas langsung ke wadah lain menggunakan alamat IP internalnya.

  Pendekatan service-to-service komunikasi ini memberikan latensi rendah. Tidak ada komponen tambahan di antara wadah. Lalu lintas bergerak langsung dari satu kontainer ke kontainer lainnya.

  Pendekatan ini cocok saat menggunakan mode `awsvpc` jaringan, di mana setiap tugas memiliki alamat IP uniknya sendiri. Sebagian besar perangkat lunak hanya mendukung penggunaan `A` catatan DNS, yang menyelesaikan langsung ke alamat IP. Saat menggunakan mode `awsvpc` jaringan, alamat IP untuk setiap tugas adalah `A` catatan. Namun, jika Anda menggunakan mode `bridge` jaringan, beberapa kontainer dapat berbagi alamat IP yang sama. Selain itu, pemetaan

port dinamis menyebabkan kontainer diberi nomor port secara acak pada alamat IP tunggal itu. Pada titik ini, `A` catatan tidak lagi cukup untuk penemuan layanan. Anda juga harus menggunakan `SRV` catatan. Jenis catatan ini dapat melacak alamat IP dan nomor port tetapi mengharuskan Anda mengonfigurasi aplikasi dengan tepat. Beberapa aplikasi bawaan yang Anda gunakan mungkin tidak mendukung `SRV` catatan.

Keuntungan lain dari mode `awsvpc` jaringan adalah Anda memiliki grup keamanan unik untuk setiap layanan. Anda dapat mengonfigurasi grup keamanan ini untuk mengizinkan koneksi masuk hanya dari layanan hulu tertentu yang perlu berbicara dengan layanan tersebut.

Kerugian utama dari service-to-service komunikasi langsung menggunakan penemuan layanan adalah Anda harus menerapkan logika tambahan untuk mencoba ulang dan menangani kegagalan koneksi. Catatan DNS memiliki periode time-to-live (TTL) yang mengontrol berapa lama mereka di-cache. Dibutuhkan beberapa waktu agar catatan DNS diperbarui dan cache kedaluwarsa sehingga aplikasi Anda dapat mengambil versi terbaru dari catatan DNS. Jadi, aplikasi Anda mungkin akan menyelesaikan catatan DNS untuk menunjuk ke wadah lain yang sudah tidak ada lagi. Aplikasi Anda perlu menangani percobaan ulang dan memiliki logika untuk mengabaikan backend yang buruk.

Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS

- Kisi VPC Amazon

  Amazon VPC Lattice adalah layanan jaringan aplikasi terkelola yang digunakan pelanggan Amazon ECS untuk mengamati, mengamankan, dan memantau aplikasi yang dibangun di seluruh layanan AWS komputasi VPCs, dan akun tanpa harus mengubah kode mereka.

  VPC Lattice menggunakan kelompok target, yang merupakan kumpulan sumber daya komputasi. Target ini menjalankan aplikasi atau layanan Anda dan dapat berupa EC2 instans Amazon, alamat IP, fungsi Lambda, dan Application Load Balancer. Dengan mengaitkan layanan Amazon ECS mereka dengan grup target VPC Lattice, pelanggan sekarang dapat mengaktifkan tugas Amazon ECS sebagai target IP di VPC Lattice. Amazon ECS secara otomatis mendaftarkan tugas ke grup target VPC Lattice saat tugas untuk layanan terdaftar diluncurkan.

  Untuk informasi selengkapnya, lihat Gunakan Amazon VPC Lattice untuk menghubungkan, mengamati, dan mengamankan layanan Amazon ECS Anda.

Tabel kompatibilitas mode jaringan

Tabel berikut mencakup kompatibilitas antara opsi ini dan mode jaringan tugas. Dalam tabel, “klien” mengacu pada aplikasi yang membuat koneksi dari dalam tugas Amazon ECS.

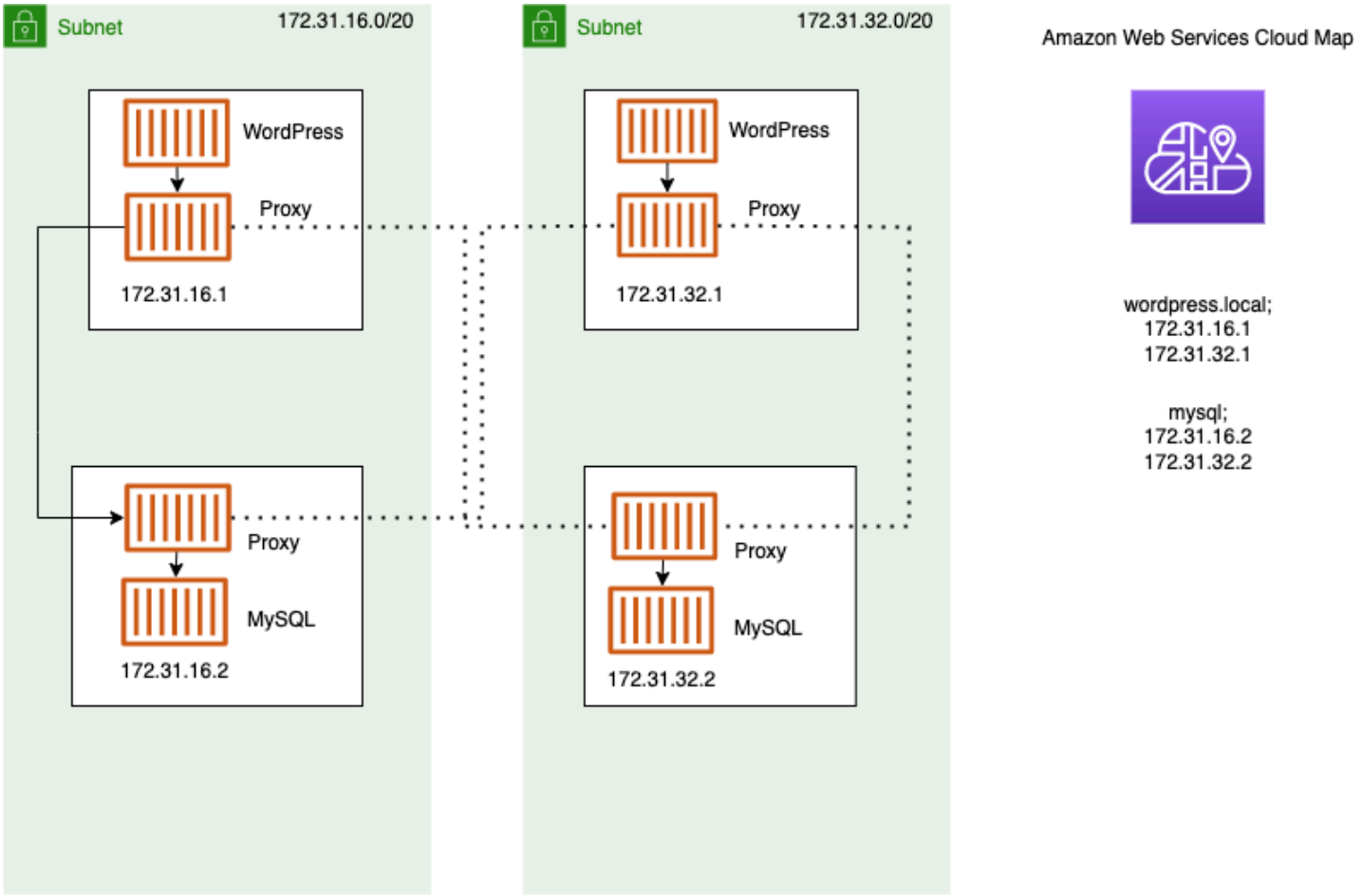
Opsi Interkoneksi	Dijembatani	**`awsvpc`**	Host
Penemuan layanan	ya, tetapi mengharuskan klien mengetahui catatan SRV di DNS tanpa. `hostPort`	Ya	ya, tetapi mengharuskan klien mengetahui catatan SRV di DNS tanpa. `hostPort`
Layanan Connect	Ya	ya	tidak
Kisi VPC	Ya	Ya	Ya

Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek

Amazon ECS Service Connect menyediakan manajemen service-to-service komunikasi sebagai konfigurasi Amazon ECS. Ini membangun penemuan layanan dan mesh layanan di Amazon ECS. Ini menyediakan konfigurasi lengkap di dalam setiap layanan yang Anda kelola berdasarkan penerapan layanan, cara terpadu untuk merujuk ke layanan Anda dalam ruang nama yang tidak bergantung pada konfigurasi DNS VPC, serta metrik dan log standar untuk memantau semua aplikasi Anda. Service Connect hanya menghubungkan layanan.

Diagram berikut menunjukkan contoh jaringan Service Connect dengan 2 subnet di VPC dan 2 layanan. Layanan klien yang berjalan WordPress dengan 1 tugas di setiap subnet. Layanan server yang menjalankan MySQL dengan 1 tugas di setiap subnet. Kedua layanan sangat tersedia dan tahan terhadap masalah tugas dan Availability Zone karena setiap layanan menjalankan beberapa tugas yang tersebar di 2 subnet. Panah padat menunjukkan koneksi dari WordPress ke MySQL. Misalnya, perintah `mysql --host=mysql` CLI yang dijalankan dari dalam WordPress wadah dalam tugas dengan alamat IP. `172.31.16.1` Perintah menggunakan nama pendek `mysql` pada port default untuk MySQL. Nama dan port ini terhubung ke proxy Service Connect dalam tugas yang sama. Proxy dalam WordPress tugas menggunakan penyeimbangan beban round-robin dan

informasi kegagalan sebelumnya dalam deteksi outlier untuk memilih tugas MySQL mana yang akan dihubungkan. Seperti yang ditunjukkan oleh panah padat dalam diagram, proxy terhubung ke proxy kedua dalam tugas MySQL dengan Alamat IP. `172.31.16.2` Proxy kedua terhubung ke server MySQL lokal dalam tugas yang sama. Kedua proxy melaporkan kinerja koneksi yang terlihat dalam grafik di Amazon ECS dan konsol CloudWatch Amazon sehingga Anda bisa mendapatkan metrik kinerja dari semua jenis aplikasi dengan cara yang sama.



Istilah berikut digunakan dengan Service Connect.

nama port

Konfigurasi definisi tugas Amazon ECS yang memberikan nama ke pemetaan port tertentu. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

alias klien

Konfigurasi layanan Amazon ECS yang menetapkan nomor port yang digunakan di titik akhir. Selain itu, alias klien dapat menetapkan nama DNS dari titik akhir, mengesampingkan nama penemuan. Jika nama penemuan tidak disediakan di layanan Amazon ECS, nama alias klien

akan mengganti nama port sebagai nama titik akhir. Untuk contoh endpoint, lihat definisi endpoint. Beberapa alias klien dapat ditetapkan ke layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

nama penemuan

Nama perantara opsional yang dapat Anda buat untuk port tertentu dari definisi tugas. Nama ini digunakan untuk membuat AWS Cloud Map layanan. Jika nama ini tidak disediakan, nama port dari definisi tugas akan digunakan. Beberapa nama penemuan dapat ditetapkan ke port tertentu layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

AWS Cloud Map nama layanan harus unik dalam namespace. Karena batasan ini, Anda hanya dapat memiliki satu konfigurasi Service Connect tanpa nama penemuan untuk definisi tugas tertentu di setiap namespace.

titik akhir

URL untuk terhubung ke API atau situs web. URL berisi protokol, nama DNS, dan port. Untuk informasi lebih lanjut tentang titik akhir secara umum, lihat titik akhir dalam AWS glosarium di. Referensi Umum Amazon Web

Service Connect membuat titik akhir yang terhubung ke layanan Amazon ECS dan mengonfigurasi tugas di layanan Amazon ECS untuk terhubung ke titik akhir. URL berisi protokol, nama DNS, dan port. Anda memilih protokol dan nama port dalam definisi tugas, karena port harus cocok dengan aplikasi yang ada di dalam gambar kontainer. Dalam layanan, Anda memilih setiap port dengan nama dan dapat menetapkan nama DNS. Jika Anda tidak menentukan nama DNS dalam konfigurasi layanan Amazon ECS, nama port dari definisi tugas akan digunakan secara default. Misalnya, titik akhir Service Connect dapat berupa`http://blog:80`,`grpc://checkout:8080`, atau`http://_db.production.internal:99`.

Layanan Connect Service

Konfigurasi titik akhir tunggal dalam layanan Amazon ECS. Ini adalah bagian dari konfigurasi Service Connect, yang terdiri dari satu baris dalam konfigurasi Service Connect dan discovery name di konsol, atau satu objek dalam `services` daftar dalam konfigurasi JSON dari layanan Amazon ECS. Konfigurasi ini hanya digunakan oleh Amazon ECS Service Connect.

Untuk informasi selengkapnya, lihat ServiceConnectServicedi Referensi API Amazon Elastic Container Service.

namespace

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari AWS Cloud Map namespace untuk digunakan dengan Service Connect. Namespace harus Wilayah AWS sama dengan layanan Amazon ECS dan cluster. Jenis namespace di AWS Cloud Map tidak memengaruhi Service Connect.

Service Connect menggunakan AWS Cloud Map namespace sebagai pengelompokan logis tugas Amazon ECS yang berbicara satu sama lain. Setiap layanan Amazon ECS hanya dapat dimiliki oleh satu namespace. Layanan dalam namespace dapat tersebar di berbagai kluster Amazon ECS yang berbeda dalam hal yang sama Wilayah AWS . Akun AWS Anda dapat dengan bebas mengatur layanan dengan kriteria apa pun.

layanan klien

Layanan yang menjalankan aplikasi klien jaringan. Layanan ini harus memiliki namespace yang dikonfigurasi. Setiap tugas dalam layanan dapat menemukan dan terhubung ke semua titik akhir di namespace melalui container proxy Service Connect.

Jika salah satu kontainer Anda dalam tugas perlu terhubung ke titik akhir dari layanan di namespace, pilih layanan klien. Jika aplikasi frontend, reverse proxy, atau load balancer menerima lalu lintas eksternal melalui metode lain seperti dari Elastic Load Balancing, itu bisa menggunakan konfigurasi Service Connect jenis ini.

layanan client-server

Layanan Amazon ECS yang menjalankan aplikasi jaringan atau layanan web. Layanan ini harus memiliki namespace dan setidaknya satu titik akhir yang dikonfigurasi. Setiap tugas dalam layanan dapat dijangkau dengan menggunakan titik akhir. Container proxy Service Connect mendengarkan nama endpoint dan port untuk mengarahkan lalu lintas ke container aplikasi dalam tugas.

Jika salah satu kontainer mengekspos dan mendengarkan pada port untuk lalu lintas jaringan, pilih layanan client-server. Aplikasi ini tidak perlu terhubung ke layanan client-server lain di namespace yang sama, tetapi konfigurasi klien diperlukan. Backend, middleware, tingkat bisnis, atau sebagian besar layanan mikro dapat menggunakan konfigurasi Service Connect jenis ini. Jika Anda ingin aplikasi frontend, reverse proxy, atau load balancer menerima lalu lintas dari layanan lain yang dikonfigurasi dengan Service Connect di namespace yang sama, layanan ini harus menggunakan jenis konfigurasi Service Connect ini.

Fitur Service Connect menciptakan jaringan virtual layanan terkait. Konfigurasi layanan yang sama dapat digunakan di beberapa ruang nama yang berbeda untuk menjalankan set aplikasi yang independen namun identik. Service Connect mendefinisikan wadah proxy di layanan Amazon ECS. Dengan cara ini, definisi tugas yang sama dapat digunakan untuk menjalankan aplikasi identik di ruang nama yang berbeda dengan konfigurasi Service Connect yang berbeda. Setiap tugas yang dibuat oleh layanan menjalankan wadah proxy dalam tugas.

Service Connect cocok untuk koneksi antara layanan Amazon ECS dalam namespace yang sama. Untuk aplikasi berikut, Anda perlu menggunakan metode interkoneksi tambahan untuk menyambung ke layanan Amazon ECS yang dikonfigurasi dengan Service Connect:

- Tugas yang dikonfigurasi di ruang nama lain
- Tugas yang tidak dikonfigurasi untuk Service Connect
- Aplikasi lain di luar Amazon ECS

Aplikasi ini dapat terhubung melalui proxy Service Connect tetapi tidak dapat menyelesaikan nama titik akhir Service Connect.

Agar aplikasi ini dapat menyelesaikan alamat IP tugas Amazon ECS, Anda perlu menggunakan metode interkoneksi lain.

Topik

- Harga
- Komponen Amazon ECS Service Connect
- Ikhtisar konfigurasi Amazon ECS Service Connect
- Enkripsi lalu lintas Amazon ECS Service Connect
- Mengkonfigurasi Amazon ECS Service Connect dengan AWS CLI

Harga

- Harga Amazon ECS Service Connect bergantung pada apakah Anda menggunakan AWS Fargate atau EC2 infrastruktur Amazon untuk meng-host beban kerja kontainer Anda. Saat menggunakan Amazon ECS aktif AWS Outposts, harga mengikuti model yang sama yang digunakan saat Anda menggunakan Amazon EC2 secara langsung. Untuk informasi selengkapnya, lihat Harga Amazon ECS.
- Tidak ada biaya tambahan untuk menggunakan Amazon ECS Service Connect.

- AWS Cloud Map penggunaan sepenuhnya gratis saat digunakan oleh Service Connect.
- Pelanggan membayar sumber daya komputasi, yang digunakan oleh Amazon ECS Service Connect, termasuk vCPU dan Memori. Karena agen Amazon ECS Service Connect berjalan di dalam tugas pelanggan, tidak ada biaya tambahan untuk menjalankannya. Sumber daya tugas dibagi antara beban kerja pelanggan dan agen Amazon ECS Service Connect.
- Saat menggunakan fungsionalitas enkripsi lalu lintas Amazon ECS Service Connect AWS Private CA, pelanggan membayar otoritas sertifikat pribadi yang mereka buat dan untuk setiap sertifikat TLS yang dikeluarkan. Untuk detail selengkapnya, lihat AWS Private Certificate Authority harga. Untuk memperkirakan biaya bulanan sertifikat TLS, pelanggan perlu mengetahui jumlah layanan Amazon ECS yang mengaktifkan TLS, kalikan dengan biaya sertifikat, dan kemudian kalikan dengan enam. Karena Amazon ECS Service Connect secara otomatis memutar sertifikat TLS setiap lima hari, ada enam sertifikat yang dikeluarkan per layanan Amazon ECS, rata-rata per bulan.

Komponen Amazon ECS Service Connect

Saat Anda menggunakan Amazon ECS Service Connect, Anda mengonfigurasi setiap layanan Amazon ECS untuk menjalankan aplikasi server yang menerima permintaan jaringan (layanan client-server) atau menjalankan aplikasi klien yang membuat permintaan (layanan klien).

Saat Anda bersiap untuk mulai menggunakan Service Connect, mulailah dengan layanan client-server. Anda dapat menambahkan konfigurasi Service Connect ke layanan baru atau layanan yang sudah ada. Amazon ECS membuat titik akhir Service Connect di namespace. Selain itu, Amazon ECS membuat penerapan baru di layanan untuk menggantikan tugas yang sedang berjalan.

Tugas yang ada dan aplikasi lain dapat terus terhubung ke titik akhir yang ada, dan aplikasi eksternal. Jika layanan client-server menambahkan tugas dengan skala keluar, koneksi baru dari klien akan seimbang di antara semua tugas. Jika layanan client-server diperbarui, koneksi baru dari klien akan seimbang antara tugas-tugas versi baru.

Tugas yang ada tidak dapat menyelesaikan dan terhubung ke titik akhir yang baru. Hanya tugas baru dengan konfigurasi Service Connect di namespace yang sama dan yang mulai berjalan setelah penerapan ini dapat menyelesaikan dan terhubung ke titik akhir ini.

Ini berarti bahwa operator aplikasi klien menentukan kapan konfigurasi aplikasi mereka berubah, meskipun operator aplikasi server dapat mengubah konfigurasi mereka kapan saja. Daftar titik akhir di namespace dapat berubah setiap kali layanan apa pun di namespace diterapkan. Tugas yang ada dan tugas pengganti terus berperilaku sama seperti yang mereka lakukan setelah penerapan terbaru.

Pertimbangkan contoh berikut.

Pertama, asumsikan bahwa Anda membuat aplikasi yang tersedia untuk internet publik dalam satu AWS CloudFormation template dan AWS CloudFormation tumpukan tunggal. Penemuan dan jangkauan publik harus dibuat terakhir oleh AWS CloudFormation, termasuk layanan klien frontend. Layanan perlu dibuat dalam urutan ini untuk mencegah periode waktu ketika layanan klien frontend berjalan dan tersedia untuk umum, tetapi backend tidak. Ini menghilangkan pesan kesalahan agar tidak dikirim ke publik selama periode waktu tersebut. Di AWS CloudFormation, Anda harus menggunakan `dependsOn` untuk menunjukkan AWS CloudFormation bahwa beberapa layanan Amazon ECS tidak dapat dibuat secara paralel atau bersamaan. Anda harus menambahkan `dependsOn` ke layanan klien frontend untuk setiap layanan client-server backend yang terhubung dengan tugas klien.

Kedua, asumsikan bahwa layanan frontend ada tanpa konfigurasi Service Connect. Tugas menghubungkan ke layanan backend yang ada. Tambahkan konfigurasi Service Connect client-server ke layanan backend terlebih dahulu, menggunakan nama yang sama di DNS atau `clientAlias` yang digunakan frontend. Ini menciptakan penerapan baru, sehingga semua deteksi rollback penerapan atau AWS Management Console, AWS CLI, AWS SDKs dan metode lain untuk memutar kembali dan mengembalikan layanan backend ke penerapan dan konfigurasi sebelumnya. Jika Anda puas dengan kinerja dan perilaku layanan backend, tambahkan konfigurasi Service Connect klien atau client-server ke layanan frontend. Hanya tugas dalam penerapan baru yang menggunakan proxy Service Connect yang ditambahkan ke tugas baru tersebut. Jika Anda memiliki masalah dengan konfigurasi ini, Anda dapat memutar kembali dan kembali ke konfigurasi sebelumnya dengan menggunakan deteksi rollback penerapan atau AWS Management Console, AWS CLI, AWS SDKs dan metode lain untuk memutar kembali dan mengembalikan layanan backend ke penerapan dan konfigurasi sebelumnya. Jika Anda menggunakan sistem penemuan layanan lain yang didasarkan pada DNS dan bukan Service Connect, aplikasi frontend atau klien apa pun mulai menggunakan titik akhir baru dan mengubah konfigurasi titik akhir setelah cache DNS lokal kedaluwarsa, biasanya memakan waktu beberapa jam.

Jaringan

Secara default, proxy Service Connect mendengarkan `containerPort` dari pemetaan port definisi tugas. Aturan grup keamanan Anda harus mengizinkan lalu lintas masuk (masuknya) ke port ini dari subnet tempat klien akan dijalankan.

Bahkan jika Anda menetapkan nomor port dalam konfigurasi layanan Service Connect, ini tidak mengubah port untuk layanan client-server yang didengarkan proxy Service Connect. Saat Anda menyetel nomor port ini, Amazon ECS mengubah port titik akhir yang terhubung dengan layanan

klien, pada proxy Service Connect di dalam tugas tersebut. Proxy dalam layanan klien terhubung ke proxy di layanan client-server menggunakan file. `containerPort`

Jika Anda ingin mengubah port yang didengarkan proxy Service Connect, ubah konfigurasi Service Connect pada layanan client-server. `ingressPort0verride` Jika Anda mengubah nomor port ini, Anda harus mengizinkan lalu lintas masuk pada port ini yang digunakan oleh lalu lintas ke layanan ini.

Lalu lintas yang dikirim aplikasi Anda ke layanan Amazon ECS yang dikonfigurasi untuk Service Connect mengharuskan Amazon VPC dan subnet memiliki aturan tabel rute dan aturan ACL jaringan yang memungkinkan `containerPort` `ingressPort0verride` dan nomor port yang Anda gunakan.

Anda dapat menggunakan Service Connect untuk mengirim lalu lintas di antaranya VPCs. Persyaratan yang sama untuk aturan tabel rute, jaringan ACLs, dan grup keamanan berlaku untuk keduanya VPCs.

Misalnya, dua cluster membuat tugas yang berbeda VPCs. Layanan di setiap cluster dikonfigurasi untuk menggunakan namespace yang sama. Aplikasi dalam dua layanan ini dapat menyelesaikan setiap titik akhir di namespace tanpa konfigurasi DNS VPC apa pun. Namun, proxy tidak dapat terhubung kecuali VPC peering, VPC, atau tabel rute subnet, dan jaringan VPC memungkinkan lalu lintas pada dan nomor port. ACLs `containerPort` `ingressPort0verride`

Untuk tugas yang menggunakan mode `bridge` jaringan, Anda harus membuat grup keamanan dengan aturan masuk yang memungkinkan lalu lintas pada rentang port dinamis atas. Kemudian, tetapkan grup keamanan ke semua EC2 instance di kluster Service Connect.

Proksi Service Connect

Jika Anda membuat atau memperbarui layanan dengan konfigurasi Service Connect, Amazon ECS menambahkan wadah baru ke setiap tugas baru saat dimulai. Pola penggunaan wadah terpisah ini disebut `asidecar`. Wadah ini tidak ada dalam definisi tugas dan Anda tidak dapat mengonfigurasinya. Amazon ECS mengelola konfigurasi Container dalam layanan. Ini memungkinkan Anda untuk menggunakan kembali definisi tugas yang sama antara beberapa layanan, ruang nama, dan tugas tanpa Service Connect.

Sumber daya proxy

- Untuk definisi tugas, Anda harus mengatur parameter CPU dan memori.

Sebaiknya tambahkan 256 unit CPU tambahan dan setidaknya 64 MiB memori ke CPU tugas dan memori Anda untuk wadah proxy Service Connect. Di AWS Fargate, jumlah memori terendah yang dapat Anda atur adalah memori 512 MiB. Di Amazon EC2, memori definisi tugas diperlukan.

- Untuk layanan, Anda mengatur konfigurasi log dalam konfigurasi Service Connect.
- Jika Anda mengharapkan tugas dalam layanan ini menerima lebih dari 500 permintaan per detik pada beban puncaknya, sebaiknya tambahkan 512 unit CPU ke CPU tugas Anda dalam definisi tugas ini untuk container proxy Service Connect.
- Jika Anda berharap untuk membuat lebih dari 100 layanan Service Connect di namespace atau 2000 tugas secara total di semua layanan Amazon ECS dalam namespace, sebaiknya tambahkan 128 MiB memori ke memori tugas Anda untuk container proxy Service Connect. Anda harus melakukan ini di setiap definisi tugas yang digunakan oleh semua layanan Amazon ECS di namespace.

Konfigurasi proxy

Aplikasi Anda terhubung ke proxy di wadah sespan dalam tugas yang sama dengan aplikasi. Amazon ECS mengonfigurasi tugas dan kontainer sehingga aplikasi hanya terhubung ke proxy saat aplikasi terhubung ke nama titik akhir di namespace yang sama. Semua lalu lintas lainnya tidak menggunakan proxy. Lalu lintas lainnya termasuk alamat IP di VPC yang sama, titik akhir AWS layanan, dan lalu lintas eksternal.

Penyeimbangan beban

Service Connect mengonfigurasi proxy untuk menggunakan strategi round-robin untuk load balancing antara tugas di titik akhir Service Connect. Proxy lokal yang ada dalam tugas dari mana koneksi berasal, memilih salah satu tugas dalam layanan client-server yang menyediakan titik akhir.

Misalnya, pertimbangkan tugas yang berjalan WordPress di layanan yang dikonfigurasi sebagai layanan klien di namespace yang disebut lokal. Ada layanan lain dengan 2 tugas yang menjalankan database MySQL. Layanan ini dikonfigurasi untuk menyediakan titik akhir yang dipanggil `mysql` melalui Service Connect di namespace yang sama. Dalam WordPress tugas, WordPress aplikasi terhubung ke database menggunakan nama endpoint. Koneksi ke nama ini masuk ke proxy yang berjalan dalam wadah sespan dalam tugas yang sama. Kemudian, proxy dapat terhubung ke salah satu tugas MySQL menggunakan strategi round-robin.

Strategi penyeimbangan beban: round-robin

Deteksi outlier

Fitur ini menggunakan data yang dimiliki proxy tentang koneksi gagal sebelumnya untuk menghindari pengiriman koneksi baru ke host yang memiliki koneksi gagal. Service Connect mengonfigurasi fitur deteksi outlier dari proxy untuk memberikan pemeriksaan kesehatan pasif.

Menggunakan contoh sebelumnya, proxy dapat terhubung ke salah satu tugas MySQL. Jika proxy membuat beberapa koneksi ke tugas MySQL tertentu, dan 5 atau lebih koneksi gagal dalam 30 detik terakhir, maka proxy menghindari tugas MySQL itu selama 30 hingga 300 detik.

Percobaan ulang

Service Connect mengonfigurasi proxy untuk mencoba lagi koneksi yang melewati proxy dan gagal, dan upaya kedua menghindari penggunaan host dari koneksi sebelumnya. Ini memastikan bahwa setiap koneksi melalui Service Connect tidak gagal karena alasan satu kali.

Jumlah percobaan ulang: 2

Waktu habis

Service Connect mengonfigurasi proxy untuk menunggu waktu maksimum agar aplikasi client-server Anda merespons. Nilai batas waktu default adalah 15 detik, tetapi dapat diperbarui.

Parameter opsional:

IdleTimeout - Jumlah waktu dalam detik koneksi tetap aktif saat idle. Nilai yang `0` dinonaktifkan`idleTimeout`.

`idleTimeout`Default untuk`HTTP/HTTP2/GRPC`adalah 5 menit.

`idleTimeout`Default untuk `TCP` adalah 1 jam.

perRequestTimeout- Jumlah waktu menunggu hulu untuk merespons dengan respons lengkap per permintaan. Nilai `0` dimatikan`perRequestTimeout`. Ini hanya dapat diatur ketika wadah `appProtocol` untuk aplikasi adalah`HTTP/HTTP2/GRPC`. Defaultnya adalah 15 detik.

> **Note**
>
> Jika `idleTimeout` diatur ke waktu yang kurang dari`perRequestTimeout`, koneksi akan ditutup ketika `idleTimeout` tercapai dan bukan`perRequestTimeout`.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Service Connect:

- Tugas yang berjalan di Fargate harus menggunakan versi platform Fargate Linux `1.4.0` atau lebih tinggi untuk menggunakan Service Connect.
- Versi agen Amazon ECS pada instance container harus `1.67.2` atau lebih tinggi.
- Instans penampung harus menjalankan Amazon ECS yang dioptimalkan Amazon Linux 2023 versi AMI atau yang lebih baru, atau `20230428` versi Amazon Linux 2 AMI Amazon ECS yang dioptimalkan untuk menggunakan Service Connect. `2.0.20221115` Versi ini memiliki agen Service Connect selain agen kontainer Amazon ECS. Untuk informasi selengkapnya tentang agen Service Connect, lihat Agen Amazon ECS Service Connect di GitHub.
- Instance kontainer harus memiliki `ecs:Poll` izin untuk sumber `arn:aws:ecs:`*`region`*`:`*`0123456789012`*`:task-set/`*`cluster`*`/*` daya. Jika Anda menggunakan`ecsInstanceRole`, Anda tidak perlu menambahkan izin tambahan. Kebijakan `AmazonEC2ContainerServiceforEC2Role` terkelola memiliki izin yang diperlukan. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.
- Tugas yang menggunakan mode `bridge` jaringan dan menggunakan Service Connect tidak mendukung parameter definisi `hostname` kontainer.
- Definisi tugas harus menetapkan batas memori tugas untuk menggunakan Service Connect. Untuk informasi selengkapnya, lihat Proksi Service Connect.
- Definisi tugas yang menetapkan batas memori kontainer tidak didukung.

  Anda dapat mengatur batas memori kontainer pada kontainer Anda, tetapi Anda harus mengatur batas memori tugas ke angka yang lebih besar dari jumlah batas memori kontainer. CPU dan memori tambahan dalam batas tugas yang tidak dialokasikan dalam batas kontainer digunakan oleh kontainer proxy Service Connect dan wadah lain yang tidak menetapkan batas kontainer. Untuk informasi selengkapnya, lihat Proksi Service Connect.
- Anda dapat mengonfigurasi Service Connect untuk menggunakan AWS Cloud Map namespace apa pun di Wilayah yang sama di wilayah yang sama. Akun AWS
- Setiap layanan hanya dapat dimiliki oleh satu namespace.
- Hanya tugas yang dibuat layanan yang didukung.
- Semua titik akhir harus unik dalam namespace.
- Semua nama penemuan harus unik dalam namespace.

- Anda harus menerapkan ulang layanan yang ada sebelum aplikasi dapat menyelesaikan titik akhir baru. Titik akhir baru yang ditambahkan ke namespace setelah penerapan terbaru tidak akan ditambahkan ke konfigurasi tugas. Untuk informasi selengkapnya, lihat the section called "Komponen Service Connect".
- Service Connect tidak menghapus ruang nama saat kluster dihapus. Anda harus menghapus ruang nama di. AWS Cloud Map
- Lalu lintas Application Load Balancer default ke routing melalui agen Service Connect dalam mode jaringan. `awsvpc` Jika Anda ingin lalu lintas non-layanan melewati agen Service Connect, gunakan `ingressPortOverride` parameter dalam konfigurasi layanan Service Connect Anda.
- Service Connect with TLS tidak mendukung mode jaringan Bridge. Hanya mode `awsvpc` jaringan yang didukung.

Service Connect tidak mendukung hal berikut:

- Kontainer Windows
- HTTP 1.0
- Tugas mandiri
- Layanan yang menggunakan jenis penyebaran yang blue/green didukung oleh CodeDeploy dan eksternal
- `External`instance container untuk Amazon ECS Anywhere tidak didukung dengan Service Connect.
- PPv2

Ikhtisar konfigurasi Amazon ECS Service Connect

Saat Anda menggunakan Service Connect, ada parameter yang perlu Anda konfigurasikan di sumber daya Anda.

Tabel berikut menjelaskan parameter konfigurasi untuk sumber daya Amazon ECS.

Parameter lokasi	Jenis aplikasi	Deskripsi	Wajib
Ketentuan tugas	Klien	Tidak ada perubahan yang tersedia untuk Service Connect dalam definisi tugas klien.	N/A
Ketentuan tugas	Server klien	Server harus menambahkan `name` bidang ke port di `portMappings` wadah. Untuk informasi selengkapnya, lihat `portMappings`	Ya
Ketentuan tugas	Server klien	Server secara opsional dapat menyediakan protokol aplikasi (misalnya , HTTP) untuk menerima metrik spesifik protokol untuk aplikasi server mereka (misalnya,). `HTTP 5xx`	Tidak
Definisi layanan	Klien	Layanan klien harus menambahkan a `serviceConnectConfiguration` untuk mengkonfigurasi namespace untuk bergabung. Namespace ini harus berisi semua layanan server yang perlu ditemukan oleh layanan ini. Untuk informasi selengkapnya, lihat `serviceConnectConfiguration` .	Ya
Definisi layanan	Server klien	Layanan server harus menambahkan a `serviceConnectConfiguration` untuk mengonfigurasi nama DNS, nomor port, dan namespace tempat layanan tersedia. Untuk informasi selengkapnya, lihat `serviceConnectConfiguration` .	Ya
Klaster	Klien	Cluster dapat menambahkan namespace Service Connect default.	Tidak

Parameter lokasi	Jenis aplikasi	Deskripsi	Wajib
		Layanan baru di cluster mewarisi namespace saat Service Connect dikonfigurasi dalam layanan.	
Klaster	Server klien	Tidak ada perubahan yang tersedia untuk Service Connect di cluster yang berlaku untuk layanan server. Definisi tugas server dan layanan harus mengatur konfigurasi masing-masing.	N/A

Ikhtisar langkah-langkah untuk mengkonfigurasi Service Connect

Langkah-langkah berikut memberikan gambaran umum tentang cara mengkonfigurasi Service Connect.

Important

- Service Connect membuat AWS Cloud Map layanan di akun Anda. Memodifikasi AWS Cloud Map sumber daya ini dengan registering/deregistering instance secara manual, mengubah atribut instance, atau menghapus layanan dapat menyebabkan perilaku tak terduga untuk lalu lintas aplikasi Anda atau penerapan berikutnya.
- Service Connect tidak mendukung link dalam definisi tugas.

1. Tambahkan nama port ke pemetaan port dalam definisi tugas Anda. Selain itu, Anda dapat mengidentifikasi protokol lapisan 7 aplikasi, untuk mendapatkan metrik tambahan.
2. Buat cluster dengan AWS Cloud Map namespace atau buat namespace secara terpisah. Untuk organisasi sederhana, buat cluster dengan nama yang Anda inginkan untuk namespace dan tentukan nama yang identik untuk namespace. Dalam hal ini, Amazon ECS membuat namespace HTTP baru dengan konfigurasi yang diperlukan. Service Connect tidak menggunakan atau membuat zona yang dihosting DNS di Amazon Route 53.
3. Konfigurasikan layanan untuk membuat titik akhir Service Connect di dalam namespace.

4. Menyebarkan layanan untuk membuat titik akhir. Amazon ECS menambahkan container proxy Service Connect ke setiap tugas, dan membuat titik akhir Service Connect. AWS Cloud Map Wadah ini tidak dikonfigurasi dalam definisi tugas, dan definisi tugas dapat digunakan kembali tanpa modifikasi untuk membuat beberapa layanan di namespace yang sama atau di beberapa ruang nama.
5. Menerapkan aplikasi klien sebagai layanan untuk terhubung ke titik akhir. Amazon ECS menghubungkannya ke titik akhir Service Connect melalui proxy Service Connect di setiap tugas.

   Aplikasi hanya menggunakan proxy untuk terhubung ke titik akhir Service Connect. Tidak ada konfigurasi tambahan untuk menggunakan proxy. Proxy melakukan penyeimbangan beban round-robin, deteksi outlier, dan percobaan ulang. Untuk informasi selengkapnya tentang proxy, lihatProksi Service Connect.
6. Pantau lalu lintas melalui proxy Service Connect di Amazon CloudWatch.

Konfigurasi cluster

Anda dapat mengatur namespace default untuk Service Connect saat membuat atau memperbarui klaster. Jika Anda menentukan nama namespace yang tidak ada di akun Wilayah AWS dan yang sama, namespace HTTP baru akan dibuat.

Jika Anda membuat klaster dan menentukan namespace Service Connect default, klaster menunggu dalam status `PROVISIONING` sementara Amazon ECS membuat namespace. Anda dapat melihat status klaster yang menunjukkan status namespace. `attachment` Lampiran tidak ditampilkan secara default di AWS CLI, Anda harus menambahkan `--include ATTACHMENTS` untuk melihatnya.

Konfigurasi layanan

Service Connect dirancang untuk memerlukan konfigurasi minimum. Anda perlu menetapkan nama untuk setiap pemetaan port yang ingin Anda gunakan dengan Service Connect dalam definisi tugas. Dalam layanan, Anda perlu mengaktifkan Service Connect dan memilih namespace untuk membuat layanan klien. Untuk membuat layanan client-server, Anda perlu menambahkan satu konfigurasi layanan Service Connect yang cocok dengan nama salah satu pemetaan port. Amazon ECS menggunakan kembali nomor port dan nama port dari definisi tugas untuk menentukan layanan dan titik akhir Service Connect. Untuk mengganti nilai tersebut, Anda dapat menggunakan parameter lain Discovery, DNS, dan Port di konsol, atau `discoveryName` dan`clientAliases`, masing-masing di Amazon ECS API.

Enkripsi lalu lintas Amazon ECS Service Connect

Amazon ECS Service Connect mendukung enkripsi lalu lintas otomatis dengan sertifikat Transport Layer Security (TLS) untuk layanan Amazon ECS. Saat Anda mengarahkan layanan Amazon ECS ke AWS Private Certificate Authority (AWS Private CA), Amazon ECS secara otomatis menyediakan sertifikat TLS untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda. Amazon ECS menghasilkan, memutar, dan mendistribusikan sertifikat TLS yang digunakan untuk enkripsi lalu lintas.

Enkripsi lalu lintas otomatis dengan Service Connect menggunakan kemampuan enkripsi terdepan di industri untuk mengamankan komunikasi antar-layanan Anda yang membantu Anda memenuhi persyaratan keamanan. Ini mendukung sertifikat AWS Private Certificate Authority TLS dengan `256-bit ECDSA` dan `2048-bit RSA` enkripsi. Anda juga memiliki kontrol penuh atas sertifikat pribadi dan kunci penandatanganan untuk membantu Anda memenuhi persyaratan kepatuhan. Secara default, TLS 1.3 didukung, tetapi TLS 1.0 - 1.2 tidak didukung. Service Connect mendukung TLS 1.3 dengan cipher berikut:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

> **Note**
>
> Untuk menggunakan TLS 1.3, Anda harus mengaktifkannya pada pendengar pada target. Hanya lalu lintas masuk dan keluar yang lewat melalui agen Amazon ECS dienkripsi.

Pemeriksaan kesehatan Service Connect dan Application Load Balancer

Anda dapat menggunakan Service Connect dengan pemeriksaan kesehatan Application Load Balancer dan enkripsi TLS 1.3.

Konfigurasi Application Load Balancer

Konfigurasikan Application Load Balancer dengan pengaturan berikut:

- Konfigurasikan pendengar TLS dengan kebijakan keamanan TLS 1.3 (seperti `ELBSecurityPolicy-TLS13 -1-2-2021-06`). Untuk informasi selengkapnya, lihat Kebijakan keamanan untuk Application Load Balancer Anda.

- Buat grup target dengan pengaturan berikut:
  - Atur protokol ke HTTPS
  - Lampirkan grup target ke pendengar TLS
  - Konfigurasikan port pemeriksaan kesehatan agar sesuai dengan port kontainer layanan Service Connect Anda

Konfigurasi Service Connect

Konfigurasikan layanan dengan pengaturan berikut:

- Konfigurasikan layanan untuk menggunakan mode `awsvpc` jaringan, karena mode `bridge` jaringan tidak didukung.
- Aktifkan Service Connect untuk layanan.
- Siapkan konfigurasi penyeimbang beban dengan pengaturan berikut:
  - Tentukan grup target yang Anda konfigurasikan untuk Application Load Balancer
  - Atur port kontainer agar sesuai dengan port kontainer layanan Service Connect TLS
- Hindari pengaturan `ingressPortOverride` untuk layanan. Untuk informasi selengkapnya, lihat ServiceConnectServicedi Referensi API Amazon Elastic Container Service.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Application Load Balancer, TLS dan Service Connect:

- Gunakan mode `awsvpc` jaringan alih-alih mode `bridge` jaringan untuk pemeriksaan kesehatan HTTPS saat menggunakan enkripsi Service Connect dengan TLS. Pemeriksaan kesehatan HTTP akan terus bekerja dengan `bridge` mode.
- Konfigurasikan port pemeriksaan kesehatan grup target agar sesuai dengan port kontainer layanan Service Connect, bukan port HTTPS default (443).

AWS Private Certificate Authority Sertifikat dan Service Connect

Anda harus memiliki peran IAM infrastruktur. Untuk informasi selengkapnya tentang peran ini, lihat peran IAM infrastruktur Amazon ECS.

AWS Private Certificate Authority mode untuk Service Connect

AWS Private Certificate Authority dapat berjalan dalam dua mode: tujuan umum dan berumur pendek.

- Tujuan umum - Sertifikat yang dapat dikonfigurasi dengan tanggal kedaluwarsa apa pun.
- Berumur pendek - Sertifikat dengan validitas maksimum tujuh hari.

Meskipun Amazon ECS mendukung kedua mode, kami sarankan untuk menggunakan sertifikat berumur pendek. Secara default, sertifikat berputar setiap lima hari, dan berjalan dalam mode berumur pendek menawarkan penghematan biaya yang signifikan dibandingkan tujuan umum.

Service Connect tidak mendukung pencabutan sertifikat dan sebaliknya memanfaatkan sertifikat berumur pendek dengan rotasi sertifikat yang sering. Anda memiliki wewenang untuk memodifikasi frekuensi rotasi, menonaktifkan, atau menghapus rahasia menggunakan rotasi terkelola di Secrets Manager, tetapi hal itu dapat disertai dengan konsekuensi yang mungkin berikut.

- Frekuensi Rotasi Lebih Pendek - Frekuensi rotasi yang lebih pendek menimbulkan biaya yang lebih tinggi karena AWS Private CA, AWS KMS dan Secrets Manager, dan Auto Scaling mengalami peningkatan beban kerja untuk rotasi.
- Frekuensi Rotasi Lebih Panjang - Komunikasi aplikasi Anda gagal jika frekuensi rotasi melebihi tujuh hari.
- Penghapusan Rahasia - Menghapus rahasia menghasilkan kegagalan rotasi dan berdampak pada komunikasi aplikasi pelanggan.

Jika rotasi rahasia Anda gagal, sebuah `RotationFailed` acara dipublikasikan di AWS CloudTrail. Anda juga dapat mengatur CloudWatch Alarm untuk`RotationFailed`.

> **Important**
>
> Jangan menambahkan replika Regions ke rahasia. Melakukannya mencegah Amazon ECS menghapus rahasia, karena Amazon ECS tidak memiliki izin untuk menghapus Wilayah dari replikasi. Jika Anda sudah menambahkan replikasi, jalankan perintah berikut.
>
> ```
> aws secretsmanager remove-regions-from-replication \
>  --secret-id SecretId \
>  --remove-replica-regions region-name
> ```

Otoritas Sertifikat Bawahan

Anda dapat membawa apa saja AWS Private CA, root atau bawahan, ke Service Connect TLS untuk menerbitkan sertifikat entitas akhir untuk layanan. Penerbit yang disediakan diperlakukan sebagai penandatangan dan akar kepercayaan di mana-mana. Anda dapat menerbitkan sertifikat entitas akhir untuk berbagai bagian aplikasi Anda dari bawahan yang berbeda. CAs Saat menggunakan AWS CLI, berikan Nama Sumber Daya Amazon (ARN) CA untuk membangun rantai kepercayaan.

Otoritas Sertifikat Lokal

Untuk menggunakan CA lokal, Anda membuat dan mengonfigurasi CA bawahan. AWS Private Certificate Authority Ini memastikan semua sertifikat TLS yang dikeluarkan untuk beban kerja Amazon ECS Anda berbagi rantai kepercayaan dengan beban kerja yang Anda jalankan di tempat dan dapat terhubung dengan aman.

> **Important**
>
> Tambahkan tag yang diperlukan `AmazonECSManaged : true` di AWS Private CA.

Infrastruktur sebagai kode

Saat menggunakan alat Service Connect TLS dengan Infrastructure as Code (IAc), penting untuk mengonfigurasi dependensi Anda dengan benar untuk menghindari masalah, seperti layanan yang macet dalam pengurasan. AWS KMS Kunci Anda, jika disediakan, peran IAM, dan AWS Private CA dependensi harus dihapus setelah layanan Amazon ECS Anda.

Service Connect dan Secrets Manager

Saat menggunakan Amazon ECS Service Connect dengan enkripsi TLS, layanan berinteraksi dengan Secrets Manager dengan cara berikut:

Service Connect menggunakan peran infrastruktur yang disediakan untuk membuat rahasia dalam Secrets Manager. Rahasia ini digunakan untuk menyimpan kunci pribadi terkait untuk sertifikat TLS Anda untuk mengenkripsi lalu lintas antara layanan Service Connect Anda.

> **Warning**
>
> Pembuatan dan pengelolaan rahasia ini secara otomatis oleh Service Connect menyederhanakan proses penerapan enkripsi TLS untuk layanan Anda. Namun, penting

untuk menyadari potensi implikasi keamanan. Peran IAM lain yang memiliki akses baca ke Secrets Manager mungkin dapat mengakses rahasia yang dibuat secara otomatis ini. Ini dapat mengekspos materi kriptografi sensitif kepada pihak yang tidak berwenang, jika kontrol akses tidak dikonfigurasi dengan benar.
Untuk mengurangi risiko ini, ikuti praktik terbaik berikut:

- Kelola dan batasi akses ke Secrets Manager dengan hati-hati, terutama untuk rahasia yang dibuat oleh Service Connect.
- Secara teratur mengaudit peran IAM dan izinnya untuk memastikan prinsip hak istimewa paling sedikit dipertahankan.

Saat memberikan akses baca ke Secrets Manager, pertimbangkan untuk mengecualikan kunci pribadi TLS yang dibuat oleh Service Connect. Anda dapat melakukan ini dengan menggunakan kondisi dalam kebijakan IAM Anda untuk mengecualikan rahasia ARNs yang cocok dengan pola:

```
"arn:aws:secretsmanager:::secret:ecs-sc!"
```

Contoh kebijakan IAM yang menyangkal `GetSecretValue` tindakan untuk semua rahasia dengan awalan: `ecs-sc!`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*"
        }
    ]
}
```

> **Note**
>
> Ini adalah contoh umum dan mungkin perlu disesuaikan berdasarkan kasus penggunaan spesifik dan konfigurasi AWS akun Anda. Selalu uji kebijakan IAM Anda secara menyeluruh untuk memastikan mereka menyediakan akses yang diinginkan sambil menjaga keamanan.

Dengan memahami bagaimana Service Connect berinteraksi dengan Secrets Manager, Anda dapat mengelola keamanan layanan Amazon ECS dengan lebih baik sambil memanfaatkan manfaat enkripsi TLS otomatis.

Service Connect dan AWS Key Management Service

Anda dapat menggunakan AWS Key Management Serviceuntuk mengenkripsi dan mendekripsi sumber daya Service Connect Anda. AWS KMS adalah layanan yang dikelola oleh AWS tempat Anda dapat membuat dan mengelola kunci kriptografi yang melindungi data Anda.

Saat menggunakan AWS KMS Service Connect, Anda dapat memilih untuk menggunakan kunci AWS milik yang AWS mengelola untuk Anda, atau Anda dapat memilih AWS KMS kunci yang ada. Anda juga dapat membuat AWS KMS kunci baru untuk digunakan.

Menyediakan kunci enkripsi Anda sendiri

Anda dapat menyediakan materi utama Anda sendiri, atau Anda dapat menggunakan penyimpanan kunci eksternal melalui AWS Key Management Service Impor kunci Anda sendiri AWS KMS, lalu tentukan Nama Sumber Daya Amazon (ARN) kunci tersebut di Amazon ECS Service Connect.

Berikut ini adalah contoh AWS KMS kebijakan. Ganti *user input* nilai dengan nilai Anda sendiri.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "id",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role-name"
      },
      "Action": [
```

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyPair"
      ],
      "Resource": "*"
    }
  ]
}
```

Untuk informasi selengkapnya tentang kebijakan utama, lihat Membuat kebijakan kunci di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Service Connect hanya mendukung AWS KMS kunci enkripsi simetris. Anda tidak dapat menggunakan jenis AWS KMS kunci lain untuk mengenkripsi sumber daya Service Connect. Untuk bantuan menentukan apakah AWS KMS kunci adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci KMS asimetris.

Untuk informasi selengkapnya tentang kunci enkripsi AWS Key Management Service simetris, lihat AWS KMS Kunci enkripsi simetris di Panduan AWS Key Management Service Pengembang.

Mengaktifkan TLS untuk Amazon ECS Service Connect

Anda mengaktifkan enkripsi lalu lintas saat membuat atau memperbarui layanan Service Connect.

Untuk mengaktifkan enkripsi lalu lintas untuk layanan di namespace yang ada menggunakan AWS Management Console

1. Anda harus memiliki peran IAM infrastruktur. Untuk informasi selengkapnya tentang peran ini, lihat peran IAM infrastruktur Amazon ECS.
2. Buka konsol di https://console.aws.amazon.com/ecs/v2.
3. Di panel navigasi, pilih Namespace.
4. Pilih Namespace dengan Layanan yang ingin Anda aktifkan enkripsi lalu lintas.
5. Pilih Layanan yang ingin Anda aktifkan enkripsi lalu lintas.
6. Pilih Perbarui Layanan di sudut kanan atas dan gulir ke bawah ke bagian Service Connect.

7. Pilih Aktifkan enkripsi lalu lintas di bawah informasi layanan Anda untuk mengaktifkan TLS.
8. Untuk peran Service Connect TLS, pilih peran IAM infrastruktur yang ada atau buat yang baru.
9. Untuk otoritas sertifikat Penandatangan, pilih otoritas sertifikat yang ada atau buat yang baru.

   Untuk informasi lebih lanjut, lihatAWS Private Certificate Authority Sertifikat dan Service Connect.
10. Untuk Pilih AWS KMS key, pilih kunci yang AWS dimiliki dan dikelola atau Anda dapat memilih kunci yang berbeda. Anda juga dapat memilih untuk membuat yang baru.

Untuk contoh menggunakan AWS CLI untuk mengkonfigurasi TLS untuk layanan Anda,Mengkonfigurasi Amazon ECS Service Connect dengan AWS CLI.

Memverifikasi TLS diaktifkan untuk Amazon ECS Service Connect

Service Connect memulai TLS di agen Service Connect dan menghentikannya di agen tujuan. Akibatnya, kode aplikasi tidak pernah melihat interaksi TLS. Gunakan langkah-langkah berikut untuk memverifikasi bahwa TLS diaktifkan.

1. Sertakan `openssl` CLI dalam gambar aplikasi Anda.
2. Aktifkan ECS Exec pada layanan Anda untuk terhubung ke tugas Anda melalui SSM. Sebagai alternatif, Anda dapat meluncurkan EC2 instans Amazon di VPC Amazon yang sama dengan layanannya.
3. Ambil IP dan port tugas dari layanan yang ingin Anda verifikasi. Anda dapat mengambil alamat IP tugas di AWS Cloud Map konsol. Informasi ada di halaman detail layanan untuk namespace.
4. Masuk ke salah satu tugas Anda menggunakan `execute-command` seperti pada contoh berikut. Sebagai alternatif, masuk ke EC2 instance Amazon yang dibuat di Langkah 2.

```
$ aws ecs execute-command --cluster cluster-name \
    --task task-id  \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

> **Note**
>
> Memanggil nama DNS secara langsung tidak mengungkapkan sertifikat.

5. Di shell yang terhubung, gunakan `openssl` CLI untuk memverifikasi dan melihat sertifikat yang dilampirkan pada tugas.

   Contoh:

```
openssl s_client -connect 10.0.147.43:6379 < /dev/null 2> /dev/null \
| openssl x509 -noout -text
```

   Contoh respons:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            <serial-number>
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: <issuer>
        Validity
            Not Before: Jan 23 21:38:12 2024 GMT
            Not After : Jan 30 22:38:12 2024 GMT
        Subject: <subject>
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    <pub>
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:redis.yelb-cftc
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:<key-id>

            X509v3 Subject Key Identifier:
                1D:<id>
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
```

```
    Signature Algorithm: ecdsa-with-SHA256
        <hash>
```

Mengkonfigurasi Amazon ECS Service Connect dengan AWS CLI

Anda dapat membuat layanan Amazon ECS untuk tugas Fargate yang menggunakan Service Connect dengan. AWS CLI

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Prasyarat

Berikut ini adalah prasyarat Service Connect:

- Verifikasi bahwa versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC, subnet, tabel rute, dan grup keamanan yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.
- Anda memiliki peran eksekusi tugas dengan nama `ecsTaskExecutionRole` dan kebijakan `AmazonECSTaskExecutionRolePolicy` terkelola dilampirkan ke peran. Peran ini memungkinkan Fargate untuk menulis log aplikasi NGINX dan log proxy Service Connect ke Amazon Logs. CloudWatch Untuk informasi selengkapnya, lihat Membuat peran eksekusi tugas .

Langkah 1: Buat cluster

Gunakan langkah-langkah berikut untuk membuat cluster dan namespace Amazon ECS Anda.

Untuk membuat cluster dan namespace Amazon ECS AWS Cloud Map

1. Buat klaster Amazon ECS bernama `tutorial` untuk digunakan. Parameter `--service-connect-defaults` menetapkan namespace default cluster. Dalam contoh keluaran, AWS

Cloud Map namespace nama `service-connect` tidak ada di akun ini dan Wilayah AWS, jadi namespace dibuat oleh Amazon ECS. Namespace dibuat AWS Cloud Map di akun, dan terlihat dengan semua ruang nama lainnya, jadi gunakan nama yang menunjukkan tujuannya.

```
aws ecs create-cluster --cluster-name tutorial --service-connect-defaults
 namespace=service-connect
```

Output:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "clusterName": "tutorial",
        "serviceConnectDefaults": {
            "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
        },
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": [],
        "attachments": [
            {
                "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "type": "sc",
                "status": "ATTACHING",
                "details": []
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
```

```
}
}
```

2. Verifikasi bahwa cluster dibuat:

```
aws ecs describe-clusters --clusters tutorial
```

Output:

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
            "clusterName": "tutorial",
            "serviceConnectDefaults": {
                "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
            },
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

3. (Opsional) Verifikasi bahwa namespace dibuat di. AWS Cloud Map Anda dapat menggunakan AWS Management Console atau AWS CLI konfigurasi normal saat ini dibuat AWS Cloud Map.

Misalnya, gunakan AWS CLI:

```
aws servicediscovery get-namespace --id ns-EXAMPLE
```

Output:

```
{
    "Namespace": {
        "Id": "ns-EXAMPLE",
        "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
EXAMPLE",
        "Name": "service-connect",
        "Type": "HTTP",
        "Properties": {
            "DnsProperties": {
                "SOA": {}
            },
            "HttpProperties": {
                "HttpName": "service-connect"
            }
        },
        "CreateDate": 1661749852.422,
        "CreatorRequestId": "service-connect"
    }
}
```

Langkah 2: Buat layanan untuk server

Fitur Service Connect ditujukan untuk menghubungkan beberapa aplikasi di Amazon ECS. Setidaknya salah satu dari aplikasi tersebut perlu menyediakan layanan web untuk terhubung. Pada langkah ini, Anda membuat:

- Definisi tugas yang menggunakan image kontainer NGINX resmi yang tidak dimodifikasi dan menyertakan konfigurasi Service Connect.
- Definisi layanan Amazon ECS yang mengonfigurasi Service Connect untuk menyediakan penemuan layanan dan proxy mesh layanan untuk lalu lintas ke layanan ini. Konfigurasi menggunakan kembali namespace default dari konfigurasi cluster untuk mengurangi jumlah konfigurasi layanan yang Anda buat untuk setiap layanan.
- Layanan Amazon ECS. Ini menjalankan satu tugas menggunakan definisi tugas, dan menyisipkan wadah tambahan untuk proxy Service Connect. Proxy mendengarkan port dari pemetaan port kontainer dari definisi tugas. Dalam aplikasi klien yang berjalan di Amazon ECS, proxy dalam tugas klien mendengarkan koneksi keluar ke nama port definisi tugas, nama penemuan layanan atau nama alias klien layanan, dan nomor port dari alias klien.

Untuk membuat layanan web dengan Service Connect

1. Daftarkan definisi tugas yang kompatibel dengan Fargate dan gunakan mode `awsvpc` jaringan. Ikuti langkah-langkah ini:

   a. Buat file yang diberi nama `service-connect-nginx.json` dengan isi definisi tugas berikut.

   Definisi tugas ini mengonfigurasi Service Connect dengan menambahkan `name` dan `appProtocol` parameter ke pemetaan port. Nama port membuat port ini lebih dapat diidentifikasi dalam konfigurasi layanan ketika beberapa port digunakan. Nama port juga digunakan secara default sebagai nama yang dapat ditemukan untuk digunakan oleh aplikasi lain di namespace.

   Definisi tugas berisi peran IAM tugas karena layanan mengaktifkan ECS Exec.

   > **Important**
   >
   > Definisi tugas ini menggunakan a `logConfiguration` untuk mengirim output nginx dari `stdout` dan `stderr` ke Amazon Logs. CloudWatch Peran eksekusi tugas ini tidak memiliki izin tambahan yang diperlukan untuk membuat grup CloudWatch log Log. Buat grup log di CloudWatch Log menggunakan AWS Management Console or AWS CLI. Jika Anda tidak ingin mengirim log nginx ke Log, Anda CloudWatch dapat menghapus file. `logConfiguration`
   > Ganti Akun AWS id dalam peran eksekusi tugas dengan Akun AWS id Anda.

```
{
    "family": "service-connect-nginx",
    "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
        "name": "webserver",
        "image": "public.ecr.aws/docker/library/nginx:latest",
        "cpu": 100,
        "portMappings": [
            {
```

```
                "name": "nginx",
                "containerPort": 80,
                "protocol": "tcp",
                "appProtocol": "http"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "nginx"
            }
        }
        }
    ],
    "cpu": "256",
    "memory": "512"
}
```

b. Daftarkan definisi tugas menggunakan `service-connect-nginx.json` file:

```
aws ecs register-task-definition --cli-input-json file://service-connect-nginx.json
```

2. Buat layanan:

a. Buat file yang diberi nama `service-connect-nginx-service.json` dengan konten layanan Amazon ECS yang Anda buat. Contoh ini menggunakan definisi tugas yang dibuat pada langkah sebelumnya. `awsvpcConfiguration` diperlukan karena ketentuan tugas contoh menggunakan mode jaringan `awsvpc`.

Saat Anda membuat layanan ECS, tentukan jenis peluncuran Fargate, dan versi platform `LATEST` yang mendukung Service Connect. Itu `securityGroups` dan `subnets` harus milik VPC yang memiliki persyaratan untuk menggunakan Amazon ECS. Anda dapat memperoleh grup keamanan dan subnet IDs dari Konsol VPC Amazon.

Layanan ini mengkonfigurasi Service Connect dengan menambahkan `serviceConnectConfiguration` parameter. Namespace tidak diperlukan karena cluster memiliki namespace default yang dikonfigurasi. Aplikasi klien yang berjalan di

ECS di namespace terhubung ke layanan ini dengan menggunakan `portName` dan port di. `clientAliases` Misalnya, layanan ini dapat dijangkau menggunakan`http://nginx:80/`, karena nginx menyediakan halaman selamat datang di lokasi root. `/` Aplikasi eksternal yang tidak berjalan di Amazon ECS atau tidak berada di namespace yang sama dapat menjangkau aplikasi ini melalui proxy Service Connect dengan menggunakan alamat IP tugas dan nomor port dari definisi tugas. Untuk `tls` konfigurasi Anda, tambahkan sertifikat `arn` untuk peran IAM Anda `awsPcaAuthorityArn`, `kmsKey` Anda, dan `roleArn` Anda.

Layanan ini menggunakan a `logConfiguration` untuk mengirim layanan menghubungkan output proxy dari `stdout` dan `stderr` ke Amazon CloudWatch Logs. Peran eksekusi tugas ini tidak memiliki izin tambahan yang diperlukan untuk membuat grup CloudWatch log Log. Buat grup log di CloudWatch Log menggunakan AWS Management Console or AWS CLI. Kami menyarankan Anda membuat grup log ini dan menyimpan log proxy di CloudWatch Log. Jika Anda tidak ingin mengirim log proxy ke CloudWatch Log, Anda dapat menghapus`logConfiguration`.

```
{
    "cluster": "tutorial",
    "deploymentConfiguration": {
        "maximumPercent": 200,
        "minimumHealthyPercent": 0
    },
    "deploymentController": {
        "type": "ECS"
    },
    "desiredCount": 1,
    "enableECSManagedTags": true,
    "enableExecuteCommand": true,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "assignPublicIp": "ENABLED",
            "securityGroups": [
                "sg-EXAMPLE"
            ],
            "subnets": [
                "subnet-EXAMPLE",
                "subnet-EXAMPLE",
                "subnet-EXAMPLE"
```

```
            ]
           }
    },
    "platformVersion": "LATEST",
    "propagateTags": "SERVICE",
    "serviceName": "service-connect-nginx-service",
    "serviceConnectConfiguration": {
        "enabled": true,
        "services": [
            {
                "portName": "nginx",
                "clientAliases": [
                    {
                        "port": 80
                    }
                ],
                "tls": {
                   "issuerCertificateAuthority": {
                      "awsPcaAuthorityArn": "certificateArn"
                   },
                   "kmsKey": "kmsKey",
                   "roleArn": "iamRoleArn"
                }
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "service-connect-proxy"
            }
        }
    },
    "taskDefinition": "service-connect-nginx"
}
```

b. Buat layanan menggunakan `service-connect-nginx-service.json` file:

```
aws ecs create-service --cluster tutorial --cli-input-json file://service-connect-nginx-service.json
```

Output:


```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/tutorial/
service-connect-nginx-service",
        "serviceName": "service-connect-nginx-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
service-connect-nginx:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 0
        },
        "deployments": [
            {
                "id": "ecs-svc/3763308422771520962",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/service-connect-nginx:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": 1661210032.602,
                "updatedAt": 1661210032.602,
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
```

```
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "assignPublicIp": "ENABLED",
                        "securityGroups": [
                            "sg-EXAMPLE"
                        ],
                        "subnets": [
                            "subnet-EXAMPLEf",
                            "subnet-EXAMPLE",
                            "subnet-EXAMPLE"
                        ]
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-
svc/3763308422771520962 in progress.",
                "failedLaunchTaskCount": 0,
                "replacedTaskCount": 0,
                "serviceConnectConfiguration": {
                    "enabled": true,
                    "namespace": "service-connect",
                    "services": [
                        {
                            "portName": "nginx",
                            "clientAliases": [
                                {
                                    "port": 80
                                }
                            ]
                        }
                    ],
                    "logConfiguration": {
                        "logDriver": "awslogs",
                        "options": {
                            "awslogs-group": "/ecs/service-connect-proxy",
                            "awslogs-region": "us-west-2",
                            "awslogs-stream-prefix": "service-connect-proxy"
                        },
                        "secretOptions": []
                    }
                },
                "serviceConnectResources": [
                    {
                        "discoveryName": "nginx",
```

```
                        "discoveryArn": "arn:aws:servicediscovery:us-
west-2:123456789012:service/srv-EXAMPLE"
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "version": 0,
        "events": [],
        "createdAt": 1661210032.602,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "assignPublicIp": "ENABLED",
                "securityGroups": [
                    "sg-EXAMPLE"
                ],
                "subnets": [
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE"
                ]
            }
        },
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": true,
        "propagateTags": "SERVICE",
        "enableExecuteCommand": true
    }
}
```

`serviceConnectConfiguration`Yang Anda berikan muncul di dalam penyebaran pertama output. Saat Anda membuat perubahan pada layanan ECS dengan cara yang perlu melakukan perubahan pada tugas, penerapan baru dibuat oleh Amazon ECS.

Langkah 3: Verifikasi bahwa Anda dapat terhubung

Untuk memverifikasi bahwa Service Connect dikonfigurasi dan berfungsi, ikuti langkah-langkah berikut untuk menyambung ke layanan web dari aplikasi eksternal. Kemudian, lihat metrik tambahan dalam CloudWatch proxy Service Connect yang dibuat.

Untuk terhubung ke layanan web dari aplikasi eksternal

- Connect ke alamat IP tugas dan port kontainer menggunakan alamat IP tugas

  Gunakan AWS CLI untuk mendapatkan ID tugas, menggunakan file`aws ecs list-tasks --cluster tutorial`.

  Jika subnet dan grup keamanan Anda mengizinkan lalu lintas dari internet publik di port dari definisi tugas, Anda dapat terhubung ke IP publik dari komputer Anda. IP publik tidak tersedia dari `describe-tasks`, jadi langkah-langkahnya melibatkan pergi ke Amazon EC2 AWS Management Console atau AWS CLI untuk mendapatkan detail dari elastic network interface.

  Dalam contoh ini, EC2 instance Amazon di VPC yang sama menggunakan IP pribadi tugas tersebut. Aplikasi ini nginx, tetapi `server: envoy` header menunjukkan bahwa proxy Service Connect digunakan. Proxy Service Connect mendengarkan pada port kontainer dari definisi tugas.

```
$ curl -v 10.0.19.50:80/
*   Trying 10.0.19.50:80...
* Connected to 10.0.19.50 (10.0.19.50) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.0.19.50
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< server: envoy
< date: Tue, 23 Aug 2022 03:53:06 GMT
< content-type: text/html
< content-length: 612
< last-modified: Tue, 16 Apr 2019 13:08:19 GMT
< etag: "5cb5d3c3-264"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 0
```

```
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Untuk melihat metrik Service Connect

Proxy Service Connect membuat metrik aplikasi (HTTP, HTTP2, gRPC, atau koneksi TCP) dalam metrik. CloudWatch Saat Anda menggunakan CloudWatch konsol, lihat dimensi metrik tambahan DiscoveryName, (DiscoveryName,, ClusterName) ServiceName, dan (TargetDiscoveryName, TargetDiscoveryName ServiceName, ClusterName) di bawah namespace Amazon ECS. Untuk informasi selengkapnya tentang metrik dan dimensi ini, lihat Melihat Metrik yang Tersedia di Panduan Pengguna CloudWatch Log Amazon.

Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS

Layanan Amazon ECS Anda secara opsional dapat dikonfigurasi untuk menggunakan penemuan layanan Amazon ECS. Penemuan layanan menggunakan tindakan AWS Cloud Map API untuk mengelola ruang nama HTTP dan DNS untuk layanan Amazon ECS Anda. Untuk informasi selengkapnya, lihat Apa yang AWS Cloud Map Ada di Panduan AWS Cloud Map Pengembang.

Penemuan layanan tersedia di AWS Wilayah berikut:

Nama Wilayah	Wilayah
AS Timur (Virginia Utara)	us-east-1
US East (Ohio)	us-east-2
AS Barat (California Utara)	us-west-1
US West (Oregon)	as-barat-2
Afrika (Cape Town)	af-selatan-1
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pasifik (Hyderabad)	ap-south-2
Asia Pasifik (Tokyo)	ap-northeast-1
Asia Pasifik (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pasifik (Singapura)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pasifik (Jakarta)	ap-southeast-3
Asia Pacific (Melbourne)	ap-southeast-4

Nama Wilayah	Wilayah
Kanada (Pusat)	ca-central-1
Kanada Barat (Calgary)	ca-west-1
Tiongkok (Beijing)	cn-north-1
China (Ningxia)	cn-northwest-1
Europe (Frankfurt)	eu-central-1
Eropa (Zürich)	eu-central-2
Eropa (Irlandia)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Eropa (Milan)	eu-south-1
Eropa (Stockholm)	eu-north-1
Israel (Tel Aviv)	il-central-1
Eropa (Spanyol)	eu-south-2
Timur Tengah (UEA)	me-central-1
Timur Tengah (Bahrain)	me-selatan-1
Amerika Selatan (São Paulo)	sa-east-1
AWS GovCloud (AS-Timur)	us-gov-east-1
AWS GovCloud (AS-Barat)	us-gov-west-1

Konsep Penemuan Layanan

Penemuan layanan terdiri dari komponen-komponen berikut:

- Ruang nama penemuan layanan: Grup logis layanan penemuan layanan yang berbagi nama domain yang sama, seperti`example.com`, tempat Anda ingin merutekan lalu lintas. Anda dapat membuat namespace dengan panggilan ke `aws servicediscovery create-private-dns-namespace` perintah atau di konsol Amazon ECS. Anda dapat menggunakan `aws servicediscovery list-namespaces` perintah untuk melihat informasi ringkasan tentang ruang nama yang dibuat oleh akun saat ini. Untuk informasi selengkapnya tentang perintah penemuan layanan, lihat `create-private-dns-namespace` dan `list-namespaces` di Panduan AWS CLI Referensi AWS Cloud Map (penemuan layanan).
- Layanan penemuan layanan: Ada dalam namespace penemuan layanan dan terdiri dari nama layanan dan konfigurasi DNS untuk namespace. Layanan tersebut menyediakan komponen inti berikut:
  - Registri layanan: Memungkinkan Anda mencari layanan melalui tindakan DNS atau AWS Cloud Map API dan mendapatkan kembali satu atau lebih titik akhir yang tersedia yang dapat digunakan untuk terhubung ke layanan.
- Contoh penemuan layanan: Ada dalam layanan penemuan layanan dan terdiri dari atribut yang terkait dengan setiap layanan Amazon ECS di direktori layanan.
  - Atribut instans: Metadata berikut ditambahkan sebagai atribut khusus untuk setiap layanan Amazon ECS yang dikonfigurasi untuk menggunakan penemuan layanan:
    - **`AWS_INSTANCE_IPV4`**— Sebagai `A` catatan, IPv4 alamat yang dikembalikan Route 53 sebagai respons terhadap kueri DNS dan ditampilkan AWS Cloud Map saat menemukan detail instance, misalnya,. `192.0.2.44`
    - **`AWS_INSTANCE_PORT`**— Nilai port yang terkait dengan layanan penemuan layanan.
    - **`AVAILABILITY_ZONE`**— Zona Ketersediaan tempat tugas diluncurkan. Untuk tugas yang menggunakan tipe EC2 peluncuran, ini adalah Availability Zone di mana instance container ada. Untuk tugas yang menggunakan tipe peluncuran Fargate, ini adalah Availability Zone di mana elastic network interface ada.
    - **`REGION`**— Wilayah di mana tugas itu ada.
    - **`ECS_SERVICE_NAME`**— Nama layanan Amazon ECS tempat tugas tersebut berada.
    - **`ECS_CLUSTER_NAME`**— Nama cluster Amazon ECS tempat tugas tersebut berada.
    - **`EC2_INSTANCE_ID`**— ID dari instance kontainer tempat tugas ditempatkan. Atribut kustom ini tidak ditambahkan jika tugas menggunakan jenis peluncuran Fargate.
    - **`ECS_TASK_DEFINITION_FAMILY`**— Keluarga definisi tugas yang digunakan tugas.

  - **`ECS_TASK_SET_EXTERNAL_ID`**— Jika set tugas dibuat untuk penyebaran eksternal dan dikaitkan dengan registri penemuan layanan, maka `ECS_TASK_SET_EXTERNAL_ID` atribut akan berisi ID eksternal dari set tugas.
- Pemeriksaan kesehatan Amazon ECS: Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer berkala. Jika titik akhir tidak lulus pemeriksaan kondisi, maka titik akhir akan dihapus dari perutean DNS dan ditandai dengan kondisi tidak baik.

Pertimbangan penemuan layanan

Berikut ini harus dipertimbangkan saat menggunakan penemuan layanan:

- Penemuan layanan didukung untuk tugas di Fargate yang menggunakan platform versi 1.1.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.
- Layanan yang dikonfigurasi untuk menggunakan penemuan layanan memiliki batas 1.000 tugas per layanan. Hal ini dikarenakan kuota layanan Route 53.
- Alur kerja Create Service di konsol Amazon ECS hanya mendukung pendaftaran layanan ke ruang nama DNS pribadi. Ketika namespace DNS AWS Cloud Map pribadi dibuat, zona host pribadi Route 53 akan dibuat secara otomatis.
- Atribut DNS VPC harus dikonfigurasi agar resolusi DNS berhasil. Untuk informasi tentang cara mengonfigurasi atribut, lihat Dukungan DNS di VPC Anda di Panduan Pengguna Amazon VPC.
- Catatan DNS yang dibuat untuk layanan penemuan layanan selalu mendaftar dengan alamat IP pribadi untuk tugas tersebut, bukan alamat IP publik, bahkan ketika ruang nama publik digunakan.
- Penemuan layanan mengharuskan tugas menentukan mode `awsvpc`bridge,, atau `host` jaringan (`none`tidak didukung).
- Jika definisi tugas layanan menggunakan mode `awsvpc` jaringan, Anda dapat membuat kombinasi `A` atau catatan SRV untuk setiap tugas layanan. Jika Anda menggunakan catatan SRV, maka port diperlukan.
- Jika definisi tugas layanan menggunakan mode `host` jaringan `bridge` atau, catatan SRV adalah satu-satunya jenis catatan DNS yang didukung. Buat catatan SRV untuk setiap tugas layanan. Catatan SRV harus menentukan kombinasi nama kontainer dan port kontainer dari penentuan tugas.
- Catatan DNS untuk layanan penemuan layanan dapat ditanyakan dalam VPC Anda. Mereka menggunakan format berikut: `<service-discovery-service-name>.<service-discovery-namespace>`.

- Saat melakukan kueri DNS pada nama layanan, A catatan mengembalikan satu set alamat IP yang sesuai dengan tugas Anda. Catatan SRV mengembalikan satu set alamat IP dan port untuk setiap tugas.
- Jika Anda memiliki delapan atau kurang catatan sehat, Route 53 merespons semua kueri DNS dengan semua catatan sehat.
- Saat semua catatan tidak sehat, Route 53 akan merespons kueri DNS dengan hingga delapan catatan yang tidak sehat.
- Anda dapat mengonfigurasi penemuan layanan untuk layanan yang berada di belakang penyeimbang beban, tetapi lalu lintas penemuan layanan selalu diarahkan ke tugas dan bukan penyeimbang beban.
- Penemuan layanan tidak mendukung penggunaan Classic Load Balancer.
- Kami menyarankan Anda menggunakan pemeriksaan kesehatan tingkat kontainer yang dikelola oleh Amazon ECS untuk layanan penemuan layanan Anda.
  - HealthCheckCustomConfigAmazon ECS mengelola pemeriksaan kesehatan atas nama Anda. Amazon ECS menggunakan informasi dari wadah dan pemeriksaan kesehatan, dan status tugas Anda, untuk memperbarui kesehatan. AWS Cloud Map Ini ditentukan menggunakan `--health-check-custom-config` parameter saat membuat layanan penemuan layanan Anda. Untuk informasi selengkapnya, lihat HealthCheckCustomConfig di dalam Referensi API AWS Cloud Map .
- Sumber AWS Cloud Map daya yang dibuat saat penemuan layanan digunakan harus dibersihkan secara manual.
- Tugas dan instance didaftarkan `UNHEALTHY` sampai pemeriksaan kesehatan kontainer mengembalikan nilai. Jika pemeriksaan kesehatan lulus, status diperbarui ke`HEALTHY`. Jika pemeriksaan kesehatan kontainer gagal, instance penemuan layanan dideregistrasi.

Harga penemuan layanan

Pelanggan yang menggunakan penemuan layanan Amazon ECS dikenakan biaya untuk sumber daya Route 53 dan operasi API AWS Cloud Map penemuan. Ini melibatkan biaya untuk membuat zona yang dihosting Route 53 dan kueri ke registri layanan. Untuk informasi selengkapnya, lihat AWS Cloud Map Harga di Panduan AWS Cloud Map Pengembang.

Amazon ECS melakukan pemeriksaan kesehatan tingkat kontainer dan memaparkannya ke operasi API pemeriksaan kesehatan AWS Cloud Map khusus. Layanan tersebut saat ini tersedia untuk pelanggan tanpa biaya tambahan. Jika Anda mengonfigurasi pemeriksaan kesehatan jaringan

tambahan untuk tugas yang terpapar publik, Anda dikenakan biaya untuk pemeriksaan kesehatan tersebut.

Membuat layanan Amazon ECS yang menggunakan Service Discovery

Pelajari cara membuat layanan yang berisi tugas Fargate yang menggunakan penemuan layanan dengan. AWS CLI

Untuk daftar penemuan layanan dukungan Wilayah AWS tersebut, lihatGunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.

Untuk informasi tentang Daerah yang mendukung Fargate, lihat. the section called “AWS Wilayah Fargate”

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Prasyarat

Sebelum Anda memulai tutorial ini, pastikan bahwa prasyarat berikut terpenuhi:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.
- Langkah-langkah yang dijelaskan di dalamnya Siapkan untuk menggunakan Amazon ECS sudah lengkap.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda telah membuat setidaknya satu VPC dan satu grup keamanan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Langkah 1: Buat sumber daya Penemuan Layanan di AWS Cloud Map

Ikuti langkah-langkah berikut untuk membuat namespace penemuan layanan dan layanan penemuan layanan Anda:

1. Buat namespace penemuan layanan Cloud Map pribadi. Contoh ini menciptakan namespace yang disebut. `tutorial` Ganti *vpc-abcd1234* dengan ID salah satu yang sudah ada VPCs.

```
aws servicediscovery create-private-dns-namespace \
      --name tutorial \
      --vpc vpc-abcd1234
```

   Output dari perintah ini adalah sebagai berikut.

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa namespace pribadi berhasil dibuat. Catat ID namespace karena Anda menggunakannya dalam perintah berikutnya.

```
aws servicediscovery get-operation \
      --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e
```

   Outputnya adalah sebagai berikut.

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
}
```

3. Menggunakan `NAMESPACE` ID dari output dari langkah sebelumnya, buat layanan penemuan layanan. Contoh ini menciptakan layanan bernama`myapplication`. Catat ID layanan dan ARN karena Anda menggunakannya dalam perintah berikutnya.

```
aws servicediscovery create-service \
```

```
      --name myapplication \
      --dns-config "NamespaceId="ns-
uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]" \
      --health-check-custom-config FailureThreshold=1
```

Outputnya adalah sebagai berikut.

```
{
    "Service": {
       "Id": "srv-utcrh6wavdkggqtk",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk",
        "Name": "myapplication",
        "DnsConfig": {
            "NamespaceId": "ns-uejictsjen2i4eeg",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 300
                }
            ]
        },
        "HealthCheckCustomConfig": {
            "FailureThreshold": 1
        },
        "CreatorRequestId": "e49a8797-b735-481b-a657-b74d1d6734eb"
    }
}
```

Langkah 2: Buat sumber daya Amazon ECS

Ikuti langkah-langkah berikut untuk membuat klaster Amazon ECS, definisi tugas, dan layanan:

1. Buat cluster Amazon ECS. Contoh ini menciptakan sebuah cluster yang diberi nama`tutorial`.

```
aws ecs create-cluster \
      --cluster-name tutorial
```

2. Daftarkan definisi tugas yang kompatibel dengan Fargate dan gunakan mode `awsvpc` jaringan. Ikuti langkah-langkah ini:

a. Buat file yang diberi nama `fargate-task.json` dengan isi definisi tugas berikut.

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "public.ecr.aws/docker/library/httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample
 App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a
 container in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/
htdocs/index.html && httpd-foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

b. Daftarkan definisi tugas menggunakan`fargate-task.json`.

```
aws ecs register-task-definition \
      --cli-input-json file://fargate-task.json
```

3. Buat layanan ECS dengan mengikuti langkah-langkah berikut:

   a. Buat file yang diberi nama `ecs-service-discovery.json` dengan konten layanan ECS yang Anda buat. Contoh ini menggunakan definisi tugas yang dibuat pada langkah sebelumnya. `awsvpcConfiguration` diperlukan karena ketentuan tugas contoh menggunakan mode jaringan `awsvpc`.

      Saat Anda membuat layanan ECS, tentukan jenis peluncuran Fargate, dan versi platform `LATEST` yang mendukung penemuan layanan. Ketika layanan penemuan layanan dibuat di AWS Cloud Map , `registryArn` adalah ARN dikembalikan. Itu `securityGroups` dan `subnets` harus milik VPC yang digunakan untuk membuat namespace Cloud Map. Anda dapat memperoleh grup keamanan dan subnet IDs dari Konsol VPC Amazon.

```
{
    "cluster": "tutorial",
    "serviceName": "ecs-service-discovery",
    "taskDefinition": "tutorial-task-def",
    "serviceRegistries": [
       {
          "registryArn":
 "arn:aws:servicediscovery:region:aws_account_id:service/srv-utcrh6wavdkggqtk"
       }
    ],
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234" ]
       }
    },
    "desiredCount": 1
}
```

   b. Buat layanan ECS Anda menggunakan`ecs-service-discovery.json`.

```
aws ecs create-service \
      --cli-input-json file://ecs-service-discovery.json
```

Langkah 3: Verifikasi Penemuan Layanan di AWS Cloud Map

Anda dapat memverifikasi bahwa semuanya dibuat dengan benar dengan menanyakan informasi penemuan layanan Anda. Setelah penemuan layanan dikonfigurasi, Anda dapat menggunakan operasi AWS Cloud Map API, atau menelepon `dig` dari instance dalam VPC Anda. Ikuti langkah-langkah ini:

1. Dengan menggunakan ID layanan penemuan layanan, cantumkan instance penemuan layanan. Catat ID instance (ditandai dengan huruf tebal) untuk pembersihan sumber daya.

```
aws servicediscovery list-instances \
      --service-id srv-utcrh6wavdkggqtk
```

   Outputnya adalah sebagai berikut.

```
{
    "Instances": [
        {
            "Id": "16becc26-8558-4af1-9fbd-f81be062a266",
            "Attributes": {
                "AWS_INSTANCE_IPV4": "172.31.87.2"
                "AWS_INSTANCE_PORT": "80",
                "AVAILABILITY_ZONE": "us-east-1a",
                "REGION": "us-east-1",
                "ECS_SERVICE_NAME": "ecs-service-discovery",
                "ECS_CLUSTER_NAME": "tutorial",
                "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
            }
        }
    ]
}
```

2. Gunakan namespace penemuan layanan, layanan, dan parameter tambahan seperti nama cluster ECS untuk menanyakan detail tentang instance penemuan layanan.

```
aws servicediscovery discover-instances \
      --namespace-name tutorial \
      --service-name myapplication \
      --query-parameters ECS_CLUSTER_NAME=tutorial
```

3. Catatan DNS yang dibuat di zona host Route 53 untuk layanan penemuan layanan dapat ditanyakan dengan perintah berikut: AWS CLI

   a. Menggunakan ID namespace, dapatkan informasi tentang namespace, yang mencakup ID zona yang dihosting Route 53.

   ```
   aws servicediscovery \
         get-namespace --id ns-uejictsjen2i4eeg
   ```

   Outputnya adalah sebagai berikut.

   ```
   {
       "Namespace": {
           "Id": "ns-uejictsjen2i4eeg",
           "Arn": "arn:aws:servicediscovery:region:aws_account_id:namespace/ns-
   uejictsjen2i4eeg",
           "Name": "tutorial",
           "Type": "DNS_PRIVATE",
           "Properties": {
                "DnsProperties": {
                   "HostedZoneId": "Z35JQ4ZFDRYPLV"
               }
           },
           "CreateDate": 1519777852.502,
           "CreatorRequestId": "9049a1d5-25e4-4115-8625-96dbda9a6093"
       }
   }
   ```

   b. Menggunakan ID zona yang dihosting Route 53 dari langkah sebelumnya (lihat teks dalam huruf tebal), dapatkan catatan sumber daya yang ditetapkan untuk zona yang dihosting.

   ```
   aws route53 list-resource-record-sets \
         --hosted-zone-id Z35JQ4ZFDRYPLV
   ```

4. Anda juga dapat menanyakan DNS dari instance dalam `dig` VPC Anda menggunakan.

   ```
   dig +short myapplication.tutorial
   ```

Langkah 4: Membersihkan

Setelah selesai dengan tutorial ini, bersihkan sumber daya terkait untuk menghindari biaya untuk sumber daya yang tidak digunakan. Ikuti langkah-langkah ini:

1. Batalkan pendaftaran instance layanan penemuan layanan menggunakan ID layanan dan ID instans yang Anda catat sebelumnya.

```
aws servicediscovery deregister-instance \
      --service-id srv-utcrh6wavdkggqtk \
      --instance-id 16becc26-8558-4af1-9fbd-f81be062a266
```

   Outputnya adalah sebagai berikut.

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa instance layanan penemuan layanan berhasil dideregistrasi.

```
aws servicediscovery get-operation \
      --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv
```

```
{
  "Operation": {
        "Id": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv",
        "Type": "DEREGISTER_INSTANCE",
        "Status": "SUCCESS",
        "CreateDate": 1525984073.707,
        "UpdateDate": 1525984076.426,
        "Targets": {
            "INSTANCE": "16becc26-8558-4af1-9fbd-f81be062a266",
            "ROUTE_53_CHANGE_ID": "C5NSRG1J4I1FH",
            "SERVICE": "srv-utcrh6wavdkggqtk"
        }
    }
}
```

3. Hapus layanan penemuan layanan menggunakan ID layanan.

```
aws servicediscovery delete-service \
      --id srv-utcrh6wavdkggqtk
```

4. Hapus namespace penemuan layanan menggunakan ID namespace.

```
aws servicediscovery delete-namespace \
      --id ns-uejictsjen2i4eeg
```

Outputnya adalah sebagai berikut.

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. Menggunakan `OperationId` dari output dari langkah sebelumnya, verifikasi bahwa namespace penemuan layanan berhasil dihapus.

```
aws servicediscovery get-operation \
      --operation-id c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj
```

Outputnya adalah sebagai berikut.

```
{
    "Operation": {
        "Id": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj",
        "Type": "DELETE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1525984602.211,
        "UpdateDate": 1525984602.558,
        "Targets": {
            "NAMESPACE": "ns-rymlehshst7hhukh",
            "ROUTE_53_CHANGE_ID": "CJP2A2M86XW3O"
        }
    }
}
```

6. Perbarui hitungan yang diinginkan untuk layanan Amazon ECS ke`0`. Anda harus melakukan ini untuk menghapus layanan di langkah berikutnya.

```
aws ecs update-service \
```

```
      --cluster tutorial \
      --service ecs-service-discovery \
      --desired-count 0
```

7. Hapus layanan Amazon ECS.

```
aws ecs delete-service \
      --cluster tutorial \
      --service ecs-service-discovery
```

8. Hapus cluster Amazon ECS.

```
aws ecs delete-cluster \
      --cluster tutorial
```

Gunakan Amazon VPC Lattice untuk menghubungkan, mengamati, dan mengamankan layanan Amazon ECS Anda

Amazon VPC Lattice adalah layanan jaringan aplikasi yang dikelola sepenuhnya yang memungkinkan pelanggan Amazon ECS untuk mengamati, mengamankan, dan memantau aplikasi yang dibangun di seluruh layanan AWS komputasi, VPCs, dan akun — tanpa memerlukan perubahan kode apa pun.

VPC Lattice menggunakan kelompok target, yang merupakan kumpulan sumber daya komputasi. Target ini menjalankan aplikasi atau layanan Anda dan dapat berupa EC2 instans Amazon, alamat IP, fungsi Lambda, dan Application Load Balancer. Dengan mengaitkan layanan Amazon ECS mereka dengan grup target VPC Lattice, pelanggan sekarang dapat mengaktifkan tugas Amazon ECS sebagai target IP di VPC Lattice. Amazon ECS secara otomatis mendaftarkan tugas ke grup target VPC Lattice saat tugas untuk layanan terdaftar diluncurkan.

> **Note**
>
> Saat menggunakan lima konfigurasi VPC Lattice, waktu penerapan Anda mungkin sedikit lebih lama daripada saat menggunakan konfigurasi yang lebih sedikit.

Aturan pendengar digunakan untuk meneruskan lalu lintas ke grup target tertentu saat kondisi terpenuhi. Pendengar memeriksa permintaan koneksi menggunakan protokol pada port yang Anda

konfigurasikan. Layanan merutekan permintaan ke target terdaftar berdasarkan aturan yang Anda tentukan saat mengonfigurasi listener Anda.

Amazon ECS juga secara otomatis menggantikan tugas jika menjadi tidak sehat menurut pemeriksaan kesehatan VPC Lattice. Setelah dikaitkan dengan VPC Lattice, pelanggan Amazon ECS juga dapat memanfaatkan banyak fitur konektivitas, keamanan, dan observabilitas lintas komputasi lainnya di VPC Lattice seperti menghubungkan ke layanan di seluruh cluster,, dan akun dengan, integrasi IAM untuk otorisasi dan otentikasi VPCs, dan fitur manajemen AWS Resource Access Manager lalu lintas lanjutan.

Pelanggan Amazon ECS dapat memanfaatkan VPC Lattice dengan cara berikut.

- Peningkatan produktivitas pengembang - VPC Lattice meningkatkan produktivitas pengembang dengan membiarkan Anda fokus pada membangun fitur, sementara VPC Lattice menangani tantangan jaringan, keamanan, dan pengamatan secara seragam di semua platform komputasi.
- Postur keamanan yang lebih baik - VPC Lattice memungkinkan pengembang Anda untuk dengan mudah mengautentikasi dan mengamankan komunikasi di seluruh aplikasi dan platform komputasi, menegakkan enkripsi dalam perjalanan, dan menerapkan kontrol akses granular dengan kebijakan VPC Lattice Auth. Ini memungkinkan Anda untuk mengadopsi postur keamanan yang lebih kuat yang memenuhi persyaratan peraturan dan kepatuhan terkemuka di industri.
- Peningkatan skalabilitas dan ketahanan aplikasi - VPC Lattice memungkinkan Anda membuat jaringan aplikasi yang digunakan dengan fitur seperti path, header, dan routing berbasis metode, otentikasi, otorisasi, dan pemantauan. Manfaat ini diberikan tanpa overhead sumber daya pada beban kerja dan dapat mendukung penerapan multi-cluster yang menghasilkan jutaan permintaan per detik tanpa menambahkan latensi yang signifikan.
- Fleksibilitas penerapan dengan infrastruktur heterogen - VPC Lattice menyediakan fitur yang konsisten di semua layanan komputasi seperti Amazon ECS, Fargate, Amazon, Amazon EKS, dan Lambda dan memungkinkan organisasi Anda fleksibilitas untuk memilih infrastruktur yang sesuai untuk setiap aplikasi. EC2

Bagaimana VPC Lattice bekerja dengan layanan Amazon ECS lainnya

Menggunakan VPC Lattice dengan Amazon ECS dapat mengubah cara Anda menggunakan layanan Amazon ECS lainnya, sementara yang lain tetap sama.

Application Load Balancer

Anda tidak perlu lagi membuat Application Load Balancer tertentu untuk digunakan dengan tipe grup target Application Load Balancer di VPC Lattice yang kemudian ditautkan ke layanan Amazon ECS. Anda hanya perlu mengonfigurasi layanan Amazon ECS Anda dengan grup target VPC Lattice sebagai gantinya. Anda juga dapat memilih untuk menggunakan Application Load Balancer dengan Amazon ECS secara bersamaan.

Penerapan bergulir Amazon ECS

Hanya penerapan bergulir Amazon ECS yang berfungsi dengan VPC Lattice, dan Amazon ECS dengan aman membawa tugas ke dalam dan menghapusnya dari layanan selama penerapan. Blue/Green Penyebaran dan penerapan kode tidak didukung.

Untuk mempelajari selengkapnya tentang VPC Lattice, lihat Panduan Pengguna Amazon VPC Lattice.

Buat layanan yang menggunakan VPC Lattice

Anda dapat menggunakan salah satu AWS Management Console atau AWS CLI untuk membuat layanan dengan VPC Lattice.

Prasyarat

Sebelum Anda memulai tutorial ini, pastikan bahwa prasyarat berikut terpenuhi:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, silakan lihat Menginstal AWS Command Line Interface.

  > **Note**
  >
  > Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

- Langkah-langkah yang dijelaskan di dalamnya Siapkan untuk menggunakan Amazon ECS sudah lengkap.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.

Buat layanan yang menggunakan VPC Lattice dengan AWS Management Console

Ikuti langkah-langkah ini untuk membuat layanan dengan VPC Lattice menggunakan. AWS Management Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di halaman navigasi, pilih Cluster.
3. Pada halaman Clusters, pilih cluster untuk membuat layanan di.
4. Dari tab Layanan, pilih Buat.

   Jika Anda belum pernah membuat layanan sebelumnya, ikuti langkah-langkah yang ditemukan di Membuat layanan Amazon ECS menggunakan konsol, lalu lanjutkan dengan langkah-langkah ini saat Anda mencapai bagian Kisi VPC.
5. Pilih untuk Menghidupkan Kisi VPC dengan mencentang tombol.
6. Untuk menggunakan peran yang ada, untuk peran infrastruktur ECS untuk Amazon ECS, pilih salah satu yang telah Anda buat untuk digunakan saat membuat grup target VPC Lattice. Untuk membuat peran baru, Buat peran infrastruktur ECS.
7. Pilih VPC.

   VPC tergantung pada mode jaringan yang Anda pilih saat Anda mendaftarkan definisi tugas Anda. Jika Anda menggunakan `network` mode `host` or dengan tipe EC2 peluncuran, pilih VPC Anda.

   Untuk `awsvpc` mode, VPC dipilih secara otomatis berdasarkan VPC yang Anda pilih di bawah Jaringan dan tidak dapat diubah.
8. Di bawah Grup Target pilih grup target atau grup. Anda harus memilih setidaknya satu kelompok sasaran dan dapat memiliki maksimal lima. Pilih Tambahkan grup target untuk menambahkan grup target tambahan. Pilih nama Port, Protokol, dan Port untuk setiap grup target yang Anda pilih. Untuk menghapus grup target, pilih Hapus.

> **Note**
>
> - Jika Anda ingin menambahkan grup target yang ada, Anda perlu menggunakan AWS CLI. Untuk petunjuk tentang cara menambahkan grup target menggunakan AWS CLI, lihat daftar target di Referensi. AWS Command Line Interface

- Meskipun layanan VPC Lattice dapat memiliki beberapa grup target, setiap grup target hanya dapat ditambahkan ke satu layanan.

9. Pada titik ini, Anda menavigasi ke konsol VPC Lattice untuk melanjutkan pengaturan. Di sinilah Anda menyertakan grup target baru dalam tindakan default listener atau dalam aturan layanan VPC Lattice yang ada.

   Untuk informasi selengkapnya, lihat Aturan pendengar untuk layanan Kisi VPC Anda.

> **Important**
>
> Anda harus mengizinkan `vpc-lattice` awalan aturan masuk ke grup keamanan atau tugas Anda dan pemeriksaan kesehatan dapat gagal.

Buat layanan yang menggunakan VPC Lattice dengan AWS CLI

Gunakan AWS CLI untuk membuat layanan dengan VPC Lattice. Ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

1. Buat file konfigurasi grup target. Contoh berikut diberi nama `tg-config.json`

```
{
    "ipAddressType": "IPV4",
    "port": 443,
    "protocol": "HTTPS",
    "protocolVersion": "HTTP1",
    "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
}
```

2. Gunakan perintah berikut untuk membuat grup target VPC Lattice.

```
aws vpc-lattice create-target-group \
    --name my-lattice-target-group-ip \
    --type IP \
    --config file://tg-config.json
```

   Contoh output:

```
{
    "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
    "config": {
        "healthCheck": {
            "enabled": true,
            "healthCheckIntervalSeconds": 30,
            "healthCheckTimeoutSeconds": 5,
            "healthyThresholdCount": 5,
            "matcher": {
                "httpCode": "200"
            },
            "path": "/",
            "protocol": "HTTPS",
            "protocolVersion": "HTTP1",
            "unhealthyThresholdCount": 2
        },
        "ipAddressType": "IPV4",
        "port": 443,
        "protocol": "HTTPS",
        "protocolVersion": "HTTP1",
        "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
    },
    "id": "tg-0eaa4b9ab4EXAMPLE",
    "name": "my-lattice-target-group-ip",
    "status": "CREATE_IN_PROGRESS",
    "type": "IP"
}
```

3. File JSON berikut bernama *ecs-service-vpc-lattice.json* adalah contoh yang digunakan untuk melampirkan layanan Amazon ECS ke grup target VPC Lattice. Contoh `portName` di bawah ini sama dengan yang Anda tentukan di `name` bidang `portMappings` properti definisi tugas Anda.

```
{
    "serviceName": "ecs-service-vpc-lattice",
    "taskDefinition": "ecs-task-def",
        "vpcLatticeConfigurations": [
        {
            "targetGroupArn": "arn:aws:vpc-lattice:us-
west-2:123456789012:targetgroup/tg-0eaa4b9ab4EXAMPLE",
            "portName": "testvpclattice",
```

```
            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRoleVpcLattice"
        }
    ],
    "desiredCount": 5,
    "role": "ecsServiceRole"
}
```

Gunakan perintah berikut untuk membuat layanan Amazon ECS dan melampirkannya ke grup target VPC Lattice menggunakan contoh json di atas.

```
aws ecs create-service \
    --cluster clusterName \
    --serviceName ecs-service-vpc-lattice \
    --cli-input-json file://ecs-service-vpc-lattice.json
```

> **Note**
>
> VPC Lattice tidak didukung di Amazon ECS Anywhere.

Lindungi tugas Amazon ECS Anda agar tidak dihentikan oleh peristiwa penskalaan

Anda dapat menggunakan perlindungan penskalaan tugas Amazon ECS untuk melindungi tugas agar tidak dihentikan oleh peristiwa penskalaan dari penskalaan otomatis layanan atau penerapan.

Aplikasi tertentu memerlukan mekanisme untuk melindungi tugas-tugas penting misi dari penghentian oleh peristiwa penskalaan selama masa pemanfaatan rendah atau selama penerapan layanan. Misalnya:

- Anda memiliki aplikasi asinkron pemrosesan antrian seperti pekerjaan transcoding video di mana beberapa tugas perlu dijalankan selama berjam-jam bahkan ketika pemanfaatan layanan kumulatif rendah.
- Anda memiliki aplikasi game yang menjalankan server game sebagai tugas Amazon ECS yang perlu terus berjalan bahkan jika semua pengguna telah logged-out untuk mengurangi latensi start-up dari reboot server.

- Saat Anda menerapkan versi kode baru, Anda memerlukan tugas untuk terus berjalan karena akan mahal untuk diproses ulang.

Untuk melindungi tugas yang termasuk dalam layanan Anda agar tidak dihentikan dalam peristiwa scale-in, setel atribut ke`ProtectionEnabled`. `true` Bila Anda menyetel `ProtectionEnabled` ke true, tugas dilindungi selama 2 jam secara default. Anda kemudian dapat menyesuaikan periode perlindungan dengan menggunakan `ExpiresInMinutes` atribut. Anda dapat melindungi tugas Anda selama minimal 1 menit dan hingga maksimum 2880 menit (48 jam). Jika Anda menggunakan AWS CLI, Anda dapat menentukan `--protection-enabled` opsi.

Setelah tugas menyelesaikan pekerjaan yang dipersyaratkan, Anda dapat mengatur `ProtectionEnabled` atribut ke`false`, memungkinkan tugas dihentikan oleh peristiwa skala berikutnya. Jika Anda menggunakan AWS CLI, Anda dapat menentukan `--no-protection-enabled` opsi.

Mekanisme perlindungan skala tugas

Anda dapat mengatur dan mendapatkan perlindungan skala tugas menggunakan titik akhir agen penampung Amazon ECS atau Amazon ECS API.

- Titik akhir agen kontainer Amazon ECS

  Sebaiknya gunakan titik akhir agen penampung Amazon ECS untuk tugas yang dapat menentukan sendiri kebutuhan untuk dilindungi. Gunakan pendekatan ini untuk beban kerja berbasis antrian atau pemrosesan pekerjaan.

  Saat wadah mulai memproses pekerjaan, misalnya dengan menggunakan pesan SQS, Anda dapat mengatur `ProtectionEnabled` atribut melalui jalur titik akhir perlindungan skala tugas `$ECS_AGENT_URI/task-protection/v1/state` dari dalam wadah. Amazon ECS tidak akan menghentikan tugas ini selama acara scale-in. Setelah tugas Anda selesai, Anda dapat menghapus `ProtectionEnabled` atribut menggunakan titik akhir yang sama, membuat tugas memenuhi syarat untuk penghentian selama peristiwa penskalaan berikutnya.

  Untuk informasi selengkapnya tentang titik akhir agen penampung Amazon ECS, lihat. Titik akhir perlindungan skala tugas Amazon ECS
- Amazon ECS API

  Anda dapat menggunakan Amazon ECS API untuk menyetel dan mengambil perlindungan skala tugas jika aplikasi Anda memiliki komponen yang melacak status tugas aktif. Gunakan

`UpdateTaskProtection` untuk menandai satu atau lebih tugas sebagai dilindungi. Gunakan `GetTaskProtection` untuk mengambil status perlindungan.

Contoh dari pendekatan ini adalah jika aplikasi Anda menghosting sesi server game sebagai tugas Amazon ECS. Saat pengguna masuk ke sesi di server (tugas), Anda dapat menandai tugas sebagai dilindungi. Setelah pengguna log out, Anda dapat menghapus perlindungan khusus untuk tugas ini atau secara berkala menghapus perlindungan untuk tugas serupa yang tidak lagi memiliki sesi aktif, tergantung pada kebutuhan Anda untuk menjaga server menganggur.

Untuk informasi selengkapnya, lihat UpdateTaskProtectiondan GetTaskProtectiondi Referensi API Amazon Elastic Container Service.

Anda dapat menggabungkan kedua pendekatan. Misalnya, gunakan titik akhir agen Amazon ECS untuk mengatur perlindungan tugas dari dalam wadah dan gunakan Amazon ECS API untuk menghapus perlindungan tugas dari layanan pengontrol eksternal Anda.

Pertimbangan

Pertimbangkan poin-poin berikut sebelum menggunakan perlindungan skala tugas:

- Perlindungan skala tugas hanya didukung dengan tugas yang diterapkan dari layanan.
- Sebaiknya gunakan titik akhir agen penampung Amazon ECS karena agen Amazon ECS memiliki mekanisme coba ulang bawaan dan antarmuka yang lebih sederhana.
- Anda dapat mengatur ulang periode kedaluwarsa perlindungan skala tugas dengan memanggil tugas `UpdateTaskProtection` yang telah mengaktifkan perlindungan.
- Tentukan berapa lama tugas yang dibutuhkan untuk menyelesaikan pekerjaan yang diperlukan dan mengatur `expiresInMinutes` properti yang sesuai. Jika Anda mengatur kedaluwarsa perlindungan lebih lama dari yang diperlukan, maka Anda akan dikenakan biaya dan menghadapi keterlambatan dalam penyebaran tugas baru.
- Perlindungan skala tugas didukung pada agen `1.65.0` kontainer Amazon ECS atau yang lebih baru.

  Anda dapat menambahkan dukungan untuk fitur ini di EC2 instans Amazon menggunakan versi lama agen penampung Amazon ECS dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.
- Pertimbangan penyebaran:

- Jika layanan menggunakan pembaruan bergulir, tugas baru akan dibuat tetapi tugas yang menjalankan versi lama tidak akan dihentikan sampai dihapus atau `protectionEnabled` kedaluwarsa. Anda dapat menyesuaikan `maximumPercentage` parameter dalam konfigurasi penerapan ke nilai yang memungkinkan tugas baru dibuat saat tugas lama dilindungi.
- Jika blue/green pembaruan diterapkan, penerapan biru dengan tugas yang dilindungi tidak akan dihapus jika tugas memilikiprotectionEnabled. Lalu lintas akan dialihkan ke tugas baru yang muncul dan tugas lama hanya akan dihapus ketika dihapus atau `protectionEnabled` kedaluwarsa. Bergantung pada batas waktu CodeDeploy atau CloudFormation pembaruan, penerapan mungkin batas waktu dan tugas Biru yang lebih lama mungkin masih ada.
- Jika Anda menggunakan CloudFormation, update-stack memiliki batas waktu 3 jam. Oleh karena itu, jika Anda mengatur perlindungan tugas Anda lebih dari 3 jam, maka CloudFormation penerapan Anda dapat mengakibatkan kegagalan dan kemunduran.

  Selama tugas lama Anda dilindungi, CloudFormation tumpukan ditampilkanUPDATE_IN_PROGRESS. Jika perlindungan skala tugas dihapus atau kedaluwarsa dalam jendela 3 jam, penerapan Anda akan berhasil dan pindah ke status. `UPDATE_COMPLETE` Jika penerapan macet `UPDATE_IN_PROGRESS` selama lebih dari 3 jam, itu akan gagal dan menunjukkan `UPDATE_FAILED` status, dan kemudian akan digulung kembali ke set tugas lama.
- Amazon ECS mengirimkan peristiwa layanan saat tugas yang dilindungi menjaga penerapan (bergulir atau biru/hijau) agar tidak mencapai kondisi tunak, sehingga Anda dapat mengambil tindakan perbaikan. Saat mencoba memperbarui status perlindungan suatu tugas, jika Anda menerima pesan `DEPLOYMENT_BLOCKED` kesalahan, itu berarti layanan memiliki tugas yang lebih terlindungi daripada jumlah tugas yang diinginkan untuk layanan. Untuk mengatasi kesalahan ini, lakukan salah satu hal berikut:
  - Tunggu hingga perlindungan tugas saat ini kedaluwarsa. Kemudian atur perlindungan tugas.
  - Tentukan tugas mana yang bisa dihentikan. Kemudian gunakan `UpdateTaskProtection` dengan `protectionEnabled` opsi yang disetel `false` untuk tugas-tugas ini.
  - Tingkatkan jumlah tugas yang diinginkan dari layanan menjadi lebih dari jumlah tugas yang dilindungi.

Izin IAM diperlukan untuk perlindungan skala tugas

Tugas harus memiliki peran tugas Amazon ECS dengan izin berikut:

- `ecs:GetTaskProtection`: Memungkinkan agen kontainer Amazon ECS untuk meneleponGetTaskProtection.

- `ecs:UpdateTaskProtection`: Memungkinkan agen kontainer Amazon ECS untuk meneleponUpdateTaskProtection.

Titik akhir perlindungan skala tugas Amazon ECS

Agen penampung Amazon ECS secara otomatis menyuntikkan variabel `ECS_AGENT_URI` lingkungan ke dalam wadah tugas Amazon ECS untuk menyediakan metode untuk berinteraksi dengan titik akhir API agen penampung.

Sebaiknya gunakan titik akhir agen penampung Amazon ECS untuk tugas yang dapat menentukan sendiri kebutuhan untuk dilindungi.

Saat wadah mulai memproses pekerjaan, Anda dapat mengatur `protectionEnabled` atribut menggunakan jalur titik akhir perlindungan skala tugas `$ECS_AGENT_URI/task-protection/v1/state` dari dalam wadah.

Gunakan permintaan PUT ke URI ini dari dalam wadah untuk mengatur perlindungan skala tugas. Permintaan GET ke URI ini mengembalikan status perlindungan tugas saat ini.

Parameter permintaan perlindungan skala tugas

Anda dapat mengatur perlindungan skala tugas menggunakan `${ECS_AGENT_URI}/task-protection/v1/state` titik akhir dengan parameter permintaan berikut.

`ProtectionEnabled`

Tentukan `true` untuk menandai tugas untuk perlindungan. Tentukan `false` untuk menghapus perlindungan dan membuat tugas memenuhi syarat untuk penghentian.

Jenis: Boolean

Wajib: Ya

`ExpiresInMinutes`

Jumlah menit tugas dilindungi. Anda dapat menentukan minimal 1 menit hingga 2.880 menit (48 jam). Selama periode waktu ini, tugas Anda tidak akan dihentikan oleh peristiwa penskalaan dari Auto Scaling atau penerapan layanan. Setelah periode waktu ini berlalu, `protectionEnabled` parameter diatur ke`false`.

Jika Anda tidak menentukan waktu, maka tugas secara otomatis dilindungi selama 120 menit (2 jam).

Tipe: Integer

Wajib: Tidak

Contoh berikut menunjukkan cara mengatur perlindungan tugas dengan durasi yang berbeda.

Contoh cara melindungi tugas dengan periode waktu default

Contoh ini menunjukkan cara melindungi tugas dengan periode waktu default 2 jam.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true}'
```

Contoh cara melindungi tugas selama 60 menit

Contoh ini menunjukkan cara melindungi tugas selama 60 menit menggunakan `expiresInMinutes` parameter.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":60}'
```

Contoh cara melindungi tugas selama 24 jam

Contoh ini menunjukkan cara melindungi tugas selama 24 jam menggunakan `expiresInMinutes` parameter.

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}'
```

Contoh untuk wadah Windows

Untuk wadah Windows, Anda dapat menggunakan PowerShell `Invoke-RestMethod` cmdlet alih-alih curl. Contoh berikut menunjukkan PowerShell ekuivalen dari perintah curl sebelumnya.

Contoh cara melindungi tugas penampung Windows dengan periode waktu default

Contoh ini menunjukkan cara melindungi tugas dengan periode waktu default 2 jam menggunakan PowerShell.

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body
 '{"ProtectionEnabled":true}' -ContentType 'application/json'
```

Contoh cara melindungi tugas penampung Windows selama 60 menit

Contoh ini menunjukkan cara melindungi tugas selama 60 menit menggunakan `expiresInMinutes` parameter dengan PowerShell.

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body
 '{"ProtectionEnabled":true,"ExpiresInMinutes":60}' -ContentType 'application/json'
```

Contoh cara melindungi tugas penampung Windows selama 24 jam

Contoh ini menunjukkan cara melindungi tugas selama 24 jam menggunakan `expiresInMinutes` parameter dengan PowerShell.

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body
 '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}' -ContentType 'application/json'
```

Permintaan PUT mengembalikan respon berikut.

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

Parameter respons perlindungan skala tugas

Informasi berikut dikembalikan dari titik akhir perlindungan skala tugas dalam respons `${ECS_AGENT_URI}/task-protection/v1/state` JSON.

`ExpirationDate`

Waktu zaman ketika perlindungan untuk tugas akan kedaluwarsa. Jika tugas tidak dilindungi, nilai ini adalah nol.

`ProtectionEnabled`

Status perlindungan tugas. Jika perlindungan scale-in diaktifkan untuk tugas, nilainya adalah. `true` Kalau tidak, itu`false`.

`TaskArn`

Nama lengkap Amazon Resource Name (ARN) dari tugas milik kontainer.

Contoh berikut menunjukkan detail yang dikembalikan untuk tugas yang dilindungi.

```
curl --request GET ${ECS_AGENT_URI}/task-protection/v1/state
```

Untuk wadah Windows, gunakan PowerShell perintah berikut untuk mendapatkan status perlindungan:

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Get
```

```
{
    "protection":{
        "ExpirationDate":"2023-12-20T21:57:44Z",
        "ProtectionEnabled":true,
        "TaskArn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
    }
}
```

Informasi berikut dikembalikan ketika terjadi kegagalan.

`Arn`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas tersebut.

`Detail`

Detail terkait dengan kegagalan.

`Reason`

Sebab kegagalan.

Contoh berikut menunjukkan rincian yang dikembalikan untuk tugas yang tidak dilindungi.

```
{
    "failure":{
        "Arn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0",
```

```
        "Detail":null,
        "Reason":"TASK_NOT_VALID"
    }
}
```

Informasi berikut dikembalikan ketika pengecualian terjadi.

`requestID`

ID AWS permintaan untuk panggilan Amazon ECS API yang menghasilkan pengecualian.

`Arn`

Nama Sumber Daya Amazon (ARN) lengkap dari tugas atau layanan.

`Code`

Kode kesalahan.

`Message`

Pesan kesalahan.

> **Note**
>
> Jika `RequestTimeout` kesalahan `RequestError` atau muncul, kemungkinan itu adalah masalah jaringan. Coba gunakan titik akhir VPC untuk Amazon ECS.

Contoh berikut menunjukkan rincian yang dikembalikan ketika terjadi kesalahan.

```
{
    "requestID":"12345-abc-6789-0123-abc",
    "error":{
        "Arn":"arn:aws:ecs:us-west-2:555555555555:task/my-cluster-
name/1234567890abcdef0",
        "Code":"AccessDeniedException",
        "Message":"User: arn:aws:sts::444455556666:assumed-role/my-ecs-task-
role/1234567890abcdef0 is not authorized to perform: ecs:GetTaskProtection on resource:
 arn:aws:ecs:us-west-2:555555555555:task/test/1234567890abcdef0 because no identity-
based policy allows the ecs:GetTaskProtection action"
    }
}
```

Kesalahan berikut muncul jika agen Amazon ECS tidak dapat mendapatkan respons dari titik akhir Amazon ECS karena alasan seperti masalah jaringan atau bidang kontrol Amazon ECS sedang down.

```
{
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "RequestCanceled",
    "Message": "Timed out calling Amazon ECS Task Protection API"
  }
}
```

Kesalahan berikut muncul ketika agen Amazon ECS mendapatkan pengecualian pelambatan dari Amazon ECS.

```
{
  "requestID": "12345-abc-6789-0123-abc",
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "ThrottlingException",
    "Message": "Rate exceeded"
  }
}
```

Gunakan injeksi kesalahan dengan beban kerja Amazon ECS dan Fargate Anda

Pelanggan dapat menggunakan injeksi kesalahan dengan Amazon ECS di Amazon dan EC2 Fargate untuk menguji bagaimana aplikasi mereka merespons skenario gangguan tertentu. Pengujian ini memberikan informasi yang dapat Anda gunakan untuk mengoptimalkan kinerja dan ketahanan aplikasi Anda.

Saat injeksi kesalahan diaktifkan, agen penampung Amazon ECS memungkinkan akses tugas ke titik akhir injeksi kesalahan baru. Anda perlu ikut serta untuk menggunakan injeksi kesalahan dengan menyetel nilai parameter definisi `enableFaultInjection` tugas ke`true`. Nilai default-nya adalah `false`.

```
{
    ...
```

```
    "enableFaultInjection": true
}
```

> **Note**
>
> Injeksi kesalahan hanya berfungsi dengan tugas menggunakan mode `host` jaringan `awsvpc` atau.
> Injeksi kesalahan tidak tersedia di Windows.

Untuk informasi tentang cara mengaktifkan injeksi kesalahan di AWS Management Console, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Anda harus mengaktifkan fitur untuk pengujian di file AWS Fault Injection Service. Untuk informasi selengkapnya, lihat Menggunakan tindakan AWS FIS aws:ecs:task.

> **Note**
>
> Jika Anda tidak menggunakan Amazon ECS yang dioptimalkan AMIs, atau memiliki AMI khusus, instal dependensi berikut:
>
> - `tc`
> - `sch_netem`modul kernel

Titik akhir injeksi kesalahan Amazon ECS

Agen penampung Amazon ECS secara otomatis menyuntikkan variabel `ECS_AGENT_URI` lingkungan ke dalam wadah tugas Amazon ECS untuk menyediakan metode untuk berinteraksi dengan titik akhir API agen penampung. Setiap titik akhir mencakup`/start`,`/stop`, dan titik `/status` akhir. Titik akhir hanya menerima permintaan dari tugas yang telah mengaktifkan injeksi kesalahan, dan setiap titik akhir memiliki batas laju 1 permintaan per 5 detik per kontainer. Melebihi batas ini menghasilkan kesalahan.

> **Note**
>
> Agen Amazon ECS `version 1.88.0+` diperlukan untuk menggunakan titik akhir injeksi kesalahan.

Tiga titik akhir untuk digunakan dengan injeksi kesalahan adalah:

- Titik akhir port blackhole jaringan
- Titik akhir kehilangan paket jaringan
- Titik akhir latensi jaringan

Permintaan yang berhasil menghasilkan kode respons `200` dengan pesan `running` saat Anda memanggil `/start` titik akhir, `stopped` untuk titik `/stop` akhir, dan `running` atau `not-running` untuk titik akhir. `/status`

```
{
    "Status": <string>
}
```

Permintaan yang gagal mengembalikan salah satu kode kesalahan berikut:

- `400`- Permintaan buruk
- `409`- Permintaan injeksi kesalahan bertentangan dengan kesalahan berjalan lainnya
- `429`- Permintaan dibatasi
- `500`- Server memiliki kesalahan yang tidak terduga

```
{
 "Error":  <string message>
}
```

> **Note**
>
> Salah satu kesalahan latensi jaringan atau satu kesalahan kehilangan paket jaringan dapat disuntikkan pada suatu waktu. Mencoba menyuntikkan lebih dari satu mengakibatkan permintaan ditolak.

Titik akhir port blackhole jaringan

`{ECS_AGENT_URI}/fault/v1/network-blackhole-port`Titik akhir menurunkan lalu lintas masuk atau keluar untuk port dan protokol tertentu dalam namespace jaringan tugas dan kompatibel dengan dua mode:

- awsvpc - perubahan diterapkan ke namespace jaringan tugas
- host - perubahan diterapkan ke instance kontainer namespace jaringan default

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/start

Titik akhir ini memulai injeksi kesalahan port blackhole jaringan dan memiliki parameter berikut:

Port

Port yang ditentukan untuk digunakan untuk injeksi kesalahan port lubang hitam.

Jenis: Integer

Wajib: Ya

Protokol

Protokol yang digunakan untuk injeksi kesalahan port lubang hitam.

Tipe: String

Nilai yang valid: `tcp | udp`

Wajib: Ya

TrafficType

Jenis lalu lintas yang digunakan oleh injeksi kesalahan.

Tipe: String

Nilai yang valid: `ingress | egress`

Wajib: Ya

SourcesToFilter

Sebuah array JSON IPv4 alamat atau blok CIDR yang dilindungi dari kesalahan.

Tipe: Array string

Wajib: Tidak

Berikut ini adalah contoh permintaan untuk menggunakan `start` titik akhir (ganti *red* nilai dengan milik Anda sendiri):

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/start

Http method:POST

Request payload:
{
    "Port": 1234,
    "Protocol": "tcp|udp",
    "TrafficType": "ingress|egress"
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

Titik akhir ini menghentikan kesalahan yang ditentukan dalam permintaan. Titik akhir ini memiliki parameter berikut:

Port

Port terkena dampak kesalahan yang harus dihentikan.

Jenis: Integer

Wajib: Ya

Protokol

Protokol yang digunakan untuk menghentikan kesalahan.

Tipe: String

Nilai yang valid: `tcp | udp`

Wajib: Ya

TrafficType

Jenis lalu lintas yang digunakan oleh injeksi kesalahan.

Tipe: String

Nilai yang valid: `ingress | egress`

Wajib: Ya

Berikut ini adalah contoh permintaan untuk menggunakan `stop` titik akhir (ganti *red* nilai dengan milik Anda sendiri):

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

Http method: POST

Request payload:
{
    "Port": 1234,
    "Protocol": "tcp|udp",
    "TrafficType": "ingress|egress",
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/status

Endpoint ini digunakan untuk memeriksa status injeksi kesalahan. Titik akhir ini memiliki parameter berikut:

Port

Port yang terkena dampak untuk memeriksa status kesalahan.

Jenis: Integer

Wajib: Ya

Protokol

Protokol yang digunakan saat memeriksa status kesalahan.

Tipe: String

Nilai yang valid: `tcp | udp`

Wajib: Ya

TrafficType

Jenis lalu lintas yang digunakan oleh injeksi kesalahan.

Tipe: String

Nilai yang valid: `ingress | egress`

Wajib: Ya

Berikut ini adalah contoh permintaan untuk menggunakan `status` titik akhir (ganti *red* nilai dengan milik Anda sendiri):

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/status

Http method: POST

Request payload:
{
   "Port": 1234,
   "Protocol": "tcp|udp",
   "TrafficType": "ingress|egress",
}
```

Titik akhir latensi jaringan

`{ECS_AGENT_URI}/fault/v1/network-latency`Titik akhir menambahkan penundaan dan jitter ke antarmuka jaringan tugas untuk lalu lintas ke sumber tertentu. Titik akhir kompatibel dengan dua mode:

- awsvpc - perubahan diterapkan ke antarmuka jaringan tugas
- host - perubahan diterapkan ke antarmuka jaringan default

{ECS_AGENT_URI}/fault/v1/network-latency/start

`/start`Titik akhir ini memulai injeksi kesalahan latensi jaringan dan memiliki parameter berikut:

DelayMilliseconds

Jumlah milidetik penundaan untuk ditambahkan ke antarmuka jaringan yang akan digunakan untuk injeksi kesalahan.

Jenis: Integer

Wajib: Ya

JitterMilliseconds

Jumlah milidetik jitter untuk ditambahkan ke antarmuka jaringan yang akan digunakan untuk injeksi kesalahan.

Jenis: Integer

Wajib: Ya

Sumber

Sebuah array JSON IPv4 alamat atau blok CIDR yang tujuan untuk digunakan dengan injeksi kesalahan.

Tipe: Array string

Wajib: Ya

SourcesToFilter

Sebuah array JSON IPv4 alamat atau blok CIDR yang dilindungi dari kesalahan. `SourcesToFilter`mengambil prioritas di atas`Sources`.

Tipe: Array string

Wajib: Tidak

Berikut ini adalah contoh permintaan untuk menggunakan `/start` titik akhir (ganti *red* nilai dengan milik Anda sendiri):

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/start

Http method: POST

Request payload:
{
    "DelayMilliseconds": 123,
    "JitterMilliseconds": 123,
    "Sources": ["${IP1}", "${IP2}", ...],
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-latency/stop and /status

`{ECS_AGENT_URI}/fault/v1/network-latency/stop`Titik akhir menghentikan kesalahan, dan `{ECS_AGENT_URI}/fault/v1/network-latency/status` memeriksa status kesalahan.

Berikut ini adalah dua contoh permintaan untuk menggunakan `/stop` dan `/status` endpoint. Keduanya menggunakan `POST` HTTP metode ini.

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/stop
```

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/status
```

Titik akhir kehilangan paket jaringan

`{ECS_AGENT_URI}/fault/v1/network-packet-loss`Endpoint menambahkan packet loss ke antarmuka jaringan yang diberikan. Endpoint ini kompatibel dengan dua mode:

- awsvpc - perubahan diterapkan ke antarmuka jaringan tugas
- host - perubahan diterapkan ke antarmuka jaringan default

{ECS_AGENT_URI}/fault/v1/network-packet-loss/start

`/start`Titik akhir ini memulai injeksi kesalahan kehilangan paket jaringan dan memiliki parameter berikut:

LossPercent

Persentase kehilangan paket

Jenis: Integer

Wajib: Ya

Sumber

Sebuah array JSON IPv4 alamat atau blok CIDR untuk digunakan untuk tes injeksi kesalahan.

Tipe: Array string

Wajib: Ya

SourcesToFilter

Sebuah array JSON IPv4 alamat atau blok CIDR yang dilindungi dari kesalahan. `SourcesToFilter`mengambil prioritas di atas`Sources`.

Tipe: Array string

Wajib: Tidak

Berikut ini adalah contoh permintaan untuk menggunakan `start` titik akhir (ganti *`red`* nilai dengan milik Anda sendiri):

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/start

Http method: POST

{
    "LossPercent": 6,
    "Sources": ["${IP1}", "${IP2}", ...],
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop and /status

`{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop`Titik akhir menghentikan kesalahan, dan `{ECS_AGENT_URI}/fault/v1/network-packet-loss/status` memeriksa status kesalahan. Hanya satu dari setiap jenis kesalahan yang didukung pada satu waktu.

Berikut ini adalah dua contoh permintaan untuk menggunakan `/stop` dan `/status` endpoint. Keduanya menggunakan `POST` `HTTP` metode ini.

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/stop
```

```
Endpoint: ${{ECS_AGENT_URI}/fault/v1/network-packet-loss/status
```

Perbarui parameter layanan Amazon ECS

Setelah Anda membuat layanan, ada kalanya Anda mungkin perlu memperbarui parameter layanan, misalnya, jumlah tugas.

Ketika penjadwal layanan meluncurkan tugas baru, itu menentukan penempatan tugas di klaster Anda dengan logika berikut.

- Tentukan instance kontainer mana di klaster Anda yang dapat mendukung definisi tugas layanan Anda. Misalnya, mereka memiliki atribut CPU, memori, port, dan instance container yang diperlukan.
- Secara default, penjadwal layanan mencoba menyeimbangkan tugas di seluruh Availability Zone dengan cara ini meskipun Anda dapat memilih strategi penempatan yang berbeda.
  - Urutkan instance kontainer yang valid dengan jumlah tugas yang berjalan paling sedikit untuk layanan ini di Availability Zone yang sama dengan instance. Misalnya, jika zona A memiliki satu tugas layanan berjalan, sedangkan zona B dan C masing-masing tidak memliki tugas layanan berjalan, maka instans kontainer yang valid, baik di zona B ataupun C dianggap optimal untuk penempatan.
  - Tempatkan tugas layanan baru pada instans kontainer yang valid di Availability Zone yang optimal (berdasarkan langkah sebelumnya), dan pilih instans kontainer dengan jumlah tugas yang berjalan paling sedikit untuk layanan ini.

Ketika penjadwal layanan berhenti menjalankan tugas, ia mencoba untuk menjaga keseimbangan di seluruh Availability Zone di klaster Anda menggunakan logika berikut:

- Urutkan instance kontainer berdasarkan jumlah tugas yang berjalan terbesar untuk layanan ini di Availability Zone yang sama dengan instance. Misalnya, jika zona A memiliki satu tugas layanan yang berjalan, dan zona B serta C masing-masing memiliki dua, instans kontainer di zona B atau C dianggap optimal untuk penghentian.
- Hentikan tugas pada instans kontainer di Availability Zone yang optimal (berdasarkan langkah sebelumnya), kemudian pilih instans kontainer dengan jumlah tugas yang berjalan paling banyak untuk layanan ini.

Gunakan daftar untuk menentukan apakah Anda dapat mengubah parameter layanan.

Penyeimbangan kembali Zona Ketersediaan

Menunjukkan apakah akan menggunakan penyeimbangan kembali Availability Zone untuk layanan.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Strategi penyedia kapasitas

Detail strategi penyedia kapasitas. Anda dapat mengatur penyedia kapasitas saat membuat klaster, menjalankan tugas, atau memperbarui layanan.

Ketika Anda menggunakan Fargate, penyedia kapasitas adalah `FARGATE` atau. `FARGATE_SPOT`

Saat Anda menggunakan Amazon EC2, penyedia kapasitas adalah grup Auto Scaling.

Anda dapat mengubah penyedia kapasitas untuk penerapan bergulir dan penerapan biru/hijau.

Daftar berikut menyediakan transisi yang valid:

- Perbarui jenis peluncuran Fargate ke penyedia kapasitas grup Auto Scaling.
- Perbarui jenis EC2 peluncuran Amazon ke penyedia kapasitas Fargate.
- Perbarui penyedia kapasitas Fargate ke penyedia kapasitas grup Auto Scaling.
- Perbarui penyedia EC2 kapasitas Amazon ke penyedia kapasitas Fargate.
- Perbarui grup Auto Scaling atau penyedia kapasitas Fargate kembali ke jenis peluncuran. Saat Anda menggunakan CLI, atau API, Anda meneruskan daftar kosong di parameter. `capacityProviderStrategy`

Kluster

Anda tidak dapat mengubah nama cluster.

Konfigurasi penyebaran

Konfigurasi penyebaran mencakup CloudWatch alarm, dan pemutus sirkuit yang digunakan untuk mendeteksi kegagalan dan konfigurasi yang diperlukan.

Pemutus sirkuit deployment menentukan apakah deployment layanan akan gagal jika layanan tidak dapat mencapai status stabil. Jika Anda menggunakan pemutus sirkuit penyebaran, penyebaran layanan akan bertransisi ke status gagal dan berhenti meluncurkan tugas baru. Jika Anda menggunakan opsi rollback, ketika penerapan layanan gagal, layanan akan digulirkan kembali ke penerapan terakhir yang berhasil diselesaikan.

Saat Anda memperbarui layanan yang menggunakan pemutus sirkuit Amazon ECS, Amazon ECS membuat penyebaran layanan dan revisi layanan. Sumber daya ini memungkinkan Anda untuk melihat informasi terperinci tentang riwayat layanan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

Penjadwal layanan menggunakan parameter persentase minimum dan maksimum yang sehat (dalam konfigurasi deployment untuk layanan) untuk menentukan strategi deployment.

Jika layanan menggunakan tipe penerapan rolling update (`ECS`), persentase sehat minimum mewakili batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan

ke bilangan bulat terdekat). Parameter ini juga berlaku saat instance kontainer apa pun berada dalam DRAINING status jika layanan berisi tugas menggunakan tipe EC2 peluncuran. Gunakan parameter ini untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan. Misalnya, jika layanan Anda memiliki jumlah empat tugas yang diinginkan dan persentase sehat minimum 50 persen, penjadwal dapat menghentikan dua tugas yang ada untuk membebaskan kapasitas cluster sebelum memulai dua tugas baru. Layanan menganggap tugas sehat untuk layanan yang tidak menggunakan penyeimbang beban jika mereka berada di RUNNING negara bagian. Layanan menganggap tugas sehat untuk layanan yang menggunakan penyeimbang beban jika mereka berada di RUNNING negara bagian dan dilaporkan sehat oleh penyeimbang beban. Nilai default untuk persen sehat minimum adalah 100 persen.

Jika layanan menggunakan tipe penerapan rolling update (ECS), parameter persen maksimum mewakili batas atas jumlah tugas dalam layanan yang diizinkan dalam,, atau STOPPING status selama penerapan PENDINGRUNNING, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Parameter ini juga berlaku saat instance kontainer apa pun berada dalam DRAINING status jika layanan berisi tugas menggunakan tipe EC2 peluncuran. Gunakan parameter ini untuk menentukan ukuran batch deployment. Misalnya, jika layanan Anda memiliki jumlah empat tugas yang diinginkan dan nilai persen maksimum 200 persen, penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas lama. Ini asalkan sumber daya cluster yang diperlukan untuk melakukan ini tersedia. Nilai default untuk persen maksimum adalah 200 persen.

Saat penjadwal layanan mengganti tugas selama pembaruan, maka layanan terlebih dahulu menghapus tugas dari penyeimbang beban (jika digunakan) dan menunggu koneksi dialihkan. Kemudian, docker stop yang setara dikeluarkan untuk kontainer yang berjalan dalam tugas. Hal ini menyebabkan sinyal SIGTERM dan waktu habis dalam 30 detik, setelah SIGKILL dikirim dan kontainer dihentikan secara paksa. Jika kontainer menangani sinyal SIGTERM dengan baik dan keluar dalam waktu 30 detik dari saat menerimanya, maka sinyal SIGKILL tidak dikirim. Penjadwal layanan memulai dan menghentikan tugas seperti yang ditentukan melalui pengaturan persentase minimum dan maksimum yang sehat.

Penjadwal layanan juga menggantikan tugas yang ditentukan tidak sehat setelah pemeriksaan kesehatan kontainer atau pemeriksaan kesehatan kelompok sasaran penyeimbang beban gagal. Penggantian ini tergantung pada parameter definisi maximumPercent dan desiredCount layanan. Jika tugas ditandai tidak sehat, penjadwal layanan akan memulai tugas pengganti terlebih dahulu. Kemudian, hal berikut terjadi.

- Jika tugas penggantian memiliki status kesehatanHEALTHY, penjadwal layanan menghentikan tugas yang tidak sehat

- Jika tugas penggantian memiliki status kesehatanUNHEALTHY, penjadwal akan menghentikan tugas penggantian yang tidak sehat atau tugas tidak sehat yang ada untuk mendapatkan jumlah tugas total yang samadesiredCount.

Jika maximumPercent parameter membatasi penjadwal untuk memulai tugas penggantian terlebih dahulu, penjadwal akan menghentikan tugas yang tidak sehat satu per satu secara acak untuk membebaskan kapasitas, dan kemudian memulai tugas pengganti. Proses start dan stop berlanjut sampai semua tugas yang tidak sehat diganti dengan tugas-tugas yang sehat. Setelah semua tugas yang tidak sehat telah diganti dan hanya tugas sehat yang berjalan, jika jumlah tugas total melebihidesiredCount, tugas sehat dihentikan secara acak hingga jumlah tugas total sama. desiredCount Untuk informasi selengkapnya tentang maximumPercent dandesiredCount, lihat Parameter definisi layanan.

Pengontrol penyebaran

Pengendali deployment yang bisa digunakan untuk layanan. Tersedia tiga tipe pengendali deployment:

- ECS
- EXTERNAL
- CODE_DEPLOY

Saat memperbarui layanan, Anda dapat memperbarui pengontrol penerapan yang digunakannya. Daftar berikut menyediakan transisi yang valid:

- Perbarui dari penerapan CodeDeploy biru/hijau (CODE_DEPLOY) ke ECS rolling atau deployments (). blue/green ECS
- Perbarui dari penerapan CodeDeploy biru/hijau (CODE_DEPLOY) ke penerapan eksternal (). EXTERNAL
- Pembaruan dari ECS rolling atau blue/green deployments (ECS) ke penerapan eksternal (). EXTERNAL
- Pembaruan dari penerapan eksternal (EXTERNAL) ke ECS rolling atau blue/green deployments (). ECS

Pertimbangkan hal berikut saat Anda memperbarui pengontrol penerapan layanan:

- Anda tidak dapat memperbarui pengontrol penyebaran layanan dari pengontrol ECS penerapan ke pengontrol lain jika menggunakan VPC Lattice atau Amazon ECS Service Connect.
- Anda tidak dapat memperbarui pengontrol penerapan layanan selama penerapan layanan yang sedang berlangsung.

- Anda tidak dapat memperbarui pengontrol penerapan layanan CODE_DEPLOY jika tidak ada penyeimbang beban pada layanan.
- Anda tidak dapat memperbarui pengontrol penyebaran layanan dari ECS ke pengontrol lain jika deploymentConfiguration menyertakan alarm, pemutus sirkuit penyebaran, atau strategi penerapan. BLUE_GREEN Untuk informasi selengkapnya, lihat Pengontrol dan strategi penyebaran layanan Amazon ECS.
- Nilai yang Anda tentukan versionConsistency dalam definisi kontainer tidak akan digunakan oleh Amazon ECS jika Anda memperbarui pengontrol penerapan layanan dari ECS pengontrol lainnya.
- Jika Anda memperbarui pengontrol penerapan layanan dari pengontrol lain ECS ke salah satu pengontrol lain, respons DescribeService API UpdateService dan akan tetap kembalideployments. taskSets Untuk informasi selengkapnya tentang UpdateService danCreateService, lihat UpdateServicedan CreateServicedi Referensi API Amazon ECS.
- Jika layanan menggunakan strategi penyebaran pembaruan bergulir, memperbarui pengontrol penerapan dari pengontrol lain ECS ke pengontrol lain akan mengubah cara maximumPercent nilai dalam digunakan. deploymentConfiguration Alih-alih hanya digunakan sebagai batas pada total tugas dalam penerapan pembaruan bergulir, maximumPercent digunakan untuk mengganti tugas yang tidak sehat. Untuk informasi selengkapnya tentang cara penjadwal menggantikan tugas yang tidak sehat, lihat. Layanan-layanan Amazon ECS
- Jika Anda memperbarui pengontrol penerapan layanan dari ECS pengontrol penerapan lainnya, semua advancedConfiguration yang Anda tentukan dengan konfigurasi penyeimbang beban Anda akan diabaikan. Untuk informasi selengkapnya, lihat LoadBalancerdan AdvancedConfigurationdi referensi Amazon ECS API.

Saat memperbarui pengontrol penerapan untuk layanan yang digunakan CloudFormation, pertimbangkan hal berikut tergantung pada jenis migrasi yang Anda lakukan.

- Jika Anda memiliki CloudFormation template yang berisi informasi pengontrol EXTERNAL penerapan serta TaskSet PrimaryTaskSet sumber daya, dan Anda menghapus sumber daya set tugas dari templat saat memperbarui dari EXTERNAL keECS, panggilan DeleteTaskSet API DescribeTaskSet dan akan menampilkan kesalahan 400 setelah pengontrol penerapan diperbarui. ECS Hal ini mengakibatkan kegagalan CloudFormation penghapusan pada sumber daya set tugas, meskipun CloudFormation tumpukan bertransisi ke UPDATE_COMPLETE status. Untuk informasi selengkapnya, lihat Sumber daya dihapus dari tumpukan tetapi tidak dihapus di Panduan AWS CloudFormation Pengguna. Untuk memperbaiki masalah ini, hapus set tugas secara langsung menggunakan Amazon ECS

`DeleteTaskSet` API. Untuk informasi selengkapnya tentang cara menghapus kumpulan tugas, lihat DeleteTaskSetdi Referensi API Amazon Elastic Container Service.

- Jika Anda bermigrasi dari `CODE_DEPLOY` ke `ECS` dengan definisi tugas baru dan CloudFormation melakukan operasi rollback, `UpdateService` permintaan Amazon ECS gagal dengan kesalahan berikut:

```
Resource handler returned message: "Invalid request provided: Unable to update
task definition on services with a CODE_DEPLOY deployment controller. Use AWS
CodeDeploy to trigger a new deployment. (Service: Ecs, Status Code: 400,
Request ID: 0abda1e2-f7b3-4e96-b6e9-c8bc585181ac) (SDK Attempt Count: 1)"
(RequestToken: ba8767eb-c99e-efed-6ec8-25011d9473f0, HandlerErrorCode:
 InvalidRequest)
```

- Setelah migrasi berhasil dari `ECS` ke pengontrol `EXTERNAL` penerapan, Anda perlu menghapus set `ACTIVE` tugas secara manual, karena Amazon ECS tidak lagi mengelola penerapan. Untuk informasi tentang cara menghapus kumpulan tugas, lihat DeleteTaskSetdi Referensi API Amazon Elastic Container Service.

Hitungan tugas yang diinginkan

Jumlah instantiasi tugas untuk ditempatkan dan terus berjalan di layanan Anda.

Jika Anda ingin menghentikan sementara layanan Anda, tetapkan nilai ini ke 0. Kemudian, ketika Anda siap untuk memulai layanan, perbarui layanan dengan nilai aslinya.

Anda dapat mengubah parameter ini untuk penerapan bergulir, dan penerapan biru/hijau.

Aktifkan tag terkelola

Menentukan apakah akan mengaktifkan tag terkelola Amazon ECS untuk tugas dalam layanan.

Hanya tugas yang diluncurkan setelah pembaruan yang akan mencerminkan pembaruan. Untuk memperbarui tag pada semua tugas, gunakan opsi penerapan paksa.

Anda dapat mengubah parameter ini untuk penerapan bergulir, dan penerapan biru/hijau.

Aktifkan ECS Exec

Menentukan apakah Amazon ECS Exec digunakan.

Jika Anda tidak ingin mengganti nilai yang ditetapkan saat layanan dibuat, Anda dapat menyetelnya ke null saat melakukan tindakan ini.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Masa tenggang pemeriksaan kesehatan

Periode waktu, dalam hitungan detik, bahwa penjadwal layanan Amazon ECS mengabaikan Elastic Load Balancing, VPC Lattice, dan pemeriksaan kesehatan kontainer yang tidak sehat setelah tugas pertama kali dimulai. Jika Anda tidak menentukan nilai masa tenggang pemeriksaan kesehatan, nilai default `0` digunakan. Jika Anda tidak menggunakan salah satu pemeriksaan kesehatan, maka tidak `healthCheckGracePeriodSeconds` digunakan.

Jika tugas layanan Anda membutuhkan waktu cukup lama untuk memulai dan menanggapi pemeriksaan kesehatan, Anda dapat menentukan masa tenggang pemeriksaan kesehatan hingga 2.147.483.647 detik (sekitar 69 tahun). Selama waktu itu, penjadwal Layanan Amazon ECS mengabaikan status pemeriksaan kondisi. Masa tenggang ini dapat mencegah penjadwal layanan menandai tugas sebagai tidak sehat dan menghentikan mereka sebelum mereka memiliki waktu untuk muncul.

Anda dapat mengubah parameter ini untuk penerapan bergulir dan penerapan biru/hijau.

Penyeimbang beban

Anda harus menggunakan peran terkait layanan saat memperbarui penyeimbang beban.

Daftar objek penyeimbang beban Elastic Load Balancing. Ini berisi nama penyeimbang beban, nama kontainer, dan port kontainer untuk mengakses dari penyeimbang beban. Nama kontainer adalah seperti yang muncul dalam definisi wadah.

Amazon ECS tidak secara otomatis memperbarui grup keamanan yang terkait dengan penyeimbang beban Elastic Load Balancing atau instans kontainer Amazon ECS.

Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai tugas baru dengan konfigurasi Elastic Load Balancing yang diperbarui, dan kemudian menghentikan tugas lama saat tugas baru berjalan.

Untuk layanan yang menggunakan pembaruan bergulir, Anda dapat menambahkan, memperbarui, atau menghapus grup target Elastic Load Balancing. Anda dapat memperbarui dari satu grup target ke beberapa grup target dan dari beberapa grup target ke satu grup target.

Untuk layanan yang menggunakan blue/green penerapan, Anda dapat memperbarui grup target Elastic Load Balancing dengan menggunakannya. `CreateDeployment` CodeDeploy Perhatikan bahwa beberapa grup target tidak didukung untuk blue/green penerapan. Untuk informasi selengkapnya lihat Mendaftarkan beberapa grup target dengan layanan.

Untuk layanan yang menggunakan pengontrol penerapan eksternal, Anda dapat menambahkan, memperbarui, atau menghapus penyeimbang beban dengan menggunakan. CreateTaskSet Perhatikan bahwa beberapa grup target tidak didukung untuk penerapan eksternal. Untuk informasi selengkapnya lihat Mendaftarkan beberapa grup target dengan layanan.

Lewati daftar kosong untuk menghapus penyeimbang beban.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Konfigurasi jaringan

Konfigurasi jaringan layanan.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Kendala penempatan

Array objek kendala penempatan tugas untuk memperbarui layanan yang akan digunakan. Jika tidak ada nilai yang ditentukan, batasan penempatan yang ada untuk layanan akan tetap tidak berubah. Jika nilai ini ditentukan, nilai ini akan mengganti batasan penempatan yang ada yang ditentukan untuk layanan. Untuk menghapus semua kendala penempatan yang ada, tentukan array kosong.

Anda dapat menentukan maksimal 10 kendala untuk setiap tugas. Batas ini mencakup batasan dalam definisi tugas dan yang ditentukan saat runtime.

Anda dapat mengubah parameter ini untuk penerapan bergulir, dan penerapan biru/hijau.

Strategi penempatan

Strategi penempatan tugas keberatan untuk memperbarui layanan yang akan digunakan. Jika tidak ada nilai yang ditentukan, strategi penempatan yang ada untuk layanan akan tetap tidak berubah. Jika nilai ini ditentukan, itu akan mengganti strategi penempatan yang ada yang ditentukan untuk layanan. Untuk menghapus strategi penempatan yang ada, tentukan objek kosong.

Anda dapat mengubah parameter ini untuk penerapan bergulir, dan penerapan biru/hijau.

Versi platform

Versi platform Fargate yang dijalankan layanan Anda.

Layanan yang menggunakan versi platform Linux tidak dapat diperbarui untuk menggunakan versi platform Windows dan sebaliknya.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Menyebarkan tag

Menentukan apakah akan menyebarkan tag dari definisi tugas atau layanan ke tugas. Jika tidak ada nilai yang ditentukan, tanda tidak disebarkan.

Hanya tugas yang diluncurkan setelah pembaruan yang akan mencerminkan pembaruan. Untuk memperbarui tag pada semua tugas, atur `forceNewDeployment` ke`true`, sehingga Amazon ECS memulai tugas baru dengan tag yang diperbarui.

Anda dapat mengubah parameter ini untuk penerapan bergulir, dan penerapan biru/hijau.

Konfigurasi Service Connect

Konfigurasi untuk Amazon ECS Service Connect. Parameter ini menentukan bagaimana layanan terhubung ke layanan lain dalam aplikasi Anda.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Registrasi layanan

Anda harus menggunakan peran terkait layanan saat memperbarui pendaftar layanan.

Rincian untuk pendaftar penemuan layanan untuk ditetapkan ke layanan ini. Untuk informasi lebih lanjut, lihat Penemuan Layanan.

Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi registrasi layanan, Amazon ECS memulai tugas baru dengan konfigurasi pendaftar layanan yang diperbarui, dan kemudian menghentikan tugas lama saat tugas baru berjalan.

Lewati daftar kosong untuk menghapus pendaftar layanan.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Definisi tugas

Definisi tugas dan revisi yang digunakan untuk layanan.

Jika Anda mengubah port yang digunakan oleh kontainer dalam definisi tugas, Anda mungkin perlu memperbarui grup keamanan agar instance kontainer berfungsi dengan port yang diperbarui.

Jika Anda memperbarui definisi tugas untuk layanan, nama kontainer dan port kontainer yang ditentukan dalam konfigurasi penyeimbang beban harus tetap dalam definisi tugas.

Perilaku tarik gambar kontainer berbeda untuk jenis peluncuran. Untuk informasi selengkapnya, lihat salah satu dari berikut ini:

- Jenis peluncuran Fargate untuk Amazon ECS
- EC2 jenis peluncuran untuk Amazon ECS
- Jenis peluncuran eksternal (Amazon ECS Anywhere) untuk Amazon ECS

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Konfigurasi volume

Detail volume yang adaconfiguredAtLaunch. Bila configuredAtLaunch disetel ke true dalam definisi tugas, parameter layanan ini mengonfigurasi satu volume Amazon EBS untuk setiap tugas dalam layanan yang akan dibuat dan dilampirkan selama penerapan. Anda dapat mengonfigurasi ukuran, VolumeType, IOPS, throughput, snapshot, dan enkripsi dalam Konfigurasi. ServiceManaged EBSVolume Volume harus sesuai dengan name dari definisi tugas. name Jika disetel ke null, tidak ada penerapan baru yang dipicu. Jika tidak, jika konfigurasi ini berbeda dari yang sudah ada, itu memicu penerapan baru.

Anda dapat mengubah parameter ini untuk penerapan bergulir.

Konfigurasi Kisi VPC

Konfigurasi VPC Lattice untuk layanan Anda. Ini menentukan bagaimana layanan Anda terintegrasi dengan VPC Lattice untuk komunikasi. service-to-service

Anda dapat mengubah parameter ini untuk penerapan bergulir.

AWS CDK pertimbangan

AWS CDK Tidak melacak status sumber daya. Ia tidak tahu apakah Anda membuat atau memperbarui layanan. Pelanggan harus menggunakan pintu keluar untuk mengakses konstruksi ecs Service L1 secara langsung.

Untuk informasi tentang escape hatch, lihat Menyesuaikan konstruksi dari AWS Construct Library di Panduan Pengembang AWS Cloud Development Kit (AWS CDK) v2.

Untuk memigrasikan layanan yang ada ke ecs.Service konstruksi, lakukan hal berikut:

1. Gunakan palka pelarian untuk mengakses konstruksi Service L1.
2. Secara manual mengatur properti berikut dalam konstruksi Service L1.

Jika layanan Anda menggunakan EC2 kapasitas Amazon:

- `daemon?`
- `placementConstraints?`
- `placementStrategies?`
- Jika Anda menggunakan mode `awsvpc` jaringan, Anda perlu mengatur `vpcSubnets?` dan `securityGroups?` konstruksi.

Jika layanan Anda menggunakan Fargate:

- `FargatePlatformVersion`
- `vpcSubnets?`dan `securityGroups?` konstruksinya.

3. Atur `launchType` sebagai berikut:

```
const cfnEcsService = service.node.findChild('Service') as ecs.CfnService;
cfnEcsService.launchType = "FARGATE";
```

Untuk bermigrasi dari jenis peluncuran ke penyedia kapasitas, lakukan hal berikut:

1. Gunakan palka pelarian untuk mengakses konstruksi `Service` L1.
2. Tambahkan `capacityProviderStrategies?` konstruksinya.
3. Menyebarkan layanan.

Memperbarui layanan Amazon ECS

Setelah Anda membuat layanan, ada kalanya Anda mungkin perlu memperbarui parameter layanan, misalnya jumlah tugas.

Saat Anda memperbarui layanan yang menggunakan pemutus sirkuit Amazon ECS, Amazon ECS membuat penyebaran layanan dan revisi layanan. Sumber daya ini memungkinkan Anda untuk melihat informasi terperinci tentang riwayat layanan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

Prasyarat

Sebelum memperbarui layanan, verifikasi parameter layanan mana yang dapat diubah untuk jenis penerapan Anda. Untuk daftar lengkap parameter yang dapat diubah, lihatPerbarui parameter layanan Amazon ECS.

Prosedur

Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Pada halaman detail klaster, di bagian Layanan, pilih kotak centang di sebelah layanan, lalu pilih Perbarui.
4. Agar layanan Anda memulai penerapan baru, pilih Paksa penerapan baru.
5. Untuk definisi Tugas, pilih keluarga definisi tugas dan revisi.

   > **Important**
   >
   > Konsol memvalidasi bahwa keluarga dan revisi definisi tugas yang dipilih kompatibel dengan konfigurasi komputasi yang ditentukan. Jika Anda menerima peringatan, verifikasi kompatibilitas definisi tugas dan konfigurasi komputasi yang Anda pilih.

6. Jika Anda memilih Replika, untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan dan dipelihara dalam layanan.
7. Jika Anda memilih Replika, agar Amazon ECS memantau distribusi tugas di seluruh Availability Zone, dan mendistribusikannya kembali ketika ada ketidakseimbangan, di bawah penyeimbangan kembali layanan Availability Zone, pilih Availability Zone service rebalancing.
8. Untuk tugas yang menjalankan Min, masukkan batas bawah pada jumlah tugas dalam layanan yang harus tetap dalam `RUNNING` status selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat). Untuk informasi selengkapnya, lihat Konfigurasi penerapan.
9. Untuk tugas yang berjalan Max, masukkan batas atas jumlah tugas dalam layanan yang diizinkan dalam `PENDING` status `RUNNING` atau selama penerapan, sebagai persentase dari jumlah tugas yang diinginkan (dibulatkan ke bilangan bulat terdekat).
10. Untuk mengonfigurasi cara tugas digunakan untuk layanan Anda, perluas opsi Deployment, lalu konfigurasikan opsi Anda.

a. Untuk jenis pengontrol Deployment, tentukan pengontrol penyebaran layanan. Konsol Amazon ECS mendukung jenis pengontrol berikut:ECS.

b. Untuk strategi Deployment, pilih strategi yang digunakan oleh Amazon ECS untuk menerapkan versi baru layanan.

c. Bergantung pada pilihan strategi Deployment, lakukan hal berikut:

Strategi penyebaran	Langkah-langkah
Pembaruan bergulir	a. Untuk Min menjalankan tugas%, tentukan nilai persentase minimum tugas yang harus dijalankan selama penerapan layanan. Untuk informasi selengkapnya, lihat Menyebarkan layanan Amazon ECS dengan mengganti tugas. b. Untuk tugas yang berjalan Maks%, tentukan nilai persentase maksimum tugas yang dapat dijalankan selama penerapan layanan. Untuk informasi selengkapnya, lihat Menyebarkan layanan Amazon ECS dengan mengganti tugas.

Strategi penyebaran	Langkah-langkah	
Biru/hijau	Untuk waktu Panggang, tentukan durasi waktu, dalam hitungan menit, bahwa revisi layanan biru dan hijau harus berjalan secara simultan. Untuk informasi selengkapnya, lihat Penerapan Amazon ECS blue/green .	

d. Untuk menjalankan fungsi Lambda untuk tahap siklus hidup, di bawah Deployment lifecyce hooks lakukan hal berikut untuk setiap fungsi Lambda yang unik:

   i. Pilih Tambahkan.

      Ulangi untuk setiap fungsi unik yang ingin Anda jalankan.

   ii. Untuk fungsi Lambda, masukkan nama fungsi.

   iii. Untuk Peran, pilih peran yang Anda buat dalam prasyarat dengan izin. blue/green

      Untuk informasi selengkapnya, lihat Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green .

   iv. Untuk tahapan Siklus Hidup, pilih tahapan yang dijalankan fungsi Lambda.

   v. (Opsional) Untuk detail Hook, masukkan pasangan nilai kunci yang memberikan informasi tentang hook.

11. Untuk mengonfigurasi cara Amazon ECS mendeteksi dan menangani kegagalan penerapan, perluas deteksi kegagalan Deployment, lalu pilih opsi Anda.

    a. Untuk menghentikan penerapan saat tugas tidak dapat dimulai, pilih Gunakan pemutus sirkuit penyebaran Amazon ECS.

       Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat pemutus sirkuit penyebaran mengatur penerapan ke status gagal, pilih Rollback on failure.

b. Untuk menghentikan penerapan berdasarkan metrik aplikasi, pilih Gunakan CloudWatch alarm. Kemudian, dari nama CloudWatch alarm, pilih alarm. Untuk membuat alarm baru, buka CloudWatch konsol.

   Agar perangkat lunak secara otomatis memutar kembali penerapan ke status penerapan yang terakhir selesai saat CloudWatch alarm menyetel penerapan ke status gagal, pilih Rollback on failure.

12. Untuk mengubah opsi komputasi, perluas konfigurasi Compute, lalu lakukan hal berikut:

    a. Untuk layanan di AWS Fargate, untuk versi Platform, pilih versi baru.

    b. Untuk layanan yang menggunakan strategi penyedia kapasitas, untuk strategi penyedia Kapasitas, lakukan hal berikut:

       - Untuk menambahkan penyedia kapasitas tambahan, pilih Tambahkan lebih banyak. Kemudian, untuk penyedia Kapasitas, pilih penyedia kapasitas.
       - Untuk menghapus penyedia kapasitas, di sebelah kanan penyedia kapasitas, pilih Hapus.

       Layanan yang menggunakan penyedia kapasitas grup Auto Scaling tidak dapat diperbarui untuk menggunakan penyedia kapasitas Fargate. Layanan yang menggunakan penyedia kapasitas Fargate tidak dapat diperbarui untuk menggunakan penyedia kapasitas grup Auto Scaling.

13. (Opsional) Untuk mengonfigurasi Auto Scaling layanan, perluas Service auto scaling, lalu tentukan parameter berikut.Untuk menggunakan predicte auto scaling, yang melihat data beban masa lalu dari arus lalu lintas, konfigurasikan setelah Anda membuat layanan. Untuk informasi selengkapnya, lihat Gunakan pola historis untuk menskalakan layanan Amazon ECS dengan penskalaan prediktif.

    a. Untuk menggunakan penskalaan otomatis servis, pilih Penskalaan otomatis layanan.

    b. Untuk Jumlah tugas minimum, masukkan batas bawah jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan berada di bawah hitungan ini.

    c. Untuk Jumlah tugas maksimum, masukkan batas atas jumlah tugas untuk penskalaan otomatis servis yang akan digunakan. Hitungan yang diinginkan tidak akan melebihi hitungan ini.

    d. Pilih jenis kebijakan. Di bawah Jenis kebijakan penskalaan, pilih salah satu opsi berikut.

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
Pelacakan target	a. Untuk Jenis kebijakan penskalaan, pilih Pelacakan target. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk metrik layanan ECS, pilih salah satu metrik berikut. • ECSServiceRata-rata CPUUtilization - Pemanfaatan CPU rata-rata layanan. • ECSServiceAverageMemoryUtilization— Pemanfaatan memori rata-rata dari layanan. • ALBRequestCountPerTarget— Jumlah permintaan yang diselesaikan per target dalam kelompok sasaran Application Load Balancer. d. Untuk nilai Target, masukkan nilai yang dipertahankan layanan untuk metrik yang dipilih.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	e. Untuk periode cooldown scale-out , masukkan jumlah waktu, dalam detik, setelah aktivitas scale-out (tambahkan tugas) yang harus dilewati sebelum aktivitas scale-out lainnya dapat dimulai. f. Untuk periode cooldown Scale-in, masukkan jumlah waktu, dalam detik, setelah aktivitas scale-in (hapus tugas) yang harus lulus sebelum aktivitas scale-in lainnya dapat dimulai. g. Untuk mencegah kebijakan melakukan aktivitas penskalaan, pilih Matikan penskalaan. h. • (Opsional) Pilih Matikan penskalaan jika Anda ingin kebijakan penskalaan Anda ditingkatkan untuk meningkatkan lalu lintas tetapi tidak membutuhkannya untuk menskalak	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	an saat lalu lintas menurun.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
Penskalaan langkah	a. Untuk Jenis kebijakan penskalaan, pilih Penskalaan langkah. b. Untuk nama Kebijakan , masukkan nama kebijakan. c. Untuk Nama alarm, masukkan nama yang unik untuk alarm. d. Untuk metrik layanan Amazon ECS, pilih metrik yang akan digunakan untuk alarm. e. Untuk Statistik, pilih statistik alarm. f. Untuk Periode, pilih periode untuk alarm. g. Untuk kondisi Alarm, pilih cara membandin gkan metrik yang dipilih dengan ambang batas yang ditentukan. h. Untuk Ambang untuk membandingkan metrik dan periode Evaluasi untuk memulai alarm, masukkan ambang batas yang digunakan untuk alarm dan berapa lama untuk mengevalu asi ambang batas.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	i. Di bawah Tindakan penskalaan, lakukan hal berikut: • Untuk Tindakan, pilih apakah akan menambah, menghapus, atau menetapkan jumlah tertentu yang diinginkan untuk layanan Anda. • Jika Anda memilih untuk menambah atau menghapus tugas, untuk Nilai, masukkan jumlah tugas (atau persen tugas yang ada) untuk ditambahkan atau dihapus saat tindakan penskalaan dimulai. Jika Anda memilih untuk mengatur hitungan yang diinginkan, masukkan jumlah tugas. Untuk Jenis, pilih apakah Nilai adalah bilangan bulat atau nilai persen dari jumlah yang diinginkan yang ada.	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	• Untuk batas Bawah dan Batas atas, masukkan batas bawah dan batas atas penyesuaian penskalaan langkah Anda. Secara default, batas bawah untuk kebijakan penambahan adalah ambang batas alarm, sedangkan batas atas adalah positif (+) tak terbatas. Secara default, batas atas untuk kebijakan penghapusan adalah ambang batas alarm, sedangkan batas bawah adalah negatif (-) tak terbatas. • (Opsional) Tambahkan opsi penskalaan tambahan. Pilih Tambahkan tindakan penskalaan baru, lalu ulangi langkah tindakan Penskalaan. • Untuk periode Cooldown, masukkan jumlah waktu, dalam detik,	

Untuk menggunakan jenis kebijakan ini	Lakukan hal berikut	
	untuk menunggu aktivitas penskalaan sebelumnya diterapkan. Untuk kebijakan add, ini adalah waktu setelah aktivitas scale-out yang kebijakan penskalaan memblokir aktivitas scale-in dan membatasi berapa banyak tugas yang dapat diskalakan pada suatu waktu. Untuk kebijakan penghapusan, ini adalah waktu setelah aktivitas penskalaan yang harus diteruskan sebelum aktivitas penskalaan lainnya dapat dimulai.	

14. (Opsional) Untuk menggunakan Service Connect, pilih Aktifkan Service Connect, lalu tentukan yang berikut ini:

    a. Di bawah konfigurasi Service Connect, tentukan mode klien.

    - Jika layanan Anda menjalankan aplikasi klien jaringan yang hanya perlu terhubung ke layanan lain di namespace, pilih sisi Klien saja.
    - Jika layanan Anda menjalankan aplikasi jaringan atau layanan web dan perlu menyediakan titik akhir untuk layanan ini, dan terhubung ke layanan lain di namespace, pilih Klien dan server.

b. Untuk menggunakan namespace yang bukan namespace cluster default, untuk Namespace, pilih namespace layanan.

15. Jika tugas Anda menggunakan volume data yang kompatibel dengan konfigurasi saat penerapan, Anda dapat mengonfigurasi volume dengan memperluas Volume.

    Nama volume dan jenis volume dikonfigurasi saat Anda membuat revisi definisi tugas dan tidak dapat diubah saat memperbarui layanan. Untuk memperbarui nama dan jenis volume, Anda harus membuat revisi definisi tugas baru dan memperbarui layanan dengan menggunakan revisi baru.

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
Amazon EBS	a. Untuk jenis volume EBS, pilih jenis volume EBS yang ingin Anda lampirkan ke tugas Anda. b. Untuk Ukuran (GiB), masukkan nilai yang valid untuk ukuran volume dalam gibibytes (GiB). Anda dapat menentukan minimal 1 GiB dan maksimum ukuran volume 16,384 GiB. Nilai ini diperlukan kecuali Anda memberikan ID snapshot. c. Untuk IOPS, masukkan jumlah input/output operasi maksimum (IOPS) yang harus disediakan volume. Nilai ini hanya dapat dikonfigurasi untuk `io1`,`io2`, dan jenis gp3 volume.	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	d. Untuk Throughput (MIB/s), masukkan throughput yang harus disediakan volume, dalam mebibytes per detik (, atau MIB/s). MiBps Nilai ini hanya dapat dikonfigurasi untuk jenis gp3 volume. e. Untuk ID Snapshot, pilih snapshot volume Amazon EBS yang ada atau masukkan ARN snapshot jika Anda ingin membuat volume dari snapshot. Anda juga dapat membuat volume baru yang kosong dengan tidak memilih atau memasukkan ID snapshot. f. Jika Anda menentukan ID Snapshot, Anda dapat menentukan tingkat inisialisasi Volume (MIB/s). Masukkan nilai antara 100 dan 300, dalam MIB/s, yang akan menentukan seberapa cepat data dimuat dari snapshot yang ditentukan menggunakan ID Snapshot untuk pembuatan volume.	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	g. Untuk tipe sistem File, pilih jenis sistem file yang akan digunakan untuk penyimpanan data dan pengambilan pada volume. Anda dapat memilih default sistem operasi atau jenis sistem file tertentu. Default untuk Linux adalah`XFS`. Untuk volume yang dibuat dari snapshot, Anda harus menentuka n jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai. h. Untuk peran Infrastruktur, pilih peran IAM dengan izin yang diperlukan yang memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas. Anda dapat melampirkan kebijakan `AmazonECS Infrastru ctureRole PolicyForVolumes` terkelola ke peran, atau	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	Anda dapat menggunakan kebijakan sebagai panduan untuk membuat dan melampirkan kebijakan Anda sendiri dengan izin yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang izin yang diperlukan, lihat Peran IAM infrastruktur Amazon ECS. i. Untuk Enkripsi, pilih Default jika Anda ingin menggunakan enkripsi Amazon EBS dengan pengaturan default. Jika akun Anda memiliki Enkripsi yang dikonfigurasi secara default, volume akan dienkripsi dengan kunci AWS Key Management Service (AWS KMS) yang ditentukan dalam pengaturan. Jika Anda memilih Default dan enkripsi default Amazon EBS tidak diaktifkan, volume tidak akan dienkripsi.	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	Jika Anda memilih Kustom, Anda dapat menentukan AWS KMS key pilihan Anda untuk enkripsi volume. Jika Anda memilih None, volume akan tidak dienkripsi kecuali Anda memiliki enkripsi secara default dikonfigurasi, atau jika Anda membuat volume dari snapshot terenkripsi. j. Jika Anda telah memilih Kustom untuk Enkripsi, Anda harus menentuka n AWS KMS key yang ingin Anda gunakan. Untuk kunci KMS, pilih AWS KMS key atau masukkan tombol ARN. Jika Anda memilih untuk mengenkripsi volume dengan menggunakan kunci terkelola pelanggan simetris, pastikan Anda memiliki izin yang tepat yang ditentukan dalam kebijakan Anda. AWS KMS key Untuk informasi selengkapnya, lihat	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	Enkripsi data untuk volume Amazon EBS. k. (Opsional) Di bawah Tag, Anda dapat menambahkan tag ke volume Amazon EBS Anda dengan menyebarkan tag dari definisi tugas atau layanan, atau dengan menyediakan tag Anda sendiri. Jika Anda ingin menyebarkan tag dari definisi tugas, pilih Definisi tugas untuk menyebarkan tag dari. Jika Anda ingin menyebarkan tag dari layanan, pilih Layanan untuk menyebarkan tag dari. Jika Anda memilih Jangan menyebarkan, atau jika Anda tidak memilih nilai, tag tidak disebarkan. Jika Anda ingin memberikan tag Anda sendiri, pilih Tambah tag dan kemudian berikan kunci dan nilai untuk setiap tag yang Anda tambahkan.	

Untuk mengkonfigurasi jenis volume ini	Lakukan hal berikut	
	Untuk informasi selengkapnya tentang menandai volume Amazon EBS, lihat Menandai volume Amazon EBS.	

16. (Opsional) Untuk membantu mengidentifikasi layanan Anda, perluas bagian Tag, lalu konfigurasikan tag Anda.
    - [Tambahkan tag] Pilih Tambah tag, dan lakukan hal berikut:
        - Untuk Kunci, masukkan nama kunci.
        - Untuk Nilai, masukkan nilai kunci.
    - [Menghapus tanda] Di samping tanda, pilih Hapus tanda.
17. Pilih Perbarui.

AWS CLI

- Jalankan `update-service`. Untuk informasi tentang menjalankan perintah, lihat update-service di Referensi. AWS Command Line Interface

  `update-service`Contoh berikut memperbarui jumlah tugas yang diinginkan dari layanan `my-http-service` ke 2.

  Ganti *`user-input`* dengan nilai-nilai Anda.

```
aws ecs update-service \
    --cluster MyCluster \
    --service my-http-service \
    --desired-count 2
```

Langkah selanjutnya

Lacak penyebaran Anda dan lihat riwayat layanan Anda untuk layanan yang merupakan pemutus sirkuit Amazon ECS. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS.

Menghapus layanan Amazon ECS menggunakan konsol

Berikut ini adalah beberapa alasan Anda akan menghapus layanan:

- Aplikasi tidak lagi diperlukan
- Anda memigrasikan layanan ke lingkungan baru
- Aplikasi ini tidak aktif digunakan
- Aplikasi ini menggunakan lebih banyak sumber daya daripada yang dibutuhkan dan Anda mencoba untuk memilih biaya Anda

Layanan secara otomatis diperkecil menjadi nol sebelum dihapus. Sumber daya penyeimbang beban atau sumber daya penemuan layanan yang terkait dengan layanan tidak terpengaruh oleh penghapusan layanan. Untuk menghapus sumber daya Elastic Load Balancing Anda, lihat salah satu topik berikut, tergantung pada jenis penyeimbang beban Anda: Hapus Application Load Balancer atau Hapus Network Load Balancer.

Saat Anda menghapus layanan, Amazon ECS menghapus semua penerapan layanan dan revisi layanan untuk layanan tersebut.

Saat Anda menghapus layanan, jika masih ada tugas yang berjalan yang memerlukan pembersihan, status layanan berpindah dari `ACTIVE` ke`DRAINING`, dan layanan tidak lagi terlihat di konsol atau dalam operasi `ListServices` API. Setelah semua tugas dialihkan, baik ke status `STOPPING` ataupun `STOPPED`, maka status layanan bergerak dari `DRAINING` ke `INACTIVE`. Layanan dalam `INACTIVE` status `DRAINING` atau masih dapat dilihat dengan operasi `DescribeServices` API.

> **Important**
>
> Jika Anda mencoba membuat layanan baru dengan nama yang sama dengan layanan yang ada di salah satu `ACTIVE` atau `DRAINING` status, Anda akan menerima kesalahan.

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster untuk layanan.
3. Pada halaman Clusters, pilih cluster.
4. Pada *name* halaman Cluster:, pilih tab Layanan.
5. Pilih layanan, lalu pilih Hapus.
6. Untuk menghapus layanan meskipun tidak diperkecil menjadi nol tugas, pilih Paksa hapus layanan.
7. Pada prompt konfirmasi, masukkan hapus, lalu pilih Hapus.

Migrasikan ARN layanan singkat Amazon ECS ke ARN yang panjang

Amazon ECS memberikan Nama Sumber Daya Amazon (ARN) unik untuk setiap layanan. Layanan yang dibuat sebelum 2021 memiliki format ARN pendek:

`arn:aws:ecs:`*`region`*`:`*`aws_account_id`*`:service/`*`service-name`*

Amazon ECS mengubah format ARN untuk menyertakan nama cluster. Ini adalah format ARN yang panjang:

`arn:aws:ecs:`*`region`*`:`*`aws_account_id`*`:service/`*`cluster-name`*`/`*`service-name`*

Layanan Anda harus memiliki format ARN yang panjang untuk menandai layanan Anda.

Anda dapat memigrasikan layanan dengan format ARN pendek ke format ARN yang panjang tanpa harus membuat ulang layanan. Anda dapat menggunakan API, CLI, atau konsol. Anda tidak dapat membatalkan operasi migrasi.

Proses migrasi berjalan lancar dan memastikan nol downtime untuk layanan Anda. Selama migrasi:

- Ketersediaan layanan: Layanan Anda terus berjalan normal tanpa gangguan pada lalu lintas atau fungsionalitas.
- Menjalankan tugas: Tugas yang ada terus berjalan tanpa gangguan. Tugas baru yang diluncurkan setelah migrasi akan menggunakan format ARN panjang jika pengaturan `taskLongArnFormat` akun diaktifkan.
- Instance kontainer: Instance kontainer tidak terpengaruh oleh migrasi ARN layanan dan terus beroperasi secara normal.

- Konfigurasi layanan: Semua pengaturan layanan, termasuk definisi tugas, jaringan, dan konfigurasi penyeimbang beban, tetap tidak berubah.

Jika Anda ingin menggunakan AWS CloudFormation untuk menandai layanan dengan format ARN pendek, Anda harus memigrasikan layanan menggunakan API, CLI, atau konsol. Setelah migrasi selesai, Anda dapat menggunakan AWS CloudFormation untuk menandai layanan.

Jika Anda ingin menggunakan Terraform untuk menandai layanan dengan format ARN pendek, Anda harus memigrasikan layanan menggunakan API, CLI, atau konsol. Setelah migrasi selesai, Anda dapat menggunakan Terraform untuk menandai layanan.

Setelah migrasi selesai, layanan memiliki perubahan berikut:

- Format ARN yang panjang

  `arn:aws:ecs:`*`region`*`:`*`aws_account_id`*`:service/`*`cluster-name`*`/`*`service-name`*

- Saat Anda bermigrasi menggunakan konsol, Amazon ECS menambahkan tag ke layanan dengan kunci disetel ke “ecs: serviceArnMigrated At” dan nilai yang disetel ke stempel waktu migrasi (format UTC).

  Tag ini diperhitungkan dalam kuota tag Anda.

- Ketika `PhysicalResourceId` dalam AWS CloudFormation tumpukan mewakili ARN layanan, nilainya tidak berubah dan akan terus menjadi ARN layanan singkat.

Prasyarat

Lakukan operasi berikut sebelum Anda memigrasikan layanan ARN.

1. Untuk melihat apakah Anda memiliki ARN layanan singkat, lihat detail layanan di konsol Amazon ECS (Anda melihat peringatan ketika layanan memiliki format ARN pendek), atau parameter pengembalian dari. `serviceARN` `describe-services` Ketika ARN tidak menyertakan nama cluster, Anda memiliki ARN pendek. Berikut ini adalah format ARN pendek:

   `arn:aws:ecs:`*`region`*`:`*`aws_account_id`*`:service/`*`service-name`*

2. Perhatikan yang dibuat pada tanggal.
3. Jika Anda memiliki kebijakan IAM yang menggunakan format ARN pendek, perbarui ke format ARN yang panjang.

Ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

```
arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
```

Untuk informasi selengkapnya, lihat Mengedit kebijakan IAM di Panduan AWS Identity and Access Management Pengguna.

4. Jika Anda memiliki alat yang menggunakan format ARN pendek, perbarui ke format ARN yang panjang.

   Ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

   ```
   arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
   ```

5. Aktifkan format ARN panjang layanan. Jalankan `put-account-setting` dengan `serviceLongArnFormat` opsi yang disetel ke`enabled`. Untuk informasi selengkapnya, lihat, put-account-settingdi Referensi API Amazon Elastic Container Service.

   Jalankan perintah sebagai pengguna root ketika layanan Anda memiliki `createdAt` tanggal yang tidak diketahui.

   ```
   aws ecs put-account-setting --name serviceLongArnFormat --value enabled
   ```

   Contoh Output

   ```
   {
       "setting": {
           "name": "serviceLongArnFormat",
           "value": "enabled",
           "principalArn": "arn:aws:iam::123456789012:role/your-role",
           "type": user
       }
   }
   ```

6. Aktifkan format ARN tugas yang panjang. Pengaturan akun ini mengontrol format ARN untuk tugas baru yang diluncurkan setelah migrasi layanan selesai. Jalankan `put-account-setting` dengan `taskLongArnFormat` opsi yang disetel ke`enabled`. Untuk informasi selengkapnya, lihat, put-account-settingdi Referensi API Amazon Elastic Container Service.

   Jalankan perintah sebagai pengguna root ketika layanan Anda memiliki `createdAt` tanggal yang tidak diketahui.

```
aws ecs put-account-setting --name taskLongArnFormat --value enabled
```

Contoh Output

```
{
    "setting": {
        "name": "taskLongArnFormat",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:role/your-role",
        "type": user
    }
}
```

> **Note**
>
> `taskLongArnFormat`Pengaturan tidak secara langsung memigrasikan tugas yang ada. Ini hanya mempengaruhi format ARN dari tugas baru yang dibuat setelah pengaturan diaktifkan. Tugas berjalan yang ada mempertahankan format ARN mereka saat ini sampai mereka diganti melalui operasi layanan normal (seperti penerapan atau aktivitas penskalaan).

Prosedur

Gunakan yang berikut ini untuk memigrasikan ARN layanan Anda.

Konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster.
3. Di bagian Layanan, pilih layanan yang memiliki peringatan di kolom ARN.

   Halaman detail layanan muncul.
4. Pilih Migrasi ke ARN panjang.

   Kotak dialog Migrasi layanan muncul.
5. Pilih Migrasikan.

CLI

Setelah Anda menyelesaikan prasyarat, Anda dapat menandai layanan Anda. Jalankan perintah berikut:

Amazon ECS mempertimbangkan untuk meneruskan format ARN panjang dalam `tag-resource` permintaan API untuk layanan dengan ARN pendek sebagai sinyal untuk memigrasikan layanan untuk menggunakan format ARN yang panjang.

```
aws ecs tag-resource \
    --resource-arn arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
    --tags key=key1,value=value1
```

Contoh tag berikut MyService dengan tag yang memiliki kunci disetel ke "TestService" dan nilai yang disetel ke "WebServers:

```
aws ecs tag-resource \
    --resource-arn arn:aws:ecs:us-east-1:123456789012:service/MyCluster/MyService
    --tags key=TestService1,value=WebServers
```

Terraform

Setelah Anda menyelesaikan prasyarat, Anda dapat menandai layanan Anda. Buat `aws_ecs_service` sumber daya dan atur `tags` referensi. Untuk informasi selengkapnya, lihat Resource: aws_ecs_service di dokumentasi Terraform.

```
resource "aws_ecs_service" "MyService" {
  name    = "example"
  cluster = aws_ecs_cluster.MyService.id

 tags = {
 "Name"  =  "MyService"
 "Environment"  =  "Production"
 "Department"  =  "QualityAssurance"
  }
}
```

Langkah selanjutnya

Anda dapat menambahkan tag ke layanan. Untuk informasi selengkapnya, lihat Menambahkan tag ke sumber daya Amazon ECS.

Jika Anda ingin Amazon ECS menyebarkan tag dari definisi tugas atau layanan ke tugas, jalankan `update-service` dengan parameter. `propagateTags` Untuk informasi selengkapnya, lihat update-service di Referensi. AWS Command Line Interface

Pemecahan Masalah

Beberapa pengguna mungkin mengalami kesalahan berikut saat mereka bermigrasi dari format ARN pendek ke format ARN yang panjang.

`There was an error while migrating the ARN of service service-name. The specified account does not have serviceLongArnFormat or taskLongArnFormat account settings enabled. Add account settings in order to enable tagging.`

Jika Anda telah mengaktifkan pengaturan `serviceLongArnFormat` akun tetapi masih mengalami kesalahan ini, mungkin karena pengaturan akun untuk format ARN yang panjang belum diaktifkan untuk prinsipal IAM tertentu yang awalnya membuat layanan.

1. Identifikasi kepala sekolah yang menciptakan layanan.

   1. Di konsol, informasi tersedia di bidang Dibuat oleh di tab Konfigurasi dan jaringan pada halaman Detail layanan di konsol Amazon ECS.
   2. Untuk AWS CLI, jalankan perintah berikut:

      Ganti *user-input* dengan nilai-nilai Anda.

      ```
      aws ecs describe-services --cluster cluster-name --services service-name --query
       'services[0].{createdBy: createdBy}'
      ```

2. Aktifkan pengaturan akun yang diperlukan untuk prinsipal tertentu. Anda dapat melakukannya dengan salah satu cara berikut:

   a. Asumsikan pengguna IAM atau peran untuk prinsipal itu. Lalu lariput`-account-setting`.

   b. Gunakan pengguna root untuk menjalankan perintah sambil menentukan prinsip pembuatan dengan. `principal-arn`

      Contoh

      Ganti *principal-arn* dengan nilai dari Langkah 1.

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --
principal-arn arn:aws:iam::123456789012:role/jdoe
```

Kedua metode mengaktifkan pengaturan `serviceLongArnFormat` akun yang diperlukan pada prinsipal yang membuat layanan, yang memungkinkan migrasi ARN untuk melanjutkan.

Logika throttle layanan Amazon ECS

Penjadwal layanan Amazon ECS menyertakan logika pelindung yang membatasi peluncuran tugas ketika tugas berulang kali gagal dimulai. Ini membantu mencegah konsumsi sumber daya yang tidak perlu dan mengurangi biaya.

Ketika tugas dalam layanan gagal bertransisi dari `PENDING` ke `RUNNING` status dan sebagai gantinya pindah langsung ke`STOPPED`, penjadwal:

- Secara bertahap meningkatkan waktu antara upaya restart
- Terus meningkatkan penundaan hingga maksimal 27 menit di antara upaya
- Menghasilkan pesan acara layanan untuk memberi tahu Anda tentang masalah

> **Note**
>
> Periode penundaan maksimum 27 menit dapat berubah di pembaruan di masa mendatang.

Ketika throttling diaktifkan, Anda menerima pesan acara layanan ini:

```
(service service-name) is unable to consistently start tasks successfully.
```

Karakteristik penting dari logika throttle:

- Layanan melanjutkan upaya coba lagi tanpa batas
- Satu-satunya modifikasi adalah peningkatan waktu antara restart
- Tidak ada parameter yang dapat dikonfigurasi pengguna

Menyelesaikan masalah pelambatan

Untuk mengatasi pelambatan, Anda dapat:

- Perbarui layanan untuk menggunakan definisi tugas baru, yang segera mengembalikan layanan ke operasi normal yang tidak dibatasi. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.
- Mengatasi penyebab yang mendasari kegagalan tugas.

Penyebab umum kegagalan tugas yang memicu pelambatan meliputi:

- Sumber daya cluster tidak mencukupi (port, memori, atau CPU)
  - Ditunjukkan oleh pesan acara layanan sumber daya yang tidak memadai
- Kegagalan tarik gambar kontainer
  - Dapat disebabkan oleh nama gambar yang tidak valid, tag, atau izin yang tidak memadai
  - Hasil `CannotPullContainerError` dalam Melihat Amazon ECS menghentikan kesalahan tugas
- Ruang disk tak cukup
  - `CannotCreateContainerError`Menghasilkan kesalahan tugas yang dihentikan
  - Untuk langkah-langkah resolusi, lihat Memecahkan masalah Docker di Amazon `API error (500): devmapper` ECS

> **Important**
>
> Skenario berikut TIDAK memicu logika throttle:
>
> - Tugas yang berhenti setelah mencapai `RUNNING` status
> - Tugas dihentikan karena pemeriksaan kesehatan Elastic Load Balancing gagal
> - Tugas di mana perintah kontainer keluar dengan kode bukan nol setelah mencapai status `RUNNING`

Parameter definisi layanan Amazon ECS

Definisi layanan menentukan cara menjalankan layanan Amazon ECS Anda. Parameter berikut dapat ditetapkan dalam ketentuan layanan.

Tipe peluncuran

`launchType`

Tipe: String

Nilai yang valid: `EC2` | `FARGATE` | `EXTERNAL`

Wajib: Tidak

Jenis peluncuran untuk menjalankan layanan Anda. Jika jenis peluncuran tidak ditentukan, default `capacityProviderStrategy` digunakan secara default. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.

Jika `launchType` ditentukan, parameter `capacityProviderStrategy` harus dihilangkan.

Strategi penyedia kapasitas

`capacityProviderStrategy`

Tipe: Array objek

Wajib: Tidak

Strategi penyedia kapasitas untuk digunakan untuk layanan.

Strategi penyedia kapasitas terdiri dari satu penyedia kapasitas atau lebih bersama dengan `base` dan `weight` yang ditetapkan padanya. Penyedia kapasitas harus dikaitkan dengan klaster sebelum ditentukan dalam strategi penyedia kapasitas. PutClusterCapacityProviders API digunakan untuk mengaitkan penyedia kapasitas dengan cluster. Hanya penyedia kapasitas dengan status `ACTIVE` atau `UPDATING` yang dapat digunakan.

Jika `capacityProviderStrategy` ditetapkan, parameter `launchType` harus dihilangkan. Jika tidak ada `capacityProviderStrategy` atau `launchType` ditentukan, `defaultCapacityProviderStrategy` untuk klaster digunakan.

Jika Anda ingin menentukan penyedia kapasitas yang menggunakan grup Auto Scaling, penyedia kapasitas harus sudah dibuat. Penyedia kapasitas baru dapat dibuat dengan operasi CreateCapacityProvider API.

Untuk menggunakan penyedia kapasitas AWS Fargate, tentukan penyedia `FARGATE_SPOT` kapasitas `FARGATE` atau penyedia. Penyedia kapasitas AWS Fargate tersedia untuk semua akun dan hanya perlu dikaitkan dengan cluster yang akan digunakan.

Operasi PutClusterCapacityProviders API digunakan untuk memperbarui daftar penyedia kapasitas yang tersedia untuk klaster setelah cluster dibuat.

`capacityProvider`

Tipe: String

Diperlukan: Ya

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari penyedia kapasitas.

`weight`

Jenis: Integer

Kisaran yang valid: Bilangan bulat antara 0 dan 1.000.

Wajib: Tidak

Nilai bobot menunjukkan persentase relatif dari jumlah total tugas yang diluncurkan yang menggunakan penyedia kapasitas yang ditentukan.

Misalnya, asumsikan bahwa Anda memiliki strategi yang berisi dua penyedia kapasitas dan keduanya memiliki bobot satu. Ketika basis terpenuhi, tugas-tugas terbagi secara merata di dua penyedia kapasitas. Menggunakan logika yang sama, asumsikan bahwa Anda menentukan bobot 1 untuk CapacityProvidera dan bobot 4 untuk CapacityProviderB. Kemudian, untuk setiap satu tugas yang dijalankan menggunakan CapacityProvidera, empat tugas menggunakan CapacityProviderB.

`base`

Jenis: Integer

Kisaran valid: Bilangan bulat antara 0 dan 100.000.

Wajib: Tidak

Nilai dasar menunjuk berapa banyak tugas, minimal, yang akan berjalan di penyedia kapasitas yang ditentukan. Hanya satu penyedia kapasitas dalam strategi penyedia kapasitas dapat memiliki basis yang ditentukan.

Ketentuan tugas

`taskDefinition`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon `revision` (ARN`family:revision`) `family` dan () atau lengkap dari definisi tugas yang akan dijalankan di layanan Anda. Jika a `revision` tidak ditentukan, `ACTIVE` revisi terbaru dari keluarga yang ditentukan akan digunakan.

Ketentuan tugas harus ditentukan ketika menggunakan pembaruan bergulir (`ECS`) pengendali deployment.

Sistem operasi platform

`platformFamily`

Jenis: string

Wajib: Bersyarat

Standar: Linux

Parameter ini diperlukan untuk layanan Amazon ECS yang dihosting di Fargate.

Parameter ini diabaikan untuk layanan Amazon ECS yang dihosting di Amazon EC2.

Sistem operasi pada kontainer yang menjalankan layanan. Nilai yang valid adalah`LINUX`,`WINDOWS_SERVER_2019_FULL`,`WINDOWS_SERVER_2019_CORE`,`WINDOWS_SERVER_202` dan`WINDOWS_SERVER_2022_CORE`.

`platformFamily`Nilai untuk setiap tugas yang Anda tentukan untuk layanan harus sesuai dengan `platformFamily` nilai layanan. Misalnya, jika Anda mengatur `platformFamily` ke`WINDOWS_SERVER_2019_FULL`, `platformFamily` nilai untuk semua tugas harus`WINDOWS_SERVER_2019_FULL`.

Versi platform

`platformVersion`

Tipe: String

Wajib: Tidak

Versi platform tempat tugas Anda dalam layanan berjalan. Versi platform hanya ditentukan untuk tugas yang menggunakan tipe peluncuran Fargate. Jika tidak ditentukan, versi terbaru (`LATEST`) digunakan secara default.

AWS Versi platform Fargate digunakan untuk merujuk ke lingkungan runtime tertentu untuk infrastruktur tugas Fargate. Saat menentukan versi `LATEST` platform saat menjalankan tugas atau membuat layanan, Anda mendapatkan versi platform terbaru yang tersedia untuk tugas Anda. Saat Anda meningkatkan layanan Anda, tugas-tugas tersebut menerima versi platform yang ditentukan pada penerapan layanan saat ini. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.

> **Note**
>
> Versi platform tidak ditentukan untuk tugas yang menggunakan jenis EC2 peluncuran.

Klaster

`cluster`

Tipe: String

Wajib: Tidak

Nama pendek atau Amazon Resource Name (ARN) lengkap dari klaster untuk menjalankan layanan Anda. Jika Anda tidak menentukan cluster, `default` cluster diasumsikan.

Nama layanan

`serviceName`

Tipe: String

Diperlukan: Ya

Nama layanan Anda. Mengizinkan hingga 255 huruf (huruf besar dan huruf kecil), angka, tanda hubung, dan garis bawah. Nama layanan harus unik dalam sebuah klaster, tetapi Anda dapat memiliki layanan yang bernama sama di beberapa klaster dalam satu Wilayah atau lebih.

Strategi penjadwalan

`schedulingStrategy`

Tipe: String

Nilai yang valid: `REPLICA` | `DAEMON`

Wajib: Tidak

Strategi penjadwalan yang boleh digunakan. Jika strategi penjadwalan tidak ditentukan, maka strategi `REPLICA` digunakan. Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS.

Ada dua strategi penjadwal layanan yang tersedia:

- `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh cluster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Strategi penjadwalan replika.
- `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Strategi penjadwalan daemon.

> **Note**
>
> Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan.

Jumlah yang diinginkan

`desiredCount`

Tipe: Integer

Wajib: Tidak

Jumlah instantiasi definisi tugas yang ditentukan untuk ditempatkan dan terus berjalan di layanan Anda.

Parameter ini diperlukan jika strategi penjadwalan `REPLICA` digunakan. Jika layanan menggunakan strategi penjadwalan `DAEMON`, parameter ini bersifat opsional.

Saat Anda menggunakan penskalaan otomatis servis, saat Anda memperbarui layanan yang sedang berjalan dengan jumlah tugas yang sedang berjalan `desiredCount` kurang dari jumlah tugas yang sedang berjalan, layanan akan menurunkan skala ke yang ditentukan`desiredCount`.

Konfigurasi deployment

`deploymentConfiguration`

Tipe: Objek

Wajib: Tidak

Parameter deployment opsional yang mengendalikan berapa banyak tugas yang berjalan selama deployment dan pengurutan tugas berhenti dan memulai.

`maximumPercent`

Tipe: Integer

Wajib: Tidak

Jika layanan menggunakan tipe penerapan rolling update (`ECS`), `maximumPercent` parameter tersebut mewakili batas atas jumlah tugas layanan Anda yang diizinkan dalam `RUNNING`STOPPING, atau `PENDING` status selama penerapan. Hal ini dinyatakan sebagai persentase dari `desiredCount` yang dibulatkan ke bawah ke bilangan bulat terdekat. Anda dapat menggunakan parameter ini untuk menentukan ukuran batch deployment.

Misalnya, jika layanan Anda menggunakan penjadwal `REPLICA` layanan dan memiliki `desiredCount` empat tugas dan `maximumPercent` nilai 200%, penjadwal memulai empat tugas baru sebelum menghentikan empat tugas lama. Ini asalkan sumber daya cluster yang diperlukan untuk melakukan ini tersedia. Nilai `maximumPercent` default untuk layanan yang menggunakan penjadwal layanan `REPLICA` adalah 200%.

Penjadwal Amazon ECS menggunakan parameter ini untuk mengganti tugas yang tidak sehat dengan memulai tugas pengganti terlebih dahulu dan kemudian menghentikan tugas yang tidak sehat, selama sumber daya cluster untuk memulai tugas pengganti tersedia. Untuk informasi selengkapnya tentang cara penjadwal menggantikan tugas yang tidak sehat, lihat. Layanan-layanan Amazon ECS

Jika layanan Anda menggunakan jenis penjadwal `DAEMON` layanan, `maximumPercent` harus tetap pada 100%. Ini adalah nilai default.

Jumlah maksimum tugas selama deployment adalah `desiredCount` dikalikan dengan `maximumPercent`/100, dibulatkan ke bawah ke nilai integer terdekat.

Jika layanan menggunakan jenis dan tugas biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan yang menggunakan tipe EC2 peluncuran, nilai persen maksimum disetel ke nilai default. Nilai digunakan untuk menentukan batas atas jumlah tugas dalam layanan yang tetap dalam `RUNNING` status saat instance kontainer berada dalam `DRAINING` status.

> **Note**
>
> Anda tidak dapat menentukan `maximumPercent` nilai kustom untuk layanan yang menggunakan jenis biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan memiliki tugas yang menggunakan jenis peluncuran. EC2

Jika layanan menggunakan tipe biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan, dan tugas dalam layanan menggunakan tipe peluncuran Fargate, nilai persen maksimum tidak digunakan. Nilai masih dikembalikan saat mendeskripsikan layanan Anda.

`minimumHealthyPercent`

Tipe: Integer

Wajib: Tidak

Jika layanan menggunakan tipe penerapan rolling update (ECS), layanan tersebut `minimumHealthyPercent` mewakili batas bawah jumlah tugas layanan Anda yang harus tetap berada dalam `RUNNING` status selama penerapan. Ini dinyatakan sebagai persentase dari `desiredCount` yang dibulatkan ke bilangan bulat terdekat. Anda dapat menggunakan parameter ini untuk menyebarkan tanpa menggunakan kapasitas cluster tambahan.

Misalnya, jika layanan Anda memiliki `desiredCount` empat tugas, 50%, dan 100%, penjadwal layanan menghentikan dua tugas yang ada untuk membebaskan kapasitas cluster sebelum memulai dua tugas baru. `minimumHealthyPercent` `maximumPercent`

Jika ada tugas yang tidak sehat dan jika `maximumPercent` tidak memungkinkan penjadwal Amazon ECS untuk memulai tugas penggantian, penjadwal menghentikan tugas yang tidak sehat one-by-one — menggunakan `minimumHealthyPercent` sebagai kendala — untuk membersihkan kapasitas untuk meluncurkan tugas pengganti. Untuk informasi selengkapnya tentang cara penjadwal menggantikan tugas yang tidak sehat, lihat. Layanan-layanan Amazon ECS

Untuk layanan yang tidak menggunakan penyeimbang beban, pertimbangkan hal berikut:

- Sebuah layanan dianggap baik jika semua kontainer penting dalam tugas dalam layanan lulus pemeriksaan kondisi.
- Jika tugas tidak memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu selama 40 detik setelah tugas mencapai `RUNNING` keadaan sebelum tugas dihitung terhadap total persen sehat minimum.
- Jika suatu tugas memiliki satu atau lebih wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu tugas mencapai status sehat sebelum menghitungnya ke total persen sehat minimum. Sebuah tugas dianggap baik ketika semua kontainer penting dalam tugas telah lulus pemeriksaan kondisi. Jumlah waktu tunggu penjadwal layanan ditentukan oleh pengaturan pemeriksaan kondisi kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.

Untuk layanan yang menggunakan load balancer, perhatikan hal berikut:

- Jika suatu tugas tidak memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu pemeriksaan kesehatan kelompok sasaran penyeimbang beban untuk mengembalikan status sehat sebelum menghitung tugas menuju total persen sehat minimum.
- Jika tugas memiliki wadah penting dengan pemeriksaan kesehatan yang ditentukan, penjadwal layanan menunggu tugas untuk mencapai status sehat dan pemeriksaan

kesehatan kelompok target penyeimbang beban untuk mengembalikan status sehat sebelum menghitung tugas menuju total persen sehat minimum.

Nilai default untuk layanan replika untuk `minimumHealthyPercent` adalah 100%. `minimumHealthyPercent`Nilai default untuk layanan yang menggunakan jadwal `DAEMON` layanan adalah 0% untuk AWS CLI AWS SDKs, the, APIs dan 50% untuk AWS Management Console.

Jumlah minimal tugas sehat selama deployment adalah `desiredCount` dikalikan dengan `minimumHealthyPercent`/100, dibulatkan ke atas ke nilai bilangan bulat terdekat.

Jika layanan menggunakan tipe biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan menjalankan tugas yang menggunakan tipe EC2 peluncuran, nilai persen sehat minimum disetel ke nilai default. Nilai digunakan untuk menentukan batas bawah pada jumlah tugas dalam layanan yang tetap dalam `RUNNING` status saat instance kontainer berada dalam `DRAINING` status.

> **Note**
>
> Anda tidak dapat menentukan `maximumPercent` nilai kustom untuk layanan yang menggunakan jenis biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan memiliki tugas yang menggunakan jenis peluncuran. EC2

Jika layanan menggunakan tipe biru/hijau (`CODE_DEPLOY`) atau `EXTERNAL` penerapan dan menjalankan tugas yang menggunakan tipe peluncuran Fargate, nilai persen sehat minimum tidak digunakan, meskipun layanan tersebut dikembalikan saat mendeskripsikan layanan Anda.

Pengendali deployment

`deploymentController`

Tipe: Objek

Wajib: Tidak

Pengendali deployment yang bisa digunakan untuk layanan. Jika pengendali deployment tidak ditentukan, maka digunakan pengendali `ECS`. Untuk informasi selengkapnya, lihat Layanan-layanan Amazon ECS.

`type`

Tipe: String

Nilai yang valid: `ECS` | `CODE_DEPLOY` | `EXTERNAL`

Wajib: ya

Tipe pengendali deployment yang bisa digunakan. Tersedia tiga tipe pengendali deployment:

`ECS`

Tipe deployment (`ECS`) pembaruan bergulir mencakup penggantian versi yang berjalan sekarang dari kontainer dengan versi terbaru. Jumlah kontainer Amazon ECS yang ditambahkan atau dihapus dari layanan selama pembaruan bergulir dikendalikan dengan menyesuaikan jumlah minimum dan maksimum tugas sehat yang diizinkan selama penerapan layanan, seperti yang ditentukan dalam. deploymentConfiguration

`CODE_DEPLOY`

Jenis penyebaran blue/green (`CODE_DEPLOY`) menggunakan model blue/green penerapan yang didukung oleh CodeDeploy, yang memungkinkan Anda memverifikasi penerapan layanan baru sebelum mengirimkan lalu lintas produksi ke sana.

`EXTERNAL`

Gunakan jenis penerapan eksternal saat Anda ingin menggunakan pengontrol penerapan pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS.

Penempatan tugas

`placementConstraints`

Tipe: Array objek

Wajib: Tidak

Aray objek batasan penempatan yang digunakan untuk tugas dalam layanan Anda. Anda dapat menentukan maksimum 10 kendala per tugas. Batas ini mencakup kendala dalam definisi tugas dan yang ditentukan pada waktu berjalan. Jika Anda menggunakan tipe peluncuran Fargate, batasan penempatan tugas tidak didukung.

`type`

Tipe: String

Wajib: Tidak

Tipe batasan. Gunakan `distinctInstance` untuk memastikan bahwa setiap tugas dalam grup tertentu berjalan di instans kontainer yang berbeda. Gunakan `memberOf` untuk membatasi seleksi ke grup kandidat yang valid. Nilai `distinctInstance` tidak didukung dalam ketentuan tugas.

`expression`

Tipe: String

Wajib: Tidak

Ekspresi bahasa kueri klaster untuk diterapkan pada batasan. Ingat bahwa Anda tidak dapat menentukan ekspresi jika tipe kendalanya adalah `distinctInstance`. Untuk informasi selengkapnya, lihat Buat ekspresi untuk menentukan instance kontainer untuk tugas Amazon ECS.

`placementStrategy`

Tipe: Array objek

Wajib: Tidak

Objek strategi penempatan untuk digunakan untuk tugas di layanan Anda. Anda dapat menentukan maksimal empat aturan strategi per layanan.

`type`

Tipe: String

Nilai yang valid: `random` | `spread` | `binpack`

Wajib: Tidak

Tipe strategi penempatan. Strategi penempatan `random` menempatkan tugas pada kandidat yang tersedia secara acak. Strategi penempatan `spread` menyebarkan penempatan di seluruh kandidat yang tersedia secara merata berdasarkan parameter `field`. `binpack`Strategi menempatkan tugas pada kandidat yang tersedia yang memiliki jumlah sumber daya paling sedikit yang tersedia yang ditentukan dengan `field` parameter. Misalnya,

jika Anda binpack pada memori, tugas ditempatkan pada instance dengan jumlah memori yang tersisa paling sedikit tetapi masih cukup untuk menjalankan tugas.

`field`

Tipe: String

Wajib: Tidak

Bidang tempat menerapkan strategi penempatan. Untuk strategi `spread` penempatan, nilai yang valid adalah `instanceId` (atau`host`, yang memiliki efek yang sama), atau platform atau atribut khusus apa pun yang diterapkan ke instance container, seperti`attribute:ecs.availability-zone`. Untuk strategi penempatan `binpack`, nilai yang valid adalah `cpu` dan `memory`. Untuk strategi penempatan `random`, bidang ini tidak digunakan.

Tanda

`tags`

Tipe: Array objek

Wajib: Tidak

Metadata yang Anda terapkan ke layanan untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tag terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Ketika layanan dihapus, tag akan dihapus juga. Maksimal 50 tag dapat diterapkan ke layanan. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Tidak

Satu bagian dari pasangan nilai kunci yang membentuk tanda. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

Bagian opsional pasangan nilai kunci yang membentuk tanda. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`enableECSManagedTags`

Jenis: Boolean

Nilai yang valid: `true` | `false`

Wajib: Tidak

Menentukan apakah akan menggunakan tag terkelola Amazon ECS untuk tugas dalam layanan. Jika tidak ada nilai yang ditentukan, nilai defaultnya adalah `false`. Untuk informasi selengkapnya, lihat Gunakan tag untuk penagihan.

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE`

Wajib: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan untuk tugas-tugas dalam layanan. Jika tidak ada nilai yang ditentukan, tag tidak disalin. Tag hanya dapat disalin ke tugas dalam layanan selama pembuatan layanan. Untuk menambahkan tag ke tugas setelah pembuatan layanan atau pembuatan tugas, gunakan tindakan `TagResource` API.

Konfigurasi jaringan

`networkConfiguration`

Tipe: Objek

Wajib: Tidak

Konfigurasi jaringan untuk layanan. Parameter ini diperlukan untuk definisi tugas yang menggunakan mode `awsvpc` jaringan untuk menerima Antarmuka Jaringan Elastis mereka sendiri, dan tidak didukung untuk mode jaringan lainnya. Jika menggunakan tipe peluncuran

Fargate, mode `awsvpc` jaringan diperlukan. Untuk informasi selengkapnya tentang jaringan untuk jenis EC2 peluncuran Amazon, lihatOpsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran. Untuk informasi selengkapnya tentang jaringan untuk jenis peluncuran Fargate, lihat opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate.

`awsvpcConfiguration`

Tipe: Objek

Wajib: Tidak

Sebuah objek yang menunjukkan subnet dan grup keamanan untuk tugas atau layanan.

`subnets`

Tipe: Array string

Wajib: Ya

Subnet yang terkait dengan tugas atau layanan. Ada batas 16 subnet yang dapat ditentukan sesuai dengan`awsvpcConfiguration`.

`securityGroups`

Tipe: Array string

Wajib: Tidak

Grup keamanan yang terkait dengan tugas atau layanan. Jika Anda tidak menentukan grup keamanan, grup keamanan default untuk VPC akan digunakan. Ada batas lima grup keamanan yang dapat ditentukan berdasarkan`awsvpcConfiguration`.

`assignPublicIP`

Tipe: String

Nilai yang valid: `ENABLED` | `DISABLED`

Wajib: Tidak

Apakah antarmuka jaringan elastis tugas menerima alamat IP publik. Jika nilai tidak ditentukan, maka digunakan nilai default `DISABLED`.

`healthCheckGracePeriodSeconds`

Tipe: Integer

Wajib: Tidak

Periode waktu, dalam hitungan detik, bahwa penjadwal layanan Amazon ECS mengabaikan Elastic Load Balancing, VPC Lattice, dan pemeriksaan kesehatan kontainer yang tidak sehat setelah tugas pertama kali dimulai. Jika Anda tidak menentukan nilai tenggang pemeriksaan kesehatan, nilai default 0 digunakan. Jika Anda tidak menggunakan salah satu pemeriksaan kesehatan, maka tidak `healthCheckGracePeriodSeconds` digunakan.

Jika tugas layanan Anda membutuhkan waktu cukup lama untuk memulai dan merespons, Anda dapat menentukan masa tenggang pemeriksaan kesehatan hingga 2.147.483.647 detik (sekitar 69 tahun). Selama waktu itu, penjadwal layanan Amazon ECS mengabaikan status pemeriksaan kesehatan. Masa tenggang ini dapat mencegah penjadwal layanan menandai tugas sebagai tidak sehat dan menghentikan mereka sebelum mereka memiliki waktu untuk muncul.

`loadBalancers`

Tipe: Array objek

Wajib: Tidak

Sebuah objek penyeimbang beban menunjukkan penyeimbang beban untuk digunakan dengan layanan Anda. Untuk layanan yang menggunakan Application Load Balancer atau Network Load Balancer, ada batasan lima grup target yang dapat Anda lampirkan ke layanan.

Setelah Anda membuat layanan, konfigurasi penyeimbang beban tidak dapat diubah dari. AWS Management Console Anda dapat menggunakan AWS Copilot, AWS CloudFormation, AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban hanya untuk pengontrol penerapan `ECS` bergulir, bukan biru/hijau atau eksternal. AWS CodeDeploy Saat Anda menambahkan, memperbarui, atau menghapus konfigurasi penyeimbang beban, Amazon ECS memulai penerapan baru dengan konfigurasi Elastic Load Balancing yang diperbarui. Hal ini menyebabkan tugas mendaftar dan membatalkan pendaftaran dari penyeimbang beban. Kami menyarankan Anda memverifikasi ini di lingkungan pengujian sebelum memperbarui konfigurasi Elastic Load Balancing. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service.

Untuk Application Load Balancers dan Network Load Balancers, objek ini harus berisi kelompok target load balancer ARN, nama kontainer (seperti yang muncul dalam definisi kontainer), dan port kontainer untuk mengakses dari penyeimbang beban. Ketika tugas dari layanan ini ditempatkan pada instance kontainer, instance kontainer dan kombinasi port terdaftar sebagai target dalam kelompok target yang ditentukan.

`targetGroupArn`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon (ARN) lengkap dari grup target Elastic Load Balancing yang terkait dengan layanan.

ARN grup target hanya ditentukan saat menggunakan Application Load Balancer atau Penyeimbang Beban Jaringan.

`loadBalancerName`

Tipe: String

Wajib: Tidak

Nama penyeimbang beban untuk dikaitkan dengan layanan.

Jika Anda menggunakan Application Load Balancer atau Network Load Balancer, hilangkan parameter nama load balancer.

`containerName`

Tipe: String

Wajib: Tidak

Nama kontainer (seperti yang muncul dalam ketentuan kontainer) untuk dikaitkan dengan penyeimbang beban.

`containerPort`

Tipe: Bilangan bulat

Wajib: Tidak

Port pada kontainer untuk dikaitkan dengan penyeimbang beban. Port ini harus sesuai dengan `containerPort` dalam ketentuan tugas yang digunakan oleh tugas dalam layanan. Untuk tugas yang menggunakan tipe EC2 peluncuran, instance container harus mengizinkan lalu lintas masuk pada `hostPort` pemetaan port.

`role`

Tipe: String

Wajib: Tidak

Nama pendek atau ARN lengkap dari peran IAM yang memungkinkan Amazon ECS melakukan panggilan ke penyeimbang beban Anda atas nama Anda. Parameter ini hanya diizinkan jika Anda menggunakan penyeimbang beban dengan satu grup target untuk layanan Anda, dan definisi tugas Anda tidak menggunakan mode `awsvpc` jaringan. Jika Anda menentukan `role` parameter, Anda juga harus menentukan objek penyeimbang beban dengan `loadBalancers` parameter.

Jika peran yang Anda tentukan memiliki jalur selain `/`, maka Anda harus menentukan ARN peran penuh (dianjurkan) atau awalan nama peran dengan jalur. Misalnya, jika peran dengan nama `bar` memiliki jalur dari `/foo/` maka Anda akan menentukan `/foo/bar` sebagai nama peran. Untuk informasi selengkapnya, lihat Nama dan Jalur Ramah di Panduan Pengguna IAM.

> **Important**
>
> Jika akun Anda telah membuat peran terkait layanan Amazon ECS, peran tersebut digunakan secara default untuk layanan Anda kecuali Anda menentukan peran di sini. Peran terkait layanan diperlukan jika definisi tugas Anda menggunakan mode jaringan awsvpc, dalam hal ini Anda tidak boleh menentukan peran di sini. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

`serviceConnectConfiguration`

Tipe: Objek

Wajib: Tidak

Konfigurasi untuk layanan ini untuk menemukan dan terhubung ke layanan, dan ditemukan oleh, dan terhubung dari, layanan lain dalam namespace.

Untuk informasi selengkapnya, lihat Gunakan Service Connect untuk menghubungkan layanan Amazon ECS dengan nama pendek.

`enabled`

Jenis: Boolean

Wajib: Ya

Menentukan apakah akan menggunakan Service Connect dengan layanan ini.

`namespace`

Tipe: String

Wajib: Tidak

Nama pendek atau nama sumber daya Amazon lengkap (ARN) dari AWS Cloud Map namespace untuk digunakan dengan Service Connect. Namespace harus Wilayah AWS sama dengan layanan Amazon ECS dan cluster. Jenis namespace tidak memengaruhi Service Connect. Untuk informasi selengkapnya AWS Cloud Map, lihat Bekerja dengan Layanan di Panduan AWS Cloud Map Pengembang.

`services`

Tipe: Array objek

Wajib: Tidak

Array objek layanan Service Connect. Ini adalah nama dan alias (juga dikenal sebagai titik akhir) yang digunakan oleh layanan Amazon ECS lainnya untuk terhubung ke layanan ini.

Bidang ini tidak diperlukan untuk layanan Amazon ECS “klien” yang merupakan anggota namespace hanya untuk terhubung ke layanan lain di dalam namespace. Contohnya adalah aplikasi frontend yang menerima permintaan masuk dari penyeimbang beban yang dilampirkan ke layanan atau dengan cara lain.

Objek memilih port dari definisi tugas, menetapkan nama untuk AWS Cloud Map layanan, dan array alias (juga dikenal sebagai titik akhir) dan port untuk aplikasi klien untuk merujuk ke layanan ini.

`portName`

Tipe: String

Diperlukan: Ya

`portName`Harus cocok dengan `name` salah satu `portMappings` dari semua wadah dalam definisi tugas layanan Amazon ECS ini.

`discoveryName`

Tipe: String

Wajib: Tidak

`discoveryName`Itu adalah nama AWS Cloud Map layanan baru yang dibuat Amazon ECS untuk layanan Amazon ECS ini. Ini harus unik di dalam AWS Cloud Map namespace.

Jika bidang ini tidak ditentukan, `portName` digunakan.

`clientAliases`

Tipe: Array objek

Wajib: Tidak

Daftar alias klien untuk layanan koneksi layanan ini. Anda menggunakan ini untuk menetapkan nama yang dapat digunakan oleh aplikasi klien. Jumlah maksimum alias klien yang dapat Anda miliki dalam daftar ini adalah 1.

Setiap alias (“endpoint”) adalah nama DNS dan nomor port yang dapat digunakan layanan Amazon ECS lainnya (“klien”) untuk terhubung ke layanan ini.

Setiap kombinasi nama dan port harus unik di dalam namespace.

Nama-nama ini dikonfigurasi dalam setiap tugas layanan klien, bukan di AWS Cloud Map. Permintaan DNS untuk menyelesaikan nama-nama ini tidak meninggalkan tugas, dan tidak dihitung terhadap kuota permintaan DNS per detik per elastic network interface.

`port`

Jenis: Integer

Wajib: Ya

Nomor port mendengarkan untuk proxy koneksi layanan. Port ini tersedia di dalam semua tugas dalam namespace yang sama.

Untuk menghindari perubahan aplikasi Anda di layanan Amazon ECS klien, atur ini ke port yang sama yang digunakan aplikasi klien secara default.

`dnsName`

Tipe: String

Wajib: Tidak

Itu `dnsName` adalah nama yang Anda gunakan dalam aplikasi tugas klien untuk terhubung ke layanan ini. Nama harus berupa label DNS yang valid.

Nilai default adalah `discoveryName.namespace` jika bidang ini tidak ditentukan. Jika `discoveryName` tidak ditentukan, definisi `portName` dari tugas digunakan.

Untuk menghindari perubahan aplikasi Anda di layanan Amazon ECS klien, atur ini ke nama yang sama dengan yang digunakan aplikasi klien secara default. Misalnya, beberapa nama umum adalah`database`,db, atau nama huruf kecil dari database, seperti `mysql` atau. `redis`

`ingressPortOverride`

Tipe: Integer

Wajib: Tidak

(Opsional) Nomor port untuk proxy Service Connect untuk didengarkan.

Gunakan nilai bidang ini untuk melewati proxy untuk lalu lintas pada nomor port yang ditentukan dalam nama `portMapping` dalam definisi tugas aplikasi ini, lalu gunakan di grup keamanan Amazon VPC Anda untuk mengizinkan lalu lintas ke proxy untuk layanan Amazon ECS ini.

Dalam `awsvpc` mode, nilai default adalah nomor port kontainer yang ditentukan dalam nama `portMapping` dalam definisi tugas aplikasi ini. Dalam `bridge` mode, nilai default adalah port singkat dinamis dari proxy Service Connect.

`logConfiguration`

Jenis: LogConfigurationObjek

Wajib: Tidak

Ini menentukan di mana log proxy Service Connect dipublikasikan. Gunakan log untuk debugging selama kejadian tak terduga. Konfigurasi ini menetapkan `logConfiguration` parameter dalam wadah proxy Service Connect di setiap tugas di layanan Amazon ECS ini. Kontainer proxy tidak ditentukan dalam definisi tugas.

Kami menyarankan Anda menggunakan konfigurasi log yang sama dengan wadah aplikasi definisi tugas untuk layanan Amazon ECS ini. Untuk FireLens, ini adalah konfigurasi log dari wadah aplikasi. Bukan wadah router FireLens log yang menggunakan gambar `fluent-bit` atau `fluentd` wadah.

`serviceRegistries`

Tipe: Array objek

Wajib: Tidak

Detail konfigurasi pencari layanan untuk layanan Anda. Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.

`registryArn`

Tipe: String

Wajib: Tidak

Nama Sumber Daya Amazon (ARN) dari registri layanan. Registri layanan yang saat ini didukung adalah AWS Cloud Map. Untuk informasi selengkapnya, lihat Bekerja dengan Layanan di Panduan AWS Cloud Map Pengembang.

`port`

Tipe: Integer

Wajib: Tidak

Nilai port yang digunakan jika layanan penemuan layanan Anda menentukan catatan SRV. Kolom ini wajib diisi jika mode jaringan `awsvpc` dan catatan SRV digunakan.

`containerName`

Tipe: String

Wajib: Tidak

Nilai nama kontainer yang akan digunakan untuk layanan penemuan layanan Anda. Nilai ini ditentukan dalam definisi tugas. Jika ketentuan tugas yang menentukan tugas layanan Anda menggunakan mode jaringan `bridge` atau `host`, Anda harus menentukan kombinasi `containerName` dan `containerPort` dari ketentuan tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda menggunakan mode jaringan `awsvpc` dan tipe catatan SRV DNS digunakan, Anda harus menentukan apakah akan menggunakan salah satu kombinasi `containerName` dan `containerPort` atau nilai `port`, tetapi tidak keduanya.

`containerPort`

Tipe: Integer

Wajib: Tidak

Nilai port yang akan digunakan untuk layanan penemuan layanan Anda. Nilai ini ditentukan dalam definisi tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda menggunakan mode jaringan `bridge` atau `host`, Anda harus menentukan kombinasi `containerName` dan `containerPort` dari ketentuan tugas. Jika ketentuan tugas yang ditentukan oleh tugas layanan Anda menggunakan mode jaringan `awsvpc` dan tipe catatan SRV DNS digunakan, Anda harus menentukan apakah akan menggunakan salah satu kombinasi `containerName` dan `containerPort` atau nilai `port`, tetapi tidak keduanya.

Token klien

`clientToken`

Tipe: String

Wajib: Tidak

Pengidentifikasi unik dan peka huruf besar/kecil yang Anda berikan untuk memastikan idempotensi permintaan. Panjangnya bisa mencapai 32 karakter ASCII.

Rebalancing Zona Ketersediaan

`availabilityZoneRebalancing`

Tipe: String

Wajib: Tidak

Menunjukkan apakah layanan menggunakan penyeimbangan kembali Availability Zone. Nilai yang valid adalah dan. `ENABLED` `DISABLED` Untuk informasi selengkapnya tentang penyeimbangan kembali Availability Zone, lihat. Menyeimbangkan layanan Amazon ECS di seluruh Availability Zone

Konfigurasi volume

`volumeConfigurations`

Tipe: Objek

Wajib: Tidak

Konfigurasi yang akan digunakan untuk membuat volume untuk tugas-tugas yang dikelola oleh layanan. Hanya volume yang ditandai seperti `configuredAtLaunch` dalam definisi tugas yang dapat dikonfigurasi dengan menggunakan objek ini.

Objek ini diperlukan untuk melampirkan volume Amazon EBS ke tugas yang dikelola oleh layanan. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EBS dengan Amazon ECS.

`name`

Tipe: String

Diperlukan: Ya

Nama volume yang dikonfigurasi saat membuat atau memperbarui layanan. Hingga 255 huruf (huruf besar dan kecil), angka, garis bawah (), dan tanda hubung (_) diperbolehkan. - Nilai ini harus cocok dengan nama volume yang ditentukan dalam definisi tugas.

`managedEBSVolume`

Tipe: Objek

Wajib: Tidak

Konfigurasi volume yang digunakan untuk membuat volume Amazon EBS yang dilampirkan ke tugas yang dikelola oleh layanan saat layanan dibuat atau diperbarui. Satu volume dilampirkan per tugas.

`encrypted`

Tipe: Boolean

Wajib: Tidak

Nilai yang valid: `true` | `false`

Menentukan apakah akan mengenkripsi setiap volume Amazon EBS dibuat. Jika Anda telah mengaktifkan enkripsi Amazon EBS secara default untuk yang khusus Wilayah AWS untuk Anda Akun AWS tetapi menetapkan parameter ini`false`, parameter ini akan diganti, dan volume akan dienkripsi dengan kunci KMS yang ditentukan untuk enkripsi secara default. Untuk informasi selengkapnya tentang enkripsi Amazon EBS secara default, lihat Mengaktifkan enkripsi Amazon EBS secara default di Panduan Pengguna Amazon EBS. Untuk informasi selengkapnya tentang mengenkripsi volume Amazon EBS yang

dilampirkan ke tugas Amazon ECS, lihat. Enkripsi data yang disimpan dalam volume Amazon EBS yang dilampirkan ke tugas Amazon ECS

`kmsKeyId`

Tipe: String

Wajib: Tidak

Pengenal kunci AWS Key Management Service (AWS KMS) yang akan digunakan untuk enkripsi Amazon EBS. Jika `kmsKeyId` ditentukan, status terenkripsi harus `true`.

Kunci yang ditentukan menggunakan parameter ini akan menggantikan default Amazon EBS atau kunci KMS tingkat cluster apa pun untuk enkripsi penyimpanan terkelola Amazon ECS yang mungkin telah Anda tentukan. Untuk informasi selengkapnya, lihat Enkripsi data yang disimpan dalam volume Amazon EBS yang dilampirkan ke tugas Amazon ECS.

Anda dapat menentukan kunci KMS dengan menggunakan salah satu dari berikut ini:

- ID Kunci — Misalnya,`1234abcd-12ab-34cd-56ef-1234567890ab`.
- Alias kunci — Misalnya,`alias/ExampleAlias`.
- ARN Kunci — Misalnya,. `arn:aws:kms:us-east-1:012345678910:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias ARN — Misalnya,. `arn:aws:kms:us-east-1:012345678910:alias/ExampleAlias`

> **Important**
>
> AWS mengautentikasi kunci KMS secara asinkron. Oleh karena itu, jika Anda menentukan ID, alias, atau ARN yang tidak valid, tindakan dapat tampak berhasil, tetapi akhirnya gagal. Untuk informasi selengkapnya, lihat Memecahkan masalah lampiran volume Amazon EBS.

`volumeType`

Tipe: String

Wajib: Tidak

Nilai yang valid: `gp2` `gp3` | `io1` | `io2` | `sc1` | `st1` | `standard`

Jenis volume Amazon EBS. Untuk informasi selengkapnya tentang jenis volume, lihat Jenis volume Amazon EBS di Panduan Pengguna Amazon EBS. Jenis volume default adalah gp3.

> **Note**
>
> Jenis `standard` volume tidak didukung untuk tugas Fargate.

`sizeInGiB`

Tipe: Integer

Wajib: Tidak

Rentang yang valid: Bilangan bulat antara 1 dan 16.384

Ukuran volume EBS dalam gibibytes (GiB). Jika Anda tidak memberikan ID snapshot untuk mengonfigurasi volume lampiran, Anda harus memberikan nilai ukuran. Jika Anda mengonfigurasi volume untuk lampiran menggunakan snapshot, nilai defaultnya adalah ukuran snapshot. Anda kemudian dapat menentukan ukuran yang lebih besar dari atau sama dengan ukuran snapshot.

Untuk gp2 dan jenis gp3 volume, kisaran yang valid adalah 1-16.384.

Untuk `io1` dan jenis `io2` volume, kisaran yang valid adalah 4-16.384.

Untuk `st1` dan jenis `sc1` volume, kisaran yang valid adalah 125-16.384.

Untuk jenis `standard` volume, kisaran yang valid adalah 1-1.024.

`snapshotId`

Tipe: String

Wajib: Tidak

ID snapshot volume Amazon EBS yang ada yang digunakan Amazon ECS untuk membuat volume baru untuk lampiran. Anda harus menentukan `snapshotId` atau `sizeInGiB`.

`volumeInitializationRate`

Tipe: Integer

Wajib: Tidak

Tingkat, dalam MIB/s, di mana data diambil dari snapshot volume Amazon EBS yang ada untuk membuat volume baru untuk lampiran. Properti ini dapat ditentukan hanya jika Anda menentukan`snapshotId`. Untuk informasi selengkapnya tentang tingkat inisialisasi volume ini, termasuk rentang tarif yang didukung untuk inisialisasi, lihat Menginisialisasi volume Amazon EBS di Panduan Pengguna Amazon EBS.

`iops`

Tipe: Integer

Wajib: Tidak

Jumlah I/O operasi per detik (IOPS). Untuk volume `gp3`, `io1`, dan `io2`, ini mewakili jumlah IOPS yang disediakan untuk volume. Untuk `gp2` volume, nilai ini mewakili kinerja dasar volume dan tingkat di mana volume mengakumulasi I/O kredit untuk meledak. Parameter ini diperlukan untuk volume `io1` dan `io2`. Parameter ini tidak didukung untuk`gp2`,`st1`,`sc1`, atau `standard` volume.

Untuk `gp3` volume, kisaran nilai yang valid adalah 3.000 hingga 16.000.

Untuk `io1` volume, kisaran nilai yang valid adalah 100 hingga 64.000.

Untuk `io2` volume, kisaran nilai yang valid adalah 100 hingga 64.000.

`throughput`

Tipe: Integer

Wajib: Tidak

Throughput untuk penyediaan volume yang melekat pada tugas-tugas yang dikelola oleh layanan.

> **Important**
>
> Parameter ini hanya didukung untuk `gp3` volume.

`roleArn`

Tipe: String

Diperlukan: Ya

Amazon Resource ARN (ARN) dari peran infrastruktur AWS Identity and Access Management (IAM) yang menyediakan izin Amazon ECS untuk mengelola sumber daya Amazon EBS untuk tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM infrastruktur Amazon ECS.

`tagSpecifications`

Tipe: Objek

Wajib: Tidak

Spesifikasi tag yang akan diterapkan pada setiap volume Amazon EBS.

`resourceType`

Tipe: String

Diperlukan: Ya

Nilai yang valid: `volume`

Jenis sumber daya untuk menandai pada penciptaan.

`tags`

Tipe: Array objek

Wajib: Tidak

Metadata yang Anda terapkan pada volume untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tag terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. `AmazonECSCreated`dan `AmazonECSManaged` merupakan tag cadangan yang ditambahkan oleh Amazon ECS atas nama Anda, sehingga Anda dapat menentukan maksimum 48 tag Anda sendiri. Ketika volume dihapus, tag dihapus juga. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.

`key`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Tidak

Salah satu bagian dari pasangan kunci-nilai yang membentuk tag. Kunci adalah label umum yang bertindak seperti kategori untuk nilai tanda yang lebih spesifik.

`value`

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

Bagian opsional dari pasangan kunci-nilai yang membentuk tag. Nilai bertindak sebagai deskriptor dalam kategori tanda (kunci).

`propagateTags`

Tipe: String

Nilai yang valid: `TASK_DEFINITION` | `SERVICE` | `NONE`

Wajib: Tidak

Menentukan apakah akan menyalin tag dari definisi tugas atau layanan ke volume. Jika `NONE` ditentukan atau tidak ada nilai yang ditentukan, tag tidak disalin.

`fileSystemType`

Tipe: String

Wajib: Tidak

Nilai yang valid: `xfs` | `ext3` | `ext4` | `NTFS`

Jenis sistem file pada volume. Jenis sistem file volume menentukan bagaimana data disimpan dan diambil dalam volume. Untuk volume yang dibuat dari snapshot, Anda harus menentukan jenis sistem file yang sama dengan volume yang digunakan saat snapshot dibuat. Jika ada ketidakcocokan tipe sistem file, tugas akan gagal dimulai.

Nilai yang valid untuk Linux adalah`xfs`, ext3`, and ext4`. Default untuk volume yang dilampirkan ke tugas Linux adalah`XFS`.

Nilai yang valid untuk Windows adalah`NTFS`. Default untuk volume yang dilampirkan ke tugas Windows adalah`NTFS`.

Templat definisi layanan

Berikut ini menunjukkan representasi JSON dari definisi layanan Amazon ECS.

Jenis EC2 peluncuran Amazon

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
```

```
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "availabilityZoneRebalancing": "ENABLED",
    "serviceConnectConfiguration": {
```

```
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "volumeInitializationRate": 0,
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
```

```
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Jenis peluncuran Fargate

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "FARGATE",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
```

```
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "platformFamily": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
```

```
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "availabilityZoneRebalancing": "ENABLED",
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
```

```
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "volumeInitializationRate": 0,
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Anda dapat membuat templat ketentuan layanan ini menggunakan perintah AWS CLI berikut.

```
aws ecs create-service --generate-cli-skeleton
```

Menandai sumber daya Amazon ECS

Untuk membantu mengelola sumber daya Amazon ECS, Anda dapat menetapkan metadata Anda sendiri secara opsional ke setiap sumber daya menggunakan tag. Setiap tag terdiri dari kunci dan nilai opsional.

Anda dapat menggunakan tag untuk mengkategorikan sumber daya Amazon ECS Anda dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna jika Anda memiliki banyak sumber daya dengan jenis yang sama. Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang Anda tetapkan padanya. Misalnya, Anda dapat menentukan satu set tag untuk instans penampung Amazon ECS akun Anda. Ini membantu Anda melacak setiap pemilik instans dan tingkat tumpukan.

Anda dapat menggunakan tag untuk laporan Biaya dan Penggunaan Anda. Anda dapat menggunakan laporan ini untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda. Untuk informasi selengkapnya, lihat the section called “Laporan Penggunaan”.

> **Warning**
>
> Ada banyak APIs yang mengembalikan kunci tag dan nilainya. Menolak akses ke `DescribeTags` tidak secara otomatis menolak akses ke tag yang dikembalikan oleh yang lain APIs. Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tag.

Sebaiknya rancang serangkaian kunci tag yang memenuhi kebutuhan setiap jenis sumber daya. Anda dapat menggunakan satu set kunci tag yang konsisten untuk pengelolaan sumber daya yang lebih mudah. Anda dapat mencari dan memfilter sumber daya berdasarkan tag yang Anda tambahkan.

Tag tidak memiliki arti semantik untuk Amazon ECS dan ditafsirkan secara ketat sebagai serangkaian karakter. Anda dapat mengedit kunci dan nilai tanda, dan dapat menghapus tanda dari sumber daya kapan saja. Anda dapat mengatur nilai tanda ke string kosong, akan tetapi Anda tidak dapat mengatur nilai tanda ke nol. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya tersebut, nilai baru akan menimpa nilai sebelumnya. Saat Anda menghapus sumber daya, tag apa pun untuk sumber daya juga dihapus.

Jika Anda menggunakan AWS Identity and Access Management (IAM), Anda dapat mengontrol pengguna mana di AWS akun Anda yang memiliki izin untuk mengelola tag.

Bagaimana sumber daya ditandai

Ada beberapa cara untuk menandai tugas, layanan, definisi tugas, dan cluster Amazon ECS:

- Pengguna secara manual menandai sumber daya dengan menggunakan AWS Management Console, Amazon ECS API, the AWS CLI, atau AWS SDK.
- Pengguna membuat layanan atau menjalankan tugas mandiri dan memilih opsi tag yang dikelola Amazon ECS.

  Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan. Untuk informasi selengkapnya, lihat the section called “Tag yang dikelola Amazon ECS”.
- Pengguna membuat sumber daya menggunakan konsol. Konsol secara otomatis menandai sumber daya.

  Tag ini dikembalikan dalam AWS CLI, dan respons AWS SDK dan ditampilkan di konsol. Anda tidak dapat mengubah atau menghapus tag ini.

  Untuk informasi tentang tag yang ditambahkan, lihat kolom Tag yang ditambahkan secara otomatis oleh konsol di tabel sumber daya Dukungan penandaan untuk Amazon ECS.

Jika Anda menentukan tag saat membuat sumber daya dan tag tidak dapat diterapkan, Amazon ECS akan mengembalikan proses pembuatan. Hal ini untuk memastikan bahwa sumber daya dibuat dengan tanda atau tidak dibuat sama sekali, dan tidak ada sumber daya yang dibiarkan tanpa tanda kapan saja. Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya.

Tabel berikut menjelaskan sumber daya Amazon ECS yang mendukung penandaan.

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Tugas-tugas Amazon ECS	Ya	Ya, dari ketentuan tugas.	Kunci: `aws:ecs:clusterName`

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
			Nilai: `cluster-name`
Layanan-layanan Amazon ECS	Ya	Ya, baik dari ketentuan tugas ataupun layanan ke dalam tugas dalam layanan.	Kunci: `ecs:service:stackId` Nilai `arn:aws:cloudformation: arn`
Set tugas Amazon ECS	Ya	Tidak	N/A
Definisi tugas Amazon ECS	Ya	Tidak	Kunci: `ecs:taskDefinition:createdFrom` Nilai: `ecs-console-v2`

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Klaster-klaster Amazon ECS	Ya	Tidak	Kunci: `aws:cloudformation:logical-id` Nilai: `ECSCluster` Kunci: `aws:cloudformation:stack-id` Nilai: `arn:aws:cloudformation: arn` Kunci: `aws:cloudformation:stack-name` Nilai: `ECS-Console-V2-Cluster- EXAMPLE`
Instans kontainer Amazon ECS	Ya	Ya, dari EC2 contoh Amazon. Untuk informasi selengkapnya, lihat Menambahkan tag ke instans penampung Amazon ECS.	N/A
Instans Eksternal Amazon ECS	Ya	Tidak	N/A

Sumber daya	Mendukung tanda	Penyebaran tanda Support	Tag secara otomatis ditambahkan oleh konsol
Penyedia kapasitas Amazon ECS	Ya. Anda tidak dapat menandai penyedia yang telah ditentukan `FARGATE` dan `FARGATE_SPOT` kapasitas.	Tidak	N/A

Menandai sumber daya pada pembuatan

Sumber daya berikut mendukung penandaan pada pembuatan menggunakan Amazon ECS API, AWS CLI, atau AWS SDK:

- Tugas-tugas Amazon ECS
- Layanan-layanan Amazon ECS
- Definisi tugas Amazon ECS
- Set tugas Amazon ECS
- Klaster-klaster Amazon ECS
- Instans kontainer Amazon ECS
- Penyedia kapasitas Amazon ECS

Amazon ECS memiliki opsi untuk menggunakan otorisasi penandaan untuk pembuatan sumber daya. Ketika Akun AWS dikonfigurasi untuk menandai otorisasi, pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, seperti. `ecsCreateCluster` Jika Anda menentukan tag dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan untuk memverifikasi apakah pengguna atau peran memiliki izin untuk membuat tag. Oleh karena itu, Anda harus memberikan izin eksplisit untuk menggunakan tindakan. `ecs:TagResource` Untuk informasi selengkapnya, lihat the section called “Memberi tanda pada sumber daya saat sumber daya dibuat”. Untuk informasi tentang cara mengkonfigurasi opsi, lihatthe section called “Otorisasi penandaan”.

Pembatasan

Pembatasan berikut berlaku untuk tag:

- Maksimal 50 tag dapat dikaitkan dengan sumber daya.
- Kunci tag tidak dapat diulang untuk satu sumber daya. Setiap kunci tag harus unik, dan hanya dapat memiliki satu nilai.
- Panjang tombol dapat mencapai 128 karakter di UTF-8.
- Nilai dapat mencapai 256 karakter di UTF-8.
- Jika beberapa Layanan AWS dan sumber daya menggunakan skema penandaan Anda, batasi jenis karakter yang Anda gunakan. Beberapa layanan mungkin memiliki batasan pada karakter yang diizinkan. Umumnya, karakter yang diizinkan adalah huruf, angka, spasi, dan karakter berikut: **+ - = . _ : /@**.
- Kunci dan nilai tanda peka huruf besar-kecil.
- Anda tidak dapat menggunakan `aws:AWS:`,, atau kombinasi huruf besar atau kecil seperti awalan untuk kunci atau nilai. Ini hanya disediakan untuk AWS digunakan. Anda tidak dapat menyunting atau menghapus kunci atau nilai tag dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap tags-per-resource batas Anda.

Tag yang dikelola Amazon ECS

Saat Anda menggunakan tag yang dikelola Amazon ECS, Amazon ECS secara otomatis menandai semua tugas yang baru diluncurkan dan volume Amazon EBS apa pun yang dilampirkan ke tugas dengan informasi klaster dan tag definisi tugas yang ditambahkan pengguna atau tag layanan. Berikut ini menjelaskan tag yang ditambahkan:

- Tugas mandiri — tag dengan Key as `aws:ecs:clusterName` dan Value yang disetel ke nama cluster. Semua tag definisi tugas yang ditambahkan oleh pengguna. Volume Amazon EBS yang dilampirkan ke tugas mandiri akan menerima tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster. Untuk informasi selengkapnya tentang penandaan volume Amazon EBS, lihat Menandai volume Amazon EBS.
- Tugas yang merupakan bagian dari layanan — tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster. Tag dengan Key as `aws:ecs:serviceName` dan Value diatur ke nama layanan. Tag dari salah satu sumber berikut:
  - Definisi tugas - Semua tag definisi tugas yang ditambahkan oleh pengguna.

- Layanan — Semua tag layanan yang ditambahkan oleh pengguna.

  Volume Amazon EBS yang dilampirkan ke tugas yang merupakan bagian dari layanan akan menerima tag dengan Kunci sebagai `aws:ecs:clusterName` dan Nilai yang disetel ke nama cluster, dan tag dengan Kunci sebagai `aws:ecs:serviceName` dan Nilai yang disetel ke nama layanan. Untuk informasi selengkapnya tentang penandaan volume Amazon EBS, lihat Menandai volume Amazon EBS.

Opsi berikut diperlukan untuk fitur ini:

- Anda harus ikut serta dalam format Amazon Resource Name (ARN) dan resource identifier (ID) yang baru. Untuk informasi selengkapnya, lihat Nama Sumber Daya Amazon (ARNs) dan IDs.
- Bila Anda menggunakan APIs untuk membuat layanan atau menjalankan tugas, Anda harus mengatur `enableECSManagedTags` `true` untuk `run-task` dan`create-service`. Untuk informasi selengkapnya, lihat create-service dan run-task di Referensi API.AWS Command Line Interface
- Amazon ECS menggunakan tag terkelola untuk menentukan kapan beberapa fitur diaktifkan, misalnya Auto Scaling cluster. Kami menyarankan Anda untuk tidak memodifikasi tag secara manual sehingga Amazon ECS dapat mengelola fitur secara efektif.

Gunakan tag untuk penagihan

AWS menyediakan alat pelaporan yang disebut Cost Explorer yang dapat Anda gunakan untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda.

Anda dapat menggunakan Cost Explorer untuk melihat bagan penggunaan dan biaya Anda. Anda dapat melihat data dari 13 bulan terakhir, dan memperkirakan berapa banyak kemungkinan Anda akan menghabiskan untuk tiga bulan ke depan. Anda dapat menggunakan Cost Explorer untuk melihat pola terkait berapa banyak sumber daya AWS yang Anda belanjakan dari waktu ke waktu. Misalnya, Anda dapat menggunakannya untuk mengidentifikasi area yang memerlukan penyelidikan lebih lanjut dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu menurut hari atau bulan.

Anda dapat menggunakan tag yang dikelola Amazon ECS atau tag yang ditambahkan pengguna untuk Laporan Biaya dan Penggunaan Anda. Untuk informasi selengkapnya, lihat Laporan penggunaan Amazon ECS.

Untuk melihat biaya sumber daya gabungan, Anda dapat mengatur informasi penagihan berdasarkan sumber daya yang memiliki nilai kunci tanda yang sama. Misalnya, Anda dapat memberi tanda pada beberapa sumber daya dengan nama aplikasi tertentu, lalu organisir informasi penagihan Anda untuk melihat total biaya aplikasi tersebut pada beberapa layanan. Untuk informasi selengkapnya tentang pengaturan laporan alokasi biaya dengan tanda, lihat Laporan Alokasi Biaya Bulanan dalam Panduan Pengguna AWS Billing .

Selain itu, Anda dapat mengaktifkan Data Alokasi Biaya Pisahkan untuk mendapatkan CPU tingkat tugas dan data penggunaan memori di Laporan Biaya dan Penggunaan Anda. Untuk informasi selengkapnya, lihat Laporan Biaya dan Penggunaan Tingkat Tugas.

> **Note**
>
> Jika Anda mengaktifkan pelaporan, diperlukan waktu hingga 24 jam sebelum data untuk bulan berjalan tersedia untuk dilihat.

Menambahkan tag ke sumber daya Amazon ECS

Anda dapat menandai tugas, layanan, definisi tugas, atau cluster baru atau yang sudah ada. Untuk informasi selengkapnya tentang menandai instans Anda, lihat Menambahkan tag ke instans penampung Amazon ECS.

> **Warning**
>
> Jangan menambahkan informasi pengenal pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak AWS layanan, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan dalam data sensitif atau privat.

Anda dapat menggunakan sumber daya berikut untuk menentukan tag saat Anda membuat sumber daya.

Tugas	Konsol	AWS CLI	Tindakan API
Jalankan satu atau beberapa tugas.	Menjalankan aplikasi sebagai	jalankan-tugas	RunTask

Tugas	Konsol	AWS CLI	Tindakan API
	tugas Amazon ECS		
Buat sebuah layanan.	Membuat penyebaran pembaruan bergulir Amazon ECS	membuat-layanan	CreateService
Buat satu set tugas.	Menyebarkan layanan Amazon ECS menggunakan pengontrol pihak ketiga	create-task-set	CreateTaskSet
Daftarkan ketentuan tugas.	the section called “Membuat definisi tugas menggunakan konsol”	register-task-definition	RegisterTaskDefinition
Buat sebuah klaster.	Membuat cluster Amazon ECS untuk jenis peluncuran Fargate	buat-cluster	CreateCluster
Jalankan satu atau beberapa instans kontainer.	Meluncurkan instans penampung Amazon ECS Linux	run-instance	RunInstances

Menambahkan tag ke sumber daya yang ada (konsol Amazon ECS)

Anda dapat menambahkan atau menghapus tag yang terkait dengan cluster, layanan, tugas, dan definisi tugas langsung dari halaman sumber daya.

Untuk memodifikasi tag untuk sumber daya individu

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Dari bilah navigasi, pilih yang Wilayah AWS akan digunakan.
3. Di panel navigasi, pilih tipe sumber daya (misalnya, Klaster).
4. Pilih sumber daya dari daftar sumber daya, pilih tab Tag, lalu pilih Kelola tag.
5. Konfigurasikan tag Anda.

   [Tambahkan tag] Pilih Tambah tag, lalu lakukan hal berikut:

   - Untuk Kunci, masukkan nama kunci.
   - Untuk Nilai, masukkan nilai kunci.
6. Pilih Simpan.

Menambahkan tag ke sumber daya yang ada (AWS CLI)

Anda dapat menambahkan atau menimpa satu atau beberapa tag dengan menggunakan AWS CLI atau API.

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

- AWS CLI - tag-sumber daya
- Tindakan API - TagResource

Menambahkan tag ke instans penampung Amazon ECS

Anda dapat mengaitkan tanda dengan instans kontainer menggunakan salah satu metode berikut:

- Metode 1 - Saat membuat instance container menggunakan Amazon EC2 API, CLI, atau konsol, tentukan tag dengan meneruskan data pengguna ke instance menggunakan parameter konfigurasi agen penampung. `ECS_CONTAINER_INSTANCE_TAGS` Ini membuat tag yang terkait dengan instance container di Amazon ECS saja, tag tersebut tidak dapat dicantumkan menggunakan Amazon EC2 API. Untuk informasi selengkapnya, lihat Bootstrapping instance container Amazon ECS Linux untuk meneruskan data.

  > **Important**
  >
  > Jika Anda meluncurkan instance container menggunakan grup EC2 Auto Scaling Amazon, Anda harus menggunakan parameter konfigurasi agen ECS_CONTAINER_INSTANCE_TAGS untuk menambahkan tag. Ini karena cara tag ditambahkan ke EC2 instans Amazon yang diluncurkan menggunakan grup Auto Scaling.

  Berikut ini adalah contoh skrip data pengguna yang mengaitkan tag dengan instance container Anda:

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  EOF
  ```

- Metode 2 - Saat Anda membuat instance container menggunakan Amazon EC2 API, CLI, atau konsol, tentukan tag terlebih dahulu menggunakan parameter tersebut`TagSpecification.N`. Kemudian, berikan data pengguna ke instance dengan menggunakan parameter konfigurasi agen kontainer`ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM`. Melakukannya menyebarkannya dari Amazon EC2 ke Amazon ECS.

  Berikut ini adalah contoh skrip data pengguna yang menyebarkan tag yang terkait dengan EC2 instance Amazon, dan mendaftarkan instance dengan cluster yang diberi nama. `MyCluster`

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

Untuk menyediakan akses agar tag instans penampung dapat menyebar dari Amazon EC2 ke Amazon ECS, tambahkan izin berikut secara manual sebagai kebijakan sebaris ke peran IAM instans penampung Amazon ECS. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `ec2:DescribeTags`

Berikut ini adalah contoh kebijakan yang digunakan untuk menambahkan izin ini.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
             "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh kontainer eksternal

Anda dapat mengaitkan tag dengan instance kontainer eksternal Anda dengan menggunakan salah satu metode berikut.

- Metode 1 — Sebelum menjalankan skrip instalasi untuk mendaftarkan instans eksternal Anda dengan cluster Anda, buat atau edit file konfigurasi agen penampung Amazon ECS di `/etc/ecs/ecs.config` dan tambahkan parameter konfigurasi agen `ECS_CONTAINER_INSTANCE_TAGS` penampung. Metode ini membuat tanda yang terkait dengan instans eksternal.

  Berikut ini adalah contoh sintaks.

```
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
```

- Metode 2 - Setelah instance eksternal Anda terdaftar ke cluster Anda, Anda dapat menggunakan AWS Management Console untuk menambahkan tag. Untuk informasi selengkapnya, lihat Menambahkan tag ke sumber daya yang ada (konsol Amazon ECS).

Laporan penggunaan Amazon ECS

AWS menyediakan alat pelaporan yang disebut Cost Explorer yang dapat Anda gunakan untuk menganalisis biaya dan penggunaan sumber daya Amazon ECS Anda.

Anda dapat menggunakan Cost Explorer untuk melihat bagan penggunaan dan biaya Anda. Anda dapat melihat data dari 13 bulan terakhir, dan memperkirakan berapa banyak kemungkinan Anda akan menghabiskan untuk tiga bulan ke depan. Anda dapat menggunakan Cost Explorer untuk melihat pola terkait berapa banyak sumber daya AWS yang Anda belanjakan dari waktu ke waktu. Misalnya, Anda dapat menggunakannya untuk mengidentifikasi area yang memerlukan penyelidikan lebih lanjut dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu menurut hari atau bulan.

Data pengukuran dalam Laporan Biaya dan Penggunaan menunjukkan penggunaan di semua tugas Amazon ECS Anda. Data pengukuran meliputi penggunaan CPU sebagai `vCPU-Hours` dan penggunaan memori sebagai `GB-Hours` untuk setiap tugas yang dijalankan. Bagaimana data tersebut disajikan tergantung pada tipe peluncuran tugas.

Untuk tugas yang menggunakan tipe peluncuran Fargate, `lineItem/Operation` kolom menunjukkan `FargateTask` dan Anda akan melihat biaya yang terkait dengan setiap tugas.

Untuk tugas yang menggunakan tipe EC2 peluncuran, `lineItem/Operation` kolom ditampilkan `ECSTask-EC2` dan tugas tidak memiliki biaya langsung yang terkait dengannya. Data pengukuran yang ditampilkan dalam laporan, seperti penggunaan memori, mewakili total sumber daya yang dicadangkan tugas selama periode penagihan yang Anda tentukan. Anda dapat menggunakan data ini untuk menentukan biaya cluster EC2 instans Amazon yang mendasari Anda. Data biaya dan penggunaan untuk EC2 instans Amazon Anda terdaftar secara terpisah di bawah EC2 layanan Amazon.

Anda juga dapat menggunakan tag terkelola Amazon ECS untuk mengidentifikasi layanan atau klaster yang dimiliki setiap tugas. Untuk informasi selengkapnya, lihat Gunakan tag untuk penagihan.

> **Important**
>
> Data pengukuran hanya dapat dilihat untuk tugas yang diluncurkan pada atau setelah 16 November 2018. Tugas yang diluncurkan sebelum tanggal ini tidak menampilkan data pengukuran.

Berikut ini adalah contoh dari beberapa bidang yang dapat Anda gunakan untuk mengurutkan data alokasi biaya di Cost Explorer.

- Nama klaster
- Nama layanan
- Tanda sumber daya
- Jenis peluncuran
- Wilayah AWS
- Jenis penggunaan

Untuk informasi selengkapnya tentang membuat Laporan AWS Biaya dan Penggunaan, lihat Laporan AWS Biaya dan Penggunaan di Panduan AWS Billing Pengguna.

Laporan Biaya dan Penggunaan Tingkat Tugas

AWS Cost Management dapat menyediakan data penggunaan CPU dan memori di AWS Cost and Usage Report untuk setiap tugas di Amazon ECS, termasuk tugas di Fargate dan tugas di. EC2 Data ini disebut Split Cost Allocation Data. Anda dapat menggunakan data ini untuk menganalisis biaya dan penggunaan untuk aplikasi. Selain itu, Anda dapat membagi dan mengalokasikan biaya ke unit bisnis individu dan tim dengan tag alokasi biaya dan kategori biaya. Untuk informasi selengkapnya tentang Data Alokasi Biaya Pisahkan, lihat Memahami data alokasi biaya terpisah di AWS Cost and Usage Report Panduan Pengguna.

Anda dapat ikut serta dalam Data Alokasi Biaya Split tingkat tugas untuk akun di. AWS Cost Management Console Jika Anda memiliki akun manajemen (pembayar), Anda dapat ikut serta dari akun pembayar untuk menerapkan konfigurasi ini ke setiap akun yang ditautkan.

Setelah Anda mengatur Data Alokasi Biaya Split, akan ada kolom tambahan di bawah splitLineItemheader dalam laporan. Untuk informasi selengkapnya lihat Membagi detail item baris di Panduan AWS Cost and Usage Report Pengguna

Untuk tugas EC2, data ini membagi biaya EC2 instans berdasarkan penggunaan sumber daya atau reservasi dan sumber daya yang tersisa pada instance.

Berikut ini adalah prasyarat:

- Setel parameter konfigurasi agen `ECS_DISABLE_METRICS` Amazon ECS ke`false`.

  Saat pengaturan ini`false`, agen Amazon ECS mengirimkan metrik ke Amazon. CloudWatch Di Linux, pengaturan ini secara `false` default dan metrik dikirim ke CloudWatch. Di Windows, pengaturan ini secara `true` default, jadi Anda harus mengubah pengaturan `false` untuk mengirim metrik CloudWatch AWS Cost Management untuk digunakan. Untuk informasi selengkapnya tentang konfigurasi agen ECS, lihatKonfigurasi agen kontainer Amazon ECS.
- Versi Docker minimum untuk metrik yang andal adalah Docker versi v20.10.13 dan yang lebih baru, yang disertakan dalam AMI 20220607 yang dioptimalkan Amazon ECS dan yang lebih baru.

Untuk menggunakan Split Cost Allocation Data, Anda harus membuat laporan, dan pilih Split data alokasi biaya. Untuk informasi selengkapnya, lihat Membuat Laporan Biaya dan Penggunaan di Panduan AWS Cost and Usage Report Pengguna.

AWS Cost Management menghitung Data Alokasi Biaya Split dengan CPU tugas dan penggunaan memori. AWS Cost Management dapat menggunakan CPU tugas dan reservasi memori alih-alih penggunaan, jika penggunaan tidak tersedia. Jika Anda melihat CUR menggunakan reservasi, periksa apakah instance kontainer Anda memenuhi prasyarat dan metrik penggunaan sumber daya tugas muncul. CloudWatch

Memantau Amazon ECS

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon ECS dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Sebelum Anda mulai memantau Amazon ECS, buat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Metrik yang tersedia bergantung pada tipe peluncuran tugas dan layanan dalam klaster Anda. Jika Anda menggunakan jenis peluncuran Fargate untuk layanan Anda, maka metrik pemanfaatan CPU dan memori disediakan untuk membantu pemantauan layanan Anda. Untuk jenis EC2 peluncuran Amazon, Anda memiliki dan perlu memantau EC2 instance yang membuat infrastruktur dasar Anda. Metrik reservasi dan pemanfaatan CPU dan memori tambahan tersedia di cluster, layanan, dan tugas.

Langkah selanjutnya adalah menetapkan dasar untuk kinerja Amazon ECS normal di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau Amazon ECS, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan batas dasar, setidaknya Anda harus memantau item berikut:

- Metrik reservasi dan pemanfaatan CPU dan memori untuk kluster Amazon ECS Anda
- Metrik pemanfaatan CPU dan memori untuk layanan Amazon ECS Anda

  Untuk informasi selengkapnya, lihat Melihat metrik Amazon ECS.

Praktik terbaik untuk memantau Amazon ECS

Gunakan praktik terbaik berikut untuk memantau Amazon ECS.

- Jadikan pemantauan sebagai prioritas untuk mengatasi masalah kecil sebelum menjadi masalah besar
- Buat rencana pemantauan yang mencakup jawaban atas pertanyaan berikut
  - Apa saja sasaran pemantauan Anda?
  - Sumber daya apa yang akan Anda pantau?
  - Seberapa sering Anda akan memantau sumber daya ini?
  - Alat pemantauan apa yang akan Anda gunakan?
  - Siapa yang akan melakukan tugas pemantauan?
  - Siapa yang harus diberi tahu saat terjadi kesalahan?
- Otomatiskan pemantauan sebanyak mungkin.
- Periksa file log Amazon ECS. Untuk informasi selengkapnya, lihat Melihat log agen kontainer Amazon ECS.
- Gunakan Runtime Monitoring untuk membantu melindungi akun, container, beban kerja, dan data di lingkungan Anda AWS . Untuk informasi selengkapnya, lihat Identifikasi perilaku yang tidak sah menggunakan Runtime Monitoring.

Alat pemantauan untuk Amazon ECS

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Amazon ECS. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Alat pemantauan otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton Amazon ECS dan melaporkan ketika ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik

Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon Auto EC2 Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch.

Untuk layanan dengan tugas yang menggunakan tipe peluncuran Fargate, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil tugas dalam layanan Anda berdasarkan CloudWatch metrik, seperti CPU dan pemanfaatan memori. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.

Untuk cluster dengan tugas atau layanan yang menggunakan tipe EC2 peluncuran, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil instance container berdasarkan CloudWatch metrik, seperti reservasi memori cluster.

Untuk instans penampung yang diluncurkan dengan Amazon Linux AMI Amazon ECS yang dioptimalkan Amazon, Anda dapat CloudWatch menggunakan Log untuk melihat log yang berbeda dari instans penampung Anda di satu lokasi yang nyaman. Anda harus menginstal CloudWatch agen pada instance kontainer Anda. Untuk informasi selengkapnya, lihat Mengunduh dan mengonfigurasi CloudWatch agen menggunakan baris perintah di Panduan CloudWatch Pengguna Amazon. Anda juga harus menambahkan `ECS-CloudWatchLogs` kebijakan ke `ecsInstanceRole` peran tersebut. Untuk informasi selengkapnya, lihat Memantau izin instance kontainer.

- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log dari container di tugas Amazon ECS Anda dengan menentukan driver `awslogs` log dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke CloudWatch .

  Anda juga dapat memantau, menyimpan, dan mengakses sistem operasi dan file log agen penampung Amazon ECS dari instans penampung Amazon ECS Anda. Metode untuk mengakses log ini dapat digunakan pada kontainer menggunakan tipe peluncuran EC2.

- CloudWatch Acara Amazon — Cocokkan acara dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi lebih lanjut, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge di panduan ini dan EventBridge merupakan evolusi CloudWatch Acara Amazon di Panduan EventBridge Pengguna Amazon.

- Wawasan Kontainer — Kumpulkan, agregat, dan rangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Wawasan Kontainer mengumpulkan data sebagai peristiwa log performa dengan menggunakan format metrik tersemat. Peristiwa log kinerja ini adalah entri

yang menggunakan skema JSON terstruktur yang memungkinkan data kardinalitas tinggi untuk dicerna dan disimpan dalam skala besar. Dari data ini, CloudWatch buat metrik agregat di tingkat klaster, tugas, dan layanan sebagai CloudWatch metrik. Metrik yang dikumpulkan Container Insights tersedia di dasbor CloudWatch otomatis, dan juga dapat dilihat di bagian Metrik konsol. CloudWatch

- AWS CloudTrail Pemantauan log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat Log panggilan Amazon ECS API menggunakan AWS CloudTrail di panduan ini, dan Bekerja dengan File CloudTrail Log di Panduan AWS CloudTrail Pengguna.
- Runtime Monitoring — Mendeteksi ancaman untuk cluster dan kontainer dalam lingkungan Anda AWS . Runtime Monitoring menggunakan agen GuardDuty keamanan yang menambahkan visibilitas runtime ke beban kerja Amazon ECS individual, misalnya, akses file, eksekusi proses, dan koneksi jaringan.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan Amazon ECS melibatkan pemantauan secara manual item-item yang tidak CloudWatch tercakup oleh alarm. Dasbor AWS konsol CloudWatch Trusted Advisor,, dan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda untuk memeriksa file log pada instans kontainer Anda dan kontainer dalam tugas Anda.

- Konsol Amazon ECS:
  - Metrik klaster untuk jenis EC2 peluncuran
  - Metrik Layanan
  - Status kesehatan layanan
  - Acara penyebaran layanan
- CloudWatch halaman rumah:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Buat dasbor yang disesuaikan untuk memantau layanan yang Anda pedulikan.
  - Data metrik grafik untuk memecahkan masalah dan menemukan tren.

  - Cari dan telusuri semua metrik AWS sumber daya Anda.
  - Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah.
- Pemeriksaan kesehatan kontainer - Ini adalah perintah yang berjalan secara lokal pada wadah dan memvalidasi kesehatan dan ketersediaan aplikasi. Anda mengonfigurasi ini per kontainer dalam definisi tugas Anda.
- AWS Trusted Advisor dapat membantu Anda memantau AWS sumber daya Anda untuk meningkatkan kinerja, keandalan, keamanan, dan efektivitas biaya. Empat Trusted Advisor cek tersedia untuk semua pengguna; lebih dari 50 cek tersedia untuk pengguna dengan paket dukungan Bisnis atau Perusahaan. Untuk informasi selengkapnya, lihat AWS Trusted Advisor.

  Trusted Advisor memiliki pemeriksaan ini yang berhubungan dengan Amazon ECS:

  - Toleransi kesalahan yang menunjukkan bahwa Anda memiliki layanan yang berjalan di satu Availability Zone.
  - Toleransi kesalahan yang menunjukkan bahwa Anda belum menggunakan strategi penempatan spread untuk beberapa Availability Zone.
- AWS Compute Optimizer adalah layanan yang menganalisis konfigurasi dan metrik pemanfaatan sumber daya Anda. AWS Ini melaporkan apakah sumber daya Anda optimal, dan menghasilkan rekomendasi pengoptimalan untuk mengurangi biaya dan meningkatkan kinerja beban kerja Anda.

  Untuk informasi selengkapnya, lihat AWS Compute Optimizer rekomendasi untuk Amazon ECS.

Pantau Amazon ECS menggunakan CloudWatch

Anda dapat memantau sumber daya Amazon ECS menggunakan Amazon CloudWatch, yang mengumpulkan dan memproses data mentah dari Amazon ECS menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini dicatat untuk jangka waktu dua minggu sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja cluster atau layanan Anda. Data metrik Amazon ECS secara otomatis dikirim ke CloudWatch dalam periode 1 menit. Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

Amazon ECS menyediakan metrik gratis untuk cluster dan layanan. Dengan biaya tambahan, Anda dapat mengaktifkan Amazon ECS CloudWatch Container Insights untuk klaster Anda untuk metrik per tugas, termasuk pemanfaatan sistem file CPU, memori, dan EBS. Untuk informasi selengkapnya tentang Wawasan Kontainer ini, silakan lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan CloudWatch metrik Amazon ECS.

- Setiap layanan Amazon ECS yang dihosting di Fargate CloudWatch memiliki metrik pemanfaatan CPU dan memori secara otomatis, jadi Anda tidak perlu mengambil langkah manual apa pun.

- Untuk tugas atau layanan Amazon ECS yang dihosting di instans Amazon, EC2 EC2 instans Amazon memerlukan versi `1.4.0` atau yang lebih baru (Linux) atau `1.0.0` atau yang lebih baru (Windows) dari agen penampung untuk CloudWatch metrik yang akan dihasilkan. Namun, kami menyarankan untuk menggunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS.
- Versi Docker minimum untuk CloudWatch metrik yang andal adalah versi Docker dan yang lebih baru`20.10.13`.
- EC2 Instans Amazon Anda juga memerlukan `ecs:StartTelemetrySession` izin pada peran IAM yang Anda gunakan untuk meluncurkan EC2 instans Amazon. Jika Anda membuat peran IAM instans penampung Amazon ECS sebelum CloudWatch metrik tersedia untuk Amazon ECS, Anda mungkin perlu menambahkan izin ini. Untuk informasi tentang peran IAM instance container dan melampirkan kebijakan IAM terkelola untuk instance container, lihat. Peran IAM instance wadah Amazon ECS
- Anda dapat menonaktifkan pengumpulan CloudWatch metrik di EC2 instans Amazon dengan menyetel konfigurasi agen `ECS_DISABLE_METRICS=true` penampung Amazon ECS Anda. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Metrik-metrik yang direkomendasikan

Amazon ECS menyediakan CloudWatch metrik gratis yang dapat Anda gunakan untuk memantau sumber daya Anda. Reservasi CPU dan memori serta pemanfaatan sistem file CPU, memori, dan EBS di seluruh cluster Anda secara keseluruhan, dan pemanfaatan sistem file CPU, memori, dan EBS pada layanan di cluster Anda dapat diukur menggunakan metrik ini. Untuk beban kerja GPU, Anda dapat mengukur reservasi GPU di klaster Anda.

Infrastruktur tugas Amazon ECS Anda di-host di cluster Anda menentukan metrik mana yang tersedia. Untuk tugas yang dihosting di infrastruktur Fargate, Amazon ECS menyediakan metrik pemanfaatan sistem file CPU, memori, dan EBS untuk membantu pemantauan layanan Anda. Untuk

tugas yang dihosting pada EC2 instans, Amazon ECS menyediakan metrik reservasi CPU, memori, dan GPU serta metrik pemanfaatan CPU dan memori di tingkat klaster dan layanan. Anda perlu memantau EC2 instans Amazon yang membuat infrastruktur dasar Anda secara terpisah. Untuk informasi selengkapnya tentang memantau EC2 instans Amazon Anda, lihat Memantau Amazon EC2 di Panduan EC2 Pengguna Amazon.

Untuk informasi tentang alarm yang disarankan untuk digunakan dengan Amazon ECS, lihat salah satu hal berikut di Panduan Pengguna Amazon CloudWatch Logs:

- Amazon ECS
- Amazon ECS dengan Wawasan Kontainer

Melihat metrik Amazon ECS

Setelah sumber daya berjalan di klaster, Anda dapat melihat metrik di Amazon ECS dan CloudWatch konsol. Konsol Amazon ECS menyediakan tampilan maksimum, minimum, dan rata-rata 24 jam untuk metrik klaster dan layanan Anda. Konsol CloudWatch menyediakan tampilan sumber daya yang detail dan disesuaikan, serta jumlah tugas yang berjalan dalam layanan.

Konsol Amazon ECS

Amazon ECS service CPU dan metrik pemanfaatan memori tersedia di konsol Amazon ECS. Tampilan yang disediakan untuk metrik klaster menunjukkan nilai rata-rata, minimum, dan maksimum dalam periode 24 jam sebelumnya, dengan titik data yang tersedia dalam interval 5 menit. Untuk informasi selengkapnya, lihat Metrik pemanfaatan layanan Amazon ECS.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pilih klaster yang metriknya ingin Anda lihat.
3. Tentukan metrik yang akan dilihat.

Untuk melihat metrik dari	Langkah-langkah
Klaster	Pada halaman detail cluster, pilih tab Metrik. Ada juga tautan yang disediakan ke CloudWatch konsol untuk melihat metrik CloudWatc

Untuk melihat metrik dari	Langkah-langkah	
	h Wawasan Kontainer jika Anda mengaktifkannya.	
Layanan	Pada halaman detail cluster, pada tab Layanan, pilih layanan. Metrik kemudian tersedia di tab Kesehatan dan metrik.	

CloudWatch konsol

Untuk jenis peluncuran Fargate, metrik layanan Amazon ECS juga dapat dilihat di konsol. CloudWatch Konsol menyediakan tampilan metrik Amazon ECS yang paling detail, dan Anda dapat menyesuaikan tampilan sesuai dengan kebutuhan Anda. Anda dapat melihat pemanfaatan layanan dan jumlah tugas RUNNING layanan.

Untuk jenis EC2 peluncuran, cluster Amazon ECS dan metrik layanan juga dapat dilihat di CloudWatch konsol. Konsol menyediakan tampilan metrik Amazon ECS yang paling detail, dan Anda dapat menyesuaikan tampilan sesuai dengan kebutuhan Anda.

Untuk informasi tentang cara melihat metrik, lihat Melihat metrik yang tersedia di CloudWatch Panduan Pengguna Amazon.

Metrik Amazon ECS CloudWatch

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke akun Anda penggunaan sumber daya. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

Amazon ECS mengirimkan metrik ke CloudWatch interval satu menit. Metrik ini dikumpulkan untuk sumber daya yang memiliki tugas di `RUNNING` negara bagian. Jika kluster, layanan, atau sumber daya lainnya tidak memiliki tugas yang berjalan, tidak ada metrik yang akan dilaporkan untuk sumber daya tersebut selama periode tersebut. Misalnya, jika Anda memiliki klaster dengan satu layanan tetapi layanan tersebut tidak memiliki tugas dalam `RUNNING` status, tidak akan ada metrik yang dikirim ke CloudWatch. Demikian pula, jika Anda memiliki dua layanan dan salah satunya menjalankan tugas sementara yang lain tidak, hanya metrik untuk layanan dengan menjalankan tugas yang akan dikirim.

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
CPUReservation	Persentase unit CPU yang dicadangkan dalam cluster atau layanan. Reservasi CPU (disaring oleh`ClusterName`) diukur sebagai total unit CPU yang dicadangkan oleh tugas Amazon ECS di cluster, dibagi dengan total unit CPU untuk semua EC2 instans Amazon yang terdaftar di cluster. Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi CPU. Metrik ini hanya didukung untuk tugas yang	`ClusterName`	Rata-rata, Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	dihosting di EC2 instans Amazon.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`CPUUtilization`	Persentase unit CPU yang digunakan oleh cluster atau layanan. Pemanfaatan CPU tingkat cluster (disaring oleh`ClusterName`) diukur sebagai total unit CPU yang digunakan oleh tugas Amazon ECS di cluster, dibagi dengan total unit CPU untuk semua EC2 instance Amazon yang terdaftar di cluster. Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi CPU. Metrik tingkat klaster hanya didukung untuk tugas	`ClusterName` , `ServiceName`	Rata-rata , Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	yang dihosting di instans Amazon. EC2 Pemanfaatan CPU tingkat layanan (disaring oleh`ClusterName` ,`ServiceName`) diukur sebagai total unit CPU yang digunakan oleh tugas-tugas yang termasuk dalam layanan, dibagi dengan jumlah total unit CPU yang dicadangkan untuk tugas-tugas yang termasuk dalam layanan. Metrik tingkat layanan didukung untuk tugas yang dihosting di EC2 instans Amazon dan Fargate.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`MemoryReservation`	Persentase memori yang disimpan oleh tugas yang berjalan di klaster. Reservasi memori cluster diukur sebagai total memori yang dicadangkan oleh tugas Amazon ECS di cluster, dibagi dengan jumlah total memori untuk semua EC2 instans Amazon yang terdaftar di cluster. Metrik ini hanya dapat disaring oleh`ClusterName` . Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi memori. Metrik reservasi memori	`ClusterName` .	Rata-rata , Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	tingkat cluster hanya didukung untuk tugas yang dihosting di EC2 instans Amazon. **Note** Saat menghitung pemanfaatan memori, jika `MemoryReservation` ditentukan, itu digunakan dalam perhitungan alih-alih memori total.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`MemoryUtilization`	Persentase memori yang digunakan oleh cluster atau layanan. Pemanfaatan memori tingkat cluster (disaring oleh`ClusterName`) diukur sebagai total memori yang digunakan oleh tugas Amazon ECS di cluster, dibagi dengan total memori untuk semua EC2 instans Amazon yang terdaftar di cluster. Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik pemanfaatan memori. Metrik tingkat klaster hanya didukung	`ClusterName` , `ServiceName`	Rata-rata , Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	untuk tugas yang dihosting di instans Amazon. EC2 Pemanfaatan memori tingkat layanan (disaring oleh`ClusterName` ,`ServiceName`) diukur sebagai memori total yang digunakan oleh tugas-tugas milik layanan, dibagi dengan total memori yang disediakan untuk tugas-tugas yang termasuk dalam layanan. Metrik tingkat layanan didukung untuk tugas yang dihosting di EC2 instans Amazon dan Fargate.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`EBSFilesystemUtilization`	Persentase sistem file Amazon EBS yang digunakan oleh tugas dalam suatu layanan. Metrik pemanfaatan sistem file EBS tingkat layanan (disaring oleh`ClusterName` ,`ServiceName`) diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas-tugas yang termasuk dalam layanan, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas yang termasuk dalam layanan. Metrik pemanfaatan sistem file EBS tingkat layanan	`ClusterName` , `ServiceName`	Rata-rata , Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	hanya tersedia untuk tugas yang dihosting di EC2 instans Amazon (mengguna kan versi agen kontainer) `1.79.0` dan Fargate (menggunakan versi `1.4.0` platform) yang memiliki volume EBS yang terpasang. **Note** Untuk tugas yang di-host di Fargate, ada ruang pada disk yang hanya digunakan oleh Fargate. Tidak ada			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	biaya yang terkait dengan ruang yang digunakan Fargate, tetapi Anda akan melihat penyimpanan tambahan ini menggunakan alat seperti. `df`			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`GPUReservation`	Persentase total GPUs yang tersedia yang dicadangkan dengan menjalankan tugas di cluster. Metrik reservasi GPU tingkat cluster diukur sebagai jumlah tugas Amazon ECS yang GPUs dicadangkan di cluster, dibagi dengan jumlah total GPUs yang tersedia di semua EC2 instans Amazon yang GPUs terdaftar di cluster. Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik reservasi GPU.	`ClusterName`	Rata-rata, Minimum, Maksimum	Persen

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`ActiveConnectionCount`	Jumlah total koneksi bersamaan yang aktif dari klien ke proxy Amazon ECS Service Connect yang berjalan dalam tugas yang berbagi yang dipilih. `DiscoveryName` Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect. Dimensi yang valid: `DiscoveryName` dan`DiscoveryName`, `ServiceName`, `ClusterName` .	`DiscoveryName` dan `DiscoveryName ServiceName` , `ClusterName`	Rata-rata , Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`NewConnectionCount`	Jumlah total koneksi baru yang dibuat dari klien ke proxy Amazon ECS Service Connect yang berjalan dalam tugas yang berbagi yang dipilih. `DiscoveryName` Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`DiscoveryName` dan `DiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan
`ProcessedBytes`	Jumlah total byte lalu lintas masuk yang diproses oleh proxy Service Connect. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`DiscoveryName` dan `DiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Byte

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`RequestCount`	Jumlah permintaan lalu lintas masuk yang diproses oleh proxy Service Connect. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect. Anda juga perlu mengonfigurasi `appProtocol` pemetaan port dalam definisi tugas Anda.	`DiscoveryName` dan `DiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`GrpcRequestCount`	Jumlah permintaan lalu lintas gRPC masuk yang diproses oleh proxy Service Connect. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` berada `GRPC` di pemetaan port dalam definisi tugas.	`DiscoveryName` dan `DiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`HTTPCode_Target_2XX_Count`	Jumlah kode respons HTTP dengan angka 200 hingga 299 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	dalam definisi tugas. Dimensi yang valid:.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`HTTPCode_Target_3XX_Count`	Jumlah kode respons HTTP dengan angka 300 hingga 399 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	dalam definisi tugas.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`HTTPCode_Target_4XX_Count`	Jumlah kode respons HTTP dengan angka 400 hingga 499 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	dalam definisi tugas.			

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`HTTPCode_Target_5XX_Count`	Jumlah kode respons HTTP dengan angka 500 hingga 599 yang dihasilkan oleh aplikasi dalam tugas-tugas ini. Tugas-tugas ini adalah targetnya. Metrik ini hanya menghitung respons yang dikirim ke proxy Service Connect oleh aplikasi dalam tugas ini, bukan tanggapan yang dikirim secara langsung. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect dan `appProtocol` sedang `HTTP` atau `HTTP2` dalam pemetaan port	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Hitungan

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
	dalam definisi tugas.			
`RequestCountPerTarget`	Rata-rata jumlah permintaan yang diterima oleh setiap target yang berbagi yang dipilih`DiscoveryName` . Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata	Hitungan
`TargetProcessedBytes`	Jumlah total byte yang diproses oleh proxy Service Connect. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum, Jumlah	Byte

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`TargetResponseTime`	Latensi pemrosesan permintaan aplikasi. Waktu berlalu, dalam milidetik, setelah permintaan mencapai proxy Service Connect di tugas target hingga respons dari aplikasi target diterima kembali ke proxy. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`TargetDiscoveryName` dan `TargetDiscoveryName`, `ServiceName`, `ClusterName`	Rata-rata, Minimum, Maksimum	Milidetik

Metrik	Deskripsi	Dimensi yang Valid	Statistik Berguna	Unit
`ClientTLSNegotiationErrorCount`	Jumlah total kali koneksi TLS gagal. Metrik ini hanya digunakan ketika TLS diaktifkan. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`DiscoveryName` dan `DiscoveryName ServiceName` , `ClusterName`	Rata-rata , Minimum, Maksimum, Jumlah	Hitungan
`TargetTLSNegotiationErrorCount`	Jumlah total kali koneksi TLS gagal karena sertifikat klien hilang, AWS Private CA verifikasi gagal, atau verifikasi SAN gagal. Metrik ini hanya digunakan ketika TLS diaktifkan. Metrik ini hanya tersedia jika Anda telah mengonfigurasi Amazon ECS Service Connect.	`ServiceName` ,`ClusterName` , `TargetDiscoveryName` dan `TargetDiscoveryName`	Rata-rata , Minimum, Maksimum, Jumlah	Hitungan

Dimensi untuk metrik Amazon ECS

Metrik Amazon ECS menggunakan `AWS/ECS` namespace dan menyediakan metrik untuk dimensi berikut. Amazon ECS hanya mengirimkan metrik untuk sumber daya yang memiliki tugas di negara bagian. `RUNNING` Misalnya, jika Anda memiliki sebuah klaster dengan satu layanan di dalamnya namun layanan tersebut tidak memiliki tugas tidak di status `RUNNING`, maka tidak akan ada metrik yang dikirimkan ke CloudWatch. Jika Anda memiliki dua layanan dan salah satunya memiliki tugas yang sedang berjalan sementara yang satunya tidak, hanya metrik untuk layanan dengan tugas yang sedang berjalanlah yang akan dikirim.

Dimensi	Definisi
`ClusterName`	Dimensi ini memfilter data yang Anda minta untuk semua sumber daya di klaster tertentu. Semua metrik Amazon ECS difilter oleh. `ClusterName`
`ServiceName`	Dimensi ini memfilter data yang Anda minta untuk semua sumber daya di layanan tertentu dalam klaster tertentu.
`DiscoveryName`	Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan di semua kluster Amazon ECS. Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.
`DiscoveryName, ServiceName, ClusterName`	Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect tertentu di seluruh tugas yang memiliki nama penemuan ini dan yang dibuat oleh layanan ini di klaster ini. Gunakan dimensi ini untuk melihat metrik lalu lintas masuk untuk layanan tertentu, jika Anda

Dimensi	Definisi
	telah menggunakan kembali nama penemuan yang sama di beberapa layanan di ruang nama yang berbeda. Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.
`TargetDiscoveryName`	Dimensi ini menyaring data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan di semua kluster Amazon ECS. Berbeda dari`DiscoveryName` , metrik lalu lintas ini hanya mengukur lalu lintas masuk ke ini `DiscoveryName` yang berasal dari tugas Amazon ECS lain yang memiliki konfigurasi Service Connect di namespace ini. Ini termasuk tugas yang dibuat oleh layanan dengan konfigurasi Service Connect khusus klien atau server klien. Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

Dimensi	Definisi
`TargetDiscoveryName, ServiceName, ClusterName`	Dimensi ini memfilter data yang Anda minta untuk metrik lalu lintas ke nama penemuan Service Connect yang ditentukan tetapi hanya menghitung lalu lintas dari tugas yang dibuat oleh layanan ini di klaster ini. Gunakan dimensi ini untuk melihat metrik lalu lintas masuk yang berasal dari klien tertentu di layanan lain. Berbeda dari`DiscoveryName, ServiceName, ClusterName` , metrik lalu lintas ini hanya mengukur lalu lintas masuk ke ini `DiscoveryName` yang berasal dari tugas Amazon ECS lain yang memiliki konfigurasi Service Connect di namespace ini. Ini termasuk tugas yang dibuat oleh layanan dengan konfigurasi Service Connect khusus klien atau server klien. Perhatikan bahwa port tertentu dalam wadah yang sedang berjalan dapat memiliki beberapa nama penemuan.

AWS Fargate metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke akun Anda penggunaan sumber daya. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

AWS Fargate metrik penggunaan sesuai dengan kuota AWS layanan. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota layanan Fargate, titik akhir Amazon ECS dan kuota di.. Referensi Umum Amazon Web

AWS Fargate menerbitkan metrik berikut di namespace. `AWS/Usage`

Metrik	Deskripsi
`ResourceCount`	Jumlah sumber daya tertentu yang berjalan di akun Anda. Sumber daya ditentukan oleh dimensi yang terkait dengan metrik.

Dimensi berikut ini digunakan untuk memilah penggunaan metrik yang dipublikasikan oleh AWS Fargate.

Dimensi	Deskripsi
`Service`	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan AWS Fargate , nilai untuk dimensi ini adalah `Fargate`.
`Type`	Jenis entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik AWS Fargate penggunaan adalah`Resource`.
`Resource`	Tipe sumber daya yang sedang berjalan. Tipe sumber daya yang sedang berjalan. Saat ini, satu-satunya nilai yang valid untuk metrik AWS Fargate penggunaan adalah `vCPU` yang mengembalikan informasi tentang instance yang sedang berjalan.
`Class`	Kelas sumber daya yang akan dilacak. Kelas sumber daya yang akan dilacak. Untuk metrik AWS Fargate penggunaan dengan vCPU sebagai nilai dimensi Resource, nilai `Standard/OnDemand` yang valid adalah dan. `Standard/Spot`

Anda dapat menggunakan konsol Service Quotas untuk memvisualisasikan penggunaan Anda pada grafik dan mengonfigurasi alarm yang mengingatkan Anda saat AWS Fargate penggunaan mendekati kuota layanan. Untuk informasi tentang cara membuat CloudWatch alarm untuk memberi tahu Anda saat mendekati ambang nilai kuota, lihat Service Quotas dan alarm CloudWatch Service Quotas Amazon di Panduan Pengguna

.

Metrik reservasi klaster Amazon ECS

Metrik reservasi klaster diukur sebagai persentase CPU, memori, dan GPUs yang dicadangkan oleh semua tugas Amazon ECS di klaster jika dibandingkan dengan CPU agregat, memori, dan GPUs yang terdaftar untuk setiap instans kontainer aktif di cluster. Hanya instans kontainer di status `ACTIVE` atau `DRAINING` yang akan mempengaruhi metrik reservasi klaster. Metrik ini hanya digunakan pada cluster dengan tugas atau layanan yang dihosting pada EC2 instance. Itu tidak didukung pada cluster dengan tugas yang dihosting. AWS Fargate

```
                              (Total CPU units reserved by tasks in cluster) x 100
Cluster CPU reservation =
 ---------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                  (Total MiB of memory reserved by tasks in cluster x
 100)
Cluster memory reservation =
 ------------------------------------------------------------------
                               (Total MiB of memory registered by container instances in
 cluster)
```

```
                                 (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ------------------------------------------------------------------
                               (Total GPUs registered by container instances in cluster)
```

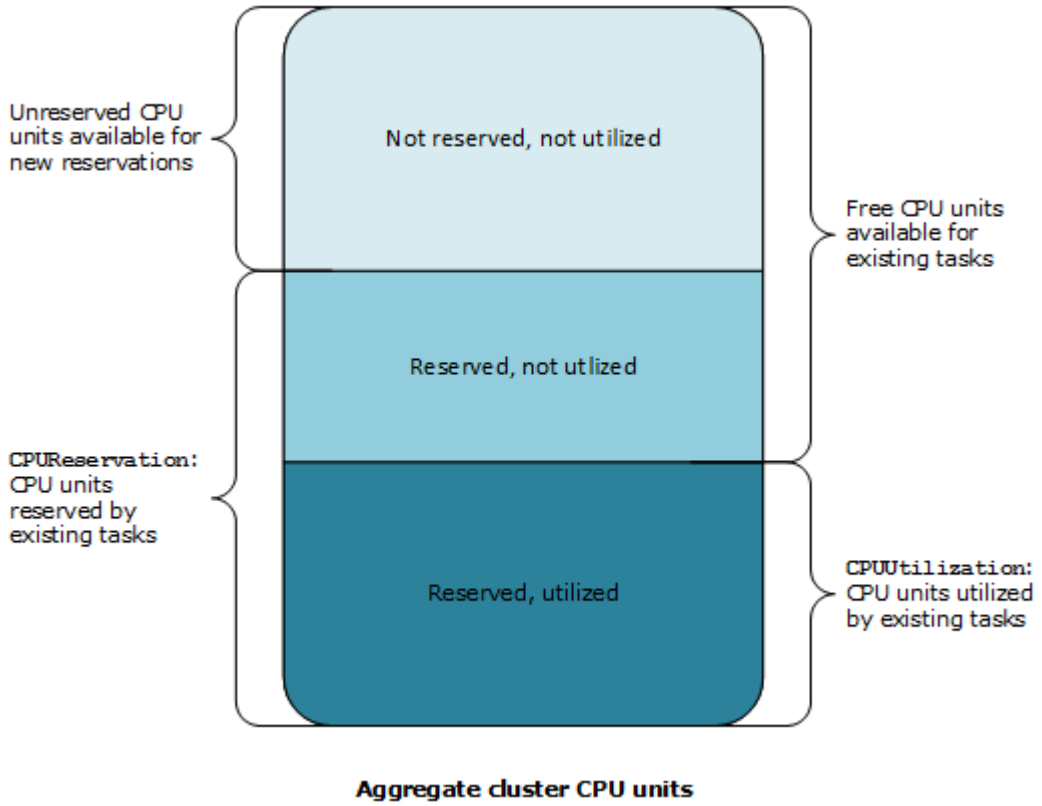
Saat Anda menjalankan tugas di klaster, Amazon ECS mem-parsing definisi tugasnya dan menyimpan unit CPU agregat, MiB memori, dan GPUs yang ditentukan dalam definisi kontainernya. Setiap menit, Amazon ECS menghitung jumlah unit CPU, MiB memori, dan GPUs yang saat ini dicadangkan untuk setiap tugas yang berjalan di cluster. Jumlah total CPU, memori, dan GPUs cadangan untuk semua tugas yang berjalan di cluster dihitung, dan angka-angka tersebut dilaporkan CloudWatch sebagai persentase dari total sumber daya terdaftar untuk cluster. Jika Anda menentukan batas lunak (`memoryReservation`) dalam definisi tugas, itu digunakan untuk menghitung jumlah memori cadangan. Jika tidak, hard limit (`memory`) akan digunakan. Total MiB memori yang dicadangkan oleh tugas dalam cluster juga mencakup ukuran volume sistem file sementara (`tmpfs`) dan `sharedMemorySize` jika didefinisikan dalam definisi tugas. Untuk informasi

selengkapnya tentang batas keras dan lunak, ukuran memori bersama, dan ukuran volume tmpfs, lihat Parameter Definisi Tugas.

Misalnya, sebuah klaster memiliki dua instans kontainer aktif yang terdaftar: instans `c4.4xlarge` dan instans `c4.large`. Instans `c4.4xlarge` mendaftar ke klaster dengan 16.384 unit CPU dan 30.158 MiB memori. Instans `c4.large` mendaftar dengan 2.048 unit CPU dan 3.768 MiB memori. Sumber daya agregat klaster ini adalah 18.432 unit CPU dan 33.926 MiB memori.

Jika suatu ketentuan tugas menyimpan 1.024 unit CPU dan 2.048 MiB memori, dan sepuluh tugas dimulai dengan ketentuan tugas di klaster ini (dan tidak ada tugas lain yang sedang berjalan), maka sejumlah 10.240 unit CPU dan 20,480 MiB memori direservasi. Ini dilaporkan CloudWatch sebagai reservasi CPU 55% dan reservasi memori 60% untuk cluster.

Ilustrasi berikut menunjukkan jumlah unit CPU yang terdaftar dalam sebuah klaster serta makna reservasi dan pemanfaatan tersebut bagi tugas yang ada dan penempatan tugas baru. Blok yang lebih rendah (Reserved, used) dan center (Reserved, not used) mewakili total unit CPU yang dicadangkan untuk tugas yang ada yang berjalan di cluster, atau `CPUReservation` CloudWatch metrik. Blok bawah mewakili unit CPU cadangan yang sebenarnya digunakan oleh tugas yang sedang berjalan di cluster, atau `CPUUtilization` CloudWatch metrik. Blok atas mewakili unit CPU yang tidak disimpan oleh tugas yang ada; unit CPU ini tersedia untuk penempatan tugas baru. Tugas yang ada juga dapat menggunakan unit CPU yang tidak tersimpan, jika kebutuhan sumber daya CPU meningkat. Untuk informasi selengkapnya, lihat dokumentasi parameter ketentuan tugas `cpu`.



Metrik pemanfaatan klaster Amazon ECS

Metrik pemanfaatan cluster tersedia untuk CPU, memori, dan, ketika ada volume EBS yang melekat pada tugas Anda, pemanfaatan sistem file EBS. Metrik ini hanya tersedia untuk klaster dengan tugas atau layanan yang dihosting di instans Amazon EC2 . Mereka tidak didukung pada cluster dengan tugas yang dihosting. AWS Fargate

Metrik pemanfaatan CPU dan memori tingkat cluster Amazon ECS

Pemanfaatan CPU dan memori diukur sebagai persentase CPU dan memori yang digunakan oleh semua tugas pada cluster jika dibandingkan dengan CPU agregat dan memori yang terdaftar untuk setiap EC2 instans Amazon aktif yang terdaftar ke cluster. Hanya EC2 instans Amazon dalam `ACTIVE` atau `DRAINING` status yang akan memengaruhi metrik pemanfaatan klaster.

```
                                (Total CPU units used by tasks in cluster) x 100
Cluster CPU utilization =
 ------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                    (Total MiB of memory used by tasks in cluster x
 100)
Cluster memory utilization =
 ------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

Setiap menit, agen penampung Amazon ECS di setiap EC2 instans Amazon menghitung jumlah unit CPU dan MiB memori yang saat ini digunakan untuk setiap tugas yang berjalan pada EC2 instans Amazon itu, dan informasi ini dilaporkan kembali ke Amazon ECS. Jumlah total CPU dan memori yang digunakan untuk semua tugas yang berjalan di cluster dihitung, dan angka-angka tersebut dilaporkan CloudWatch sebagai persentase dari total sumber daya terdaftar untuk cluster.

Misalnya, klaster memiliki dua EC2 instans Amazon aktif yang terdaftar, `c4.4xlarge` instance dan `c4.large` instance. `c4.4xlarge`Instance mendaftar ke dalam cluster dengan unit `16,384` CPU dan `30,158` MiB memori. `c4.large`Instans mendaftar dengan unit `2,048` CPU dan `3,768` MiB memori. Sumber daya agregat dari cluster ini adalah unit `18,432` CPU dan `33,926` MiB memori.

Jika sepuluh tugas berjalan pada cluster ini dan setiap tugas menghabiskan unit `1,024` CPU dan `2,048` MiB memori, `10,240` total unit CPU dan `20,480` MiB memori digunakan pada cluster. Ini dilaporkan CloudWatch sebagai 55% pemanfaatan CPU dan 60% pemanfaatan memori untuk cluster.

Pemanfaatan sistem file Amazon EBS tingkat cluster Amazon ECS

Metrik pemanfaatan sistem file EBS tingkat cluster diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas yang berjalan di cluster, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas di cluster.

```
                                       (Total GB of EBS filesystem used by tasks in
 cluster x 100)
Cluster EBS filesystem utilization =
 ------------------------------------------------------------------
                                       (Total GB of EBS filesystem allocated to tasks
 in cluster)
```

Metrik pemanfaatan layanan Amazon ECS

Metrik pemanfaatan layanan tersedia untuk CPU, memori, dan, ketika ada volume EBS yang melekat pada tugas Anda, pemanfaatan sistem file EBS. Metrik tingkat layanan didukung untuk layanan dengan tugas yang dihosting di EC2 instans Amazon dan Fargate.

Tingkat layanan CPU dan pemanfaatan memori

Pemanfaatan CPU dan memori diukur sebagai persentase CPU dan memori yang digunakan oleh tugas Amazon ECS yang termasuk dalam layanan pada cluster bila dibandingkan dengan CPU dan memori yang ditentukan dalam definisi tugas layanan.

Saat melihat metrik ini CloudWatch, Anda dapat memilih statistik yang berbeda:

- Rata-rata: Pemanfaatan rata-rata di semua tugas dalam layanan. Ini dihitung dengan menggunakan rumus di bawah ini.
- Minimum: Pemanfaatan tugas dengan penggunaan sumber daya terendah dalam layanan. Ini mewakili persentase CPU atau memori yang digunakan oleh tugas intensif sumber daya paling sedikit dibandingkan dengan apa yang ditentukan dalam definisi tugas.
- Maksimum: Pemanfaatan tugas dengan penggunaan sumber daya tertinggi dalam layanan. Ini mewakili persentase CPU atau memori yang digunakan oleh tugas yang paling intensif sumber daya dibandingkan dengan apa yang ditentukan dalam definisi tugas.

Rumus berikut menunjukkan bagaimana statistik Rata-rata dihitung:

```
                                (Total CPU units used by tasks in service) x 100
Service CPU utilization =
 ------------------------------------------------------------
                          (Total CPU units specified in the task definition) x (Number
 of tasks in the service)
```

```
                                   (Total MiB of memory used by tasks in service x
 100)
Service memory utilization =
 ------------------------------------------------------------------
                             (Total MiB of memory specified in the task definition)  x
 (Number of tasks in the service)
```

> **Note**
>
> Rumus di atas hanya berlaku untuk statistik Rata-rata. Statistik Minimum dan Maksimum mewakili tugas individu dengan pemanfaatan sumber daya terendah dan tertinggi, masing-masing, daripada perhitungan agregat di semua tugas.

Amazon ECS mengumpulkan metrik setiap 20 detik. Setiap menit, agen penampung Amazon ECS menghitung jumlah unit CPU dan MiB memori yang saat ini digunakan untuk setiap tugas yang berjalan yang dimiliki oleh layanan. Informasi ini dilaporkan kembali ke Amazon ECS. Jumlah CPU dan memori yang digunakan untuk semua tugas yang dimiliki oleh layanan yang berjalan di klaster dihitung, dan angka tersebut dilaporkan ke CloudWatch sebagai persentase jumlah sumber daya yang ditentukan untuk layanan dalam ketentuan tugas layanan. Nilai minimum dan maksimum adalah yang terkecil dan terbesar dari metrik 20 detik. Rata-rata adalah agregat dari 3 nilai.

Jika Anda menentukan soft limit (`memoryReservation`), maka ketentuan tersebut digunakan untuk menghitung jumlah memori yang disimpan. Jika tidak, hard limit (`memory`) akan digunakan. Untuk informasi lebih lanjut tentang batas keras dan lunak, lihatUkuran tugas.

Misalnya, ketentuan tugas untuk layanan menentukan total 512 unit CPU dan 1.024 MiB memori (dengan parameter hard limit `memory`) untuk semua kontainernya. Layanan ini memiliki hitungan yang diinginkan dari 1 tugas yang berjalan, layanan tersebut berjalan pada sebuah klaster dengan 1 instans kontainer `c4.large` (dengan 2.048 unit CPU dan 3.768 MiB total memori), dan tidak ada tugas lain yang berjalan di klaster. Meskipun tugas tersebut menentukan 512 unit CPU, karena itu adalah satu-satunya tugas yang berjalan pada instans kontainer dengan 2,048 unit CPU, ia dapat menggunakan hingga empat kali jumlah yang ditentukan (2,048 / 512). Namun, memori sebesar 1.024 MiB yang telah ditentukan adalah hard limit dan tidak dapat dilampaui, sehingga dalam hal ini pemanfaatan memori layanan tidak dapat melebihi 100%.

Jika contoh sebelumnya menggunakan soft limit `memoryReservation` dan bukan parameter hard limit `memory`, tugas layanan dapat menggunakan lebih dari memori yang telah ditentukan sebesar 1.024 MiB sesuai keperluan. Dalam hal ini, pemanfaatan memori layanan dapat melebihi 100%.

Jika aplikasi Anda memiliki lonjakan mendadak dalam pemanfaatan memori untuk waktu yang singkat, Anda tidak akan melihat pemanfaatan memori layanan meningkat karena Amazon ECS mengumpulkan beberapa titik data setiap menit, dan kemudian menggabungkannya ke satu titik data yang dikirim ke. CloudWatch

Jika tugas ini sedang melakukan pekerjaan dengan penggunaan CPU yang intensif selama periode tertentu dan menggunakan semua 2.048 unit CPU dan 512 MiB memori yang tersedia, maka layanan melaporkan pemanfaatan CPU 400% dan pemanfaatan memori 50%. Jika tugas idle dan menggunakan 128 CPU unit dan 128 MiB memori, maka layanan melaporkan pemanfaatan CPU 25% dan pemanfaatan memori 12.5%.

> **Note**
>
> Dalam contoh ini, pemanfaatan CPU hanya akan naik di atas 100% ketika unit CPU didefinisikan pada tingkat kontainer. Jika Anda menentukan unit CPU pada tingkat tugas, pemanfaatan tidak akan melebihi batas tingkat tugas yang ditentukan.

Pemanfaatan sistem file EBS tingkat layanan

Pemanfaatan sistem file EBS tingkat layanan diukur sebagai jumlah total sistem file EBS yang digunakan oleh tugas-tugas milik layanan, dibagi dengan jumlah total penyimpanan sistem file EBS yang dialokasikan untuk semua tugas yang termasuk dalam layanan.

Jumlah tugas layanan **`RUNNING`**

Anda dapat menggunakan CloudWatch metrik untuk melihat jumlah tugas di layanan Anda yang berada di `RUNNING` negara bagian. Misalnya, Anda dapat menyetel CloudWatch alarm untuk metrik ini untuk memberi tahu Anda jika jumlah tugas yang berjalan di layanan Anda berada di bawah nilai yang ditentukan.

Jumlah **`RUNNING`** tugas layanan di Amazon ECS CloudWatch Container Insights

Metrik “Jumlah Tugas yang Menjalankan” (`RunningTaskCount`) tersedia per klaster dan per layanan saat Anda menggunakan Amazon ECS CloudWatch Container Insights. Anda dapat menggunakan Container Insights untuk semua cluster baru yang dibuat dengan memilih masuk ke setelan `containerInsights` akun, pada kluster individual dengan mengaktifkan setelan klaster selama pembuatan klaster, atau pada cluster yang ada dengan menggunakan API. UpdateClusterSettings Metrik yang dikumpulkan oleh CloudWatch Container Insights dikenakan biaya sebagai metrik kustom. Untuk informasi lebih lanjut tentang harga CloudWatch , lihat Harga CloudWatch.

Untuk melihat metrik ini, lihat Metrik Wawasan Kontainer Amazon ECS di Panduan Pengguna Amazon CloudWatch .

Kasus penggunaan metrik pemanfaatan layanan Amazon ECS

Daftar berikut memberikan informasi tentang kapan harus menggunakan metrik Amazon ECS.

- Pemantauan pemanfaatan sumber daya: Gunakan statistik rata-rata untuk memantau pola konsumsi CPU dan memori, menetapkan garis dasar kinerja, dan mendeteksi tren penurunan kinerja secara bertahap.
- Optimalisasi biaya: Gunakan statistik rata-rata untuk memantau penggunaan sumber daya, wadah ukuran yang tepat, menyesuaikan reservasi berdasarkan pola penggunaan, dan menerapkan penskalaan terjadwal.
- Pembandingan kinerja: Gunakan statistik rata-rata untuk membandingkan metrik antara revisi layanan, menetapkan kinerja KPIs, dan memvalidasi peningkatan pengoptimalan.
- Deteksi lantai sumber daya: Gunakan statistik rata-rata untuk mengidentifikasi kebutuhan sumber daya minimum selama periode idle, mengatur reservasi yang sesuai, dan mendeteksi penurunan abnormal.
- Deteksi anomali: Gunakan statistik minimum untuk menemukan penurunan pemanfaatan sumber daya yang tidak biasa yang menunjukkan potensi masalah, seperti kegagalan inisialisasi atau periode idle yang tidak terduga.
- Penyempurnaan kebijakan penskalaan: Gunakan statistik minimum untuk menetapkan ambang batas skala optimal berdasarkan pemanfaatan minimum yang layak untuk mencegah penskalaan agresif.
- Perencanaan kapasitas: Gunakan statistik maksimum untuk mengatur ukuran tugas yang sesuai dan merencanakan infrastruktur dengan ruang kepala yang cukup untuk lonjakan lalu lintas.
- Identifikasi kemacetan kinerja: Gunakan statistik maksimum untuk mendeteksi titik saturasi sumber daya, mengidentifikasi kemacetan, dan menentukan kapan harus meningkatkan ukuran tugas.
- Konfigurasi kebijakan penskalaan: Gunakan statistik maksimum untuk menetapkan ambang batas penskalaan optimal berdasarkan pola puncak dan mengonfigurasi kapasitas burst.
- Pemantauan kepatuhan SLA: Gunakan statistik maksimum untuk melacak waktu respons puncak dan tingkat kesalahan untuk memastikan kinerja layanan memenuhi yang ditentukan SLAs.

Informasi metrik terkait

Untuk informasi tentang Wawasan Kontainer, lihat kasus penggunaan Wawasan Penampung di CloudWatch Panduan Pengguna.

Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge

Menggunakan Amazon EventBridge, Anda dapat mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Tindakan yang dapat dikonfigurasi secara otomatis untuk menyertakan yang berikut:

- Menambahkan peristiwa ke grup log di CloudWatch Log
- Memanggil fungsi AWS Lambda
- Memanggil Perintah Amazon EC2 Run
- Mengirim peristiwa ke Amazon Kinesis Data Streams
- Mengaktifkan mesin AWS Step Functions negara
- Memberitahu topik Amazon SNS atau antrian Amazon Simple Queue Service (Amazon SQS)

Untuk informasi selengkapnya, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Anda dapat menggunakan acara Amazon ECS EventBridge untuk menerima notifikasi mendekati waktu nyata mengenai status klaster Amazon ECS Anda saat ini. Jika tugas Anda menggunakan tipe EC2 peluncuran, Anda dapat melihat status instance container dan status saat ini dari semua tugas yang berjalan pada instance container tersebut. Jika tugas Anda menggunakan tipe peluncuran Fargate, Anda dapat melihat status instance container.

Dengan menggunakan EventBridge, Anda dapat membuat penjadwal khusus di atas Amazon ECS yang bertanggung jawab untuk mengatur tugas di seluruh cluster dan memantau status cluster dalam waktu dekat. Anda dapat menghilangkan kode penjadwalan dan pemantauan yang terus-menerus melakukan polling layanan Amazon ECS untuk perubahan status dan sebagai gantinya menangani perubahan status Amazon ECS secara asinkron menggunakan target apa pun. EventBridge Target mungkin termasuk AWS Lambda, Layanan Antrian Sederhana Amazon, Layanan Pemberitahuan Sederhana Amazon, atau Amazon Kinesis Data Streams.

Aliran acara Amazon ECS memastikan bahwa setiap acara dikirimkan setidaknya satu kali. Jika peristiwa duplikat dikirim, peristiwa menyediakan informasi yang cukup untuk mengidentifikasi duplikat. Untuk informasi selengkapnya, lihat Menangani acara Amazon ECS.

Peristiwa relatif berurutan, sehingga Anda dapat dengan mudah memberitahu kapan suatu peristiwa terjadi dalam kaitannya dengan peristiwa lain.

Topik

- Peristiwa Amazon ECS
- Menangani acara Amazon ECS

Peristiwa Amazon ECS

Amazon ECS melacak status setiap tugas dan layanan Anda. Jika status tugas atau layanan berubah, peristiwa dibuat dan dikirim ke Amazon EventBridge. Peristiwa ini diklasifikasikan sebagai peristiwa perubahan status tugas dan peristiwa tindakan layanan. Peristiwa dan kemungkinan penyebabnya dijelaskan secara lebih detail di bagian berikut.

Amazon ECS menghasilkan dan mengirimkan jenis acara berikut ke EventBridge:

- Perubahan status instance kontainer
- Perubahan status tugas
- Perubahan status penerapan
- Tindakan layanan

> **Note**
>
> Amazon ECS dapat menambahkan jenis, sumber, dan detail acara lainnya di masa mendatang. Jika Anda melakukan de-serialisasi data JSON peristiwa dalam kode, pastikan aplikasi Anda siap menangani properti yang tidak dikenal untuk menghindari masalah jika dan kapan properti tambahan ini ditambahkan.

Dalam beberapa kasus, beberapa peristiwa dihasilkan untuk aktivitas yang sama. Misalnya, ketika tugas dimulai pada instance kontainer, peristiwa perubahan status tugas dihasilkan untuk tugas baru. Peristiwa perubahan status instans kontainer dihasilkan untuk memperhitungkan perubahan sumber

daya yang tersedia, seperti CPU, memori, dan port yang tersedia, pada instance kontainer. Demikian juga, jika instance kontainer dihentikan, peristiwa dihasilkan untuk instance kontainer, status koneksi agen kontainer, dan setiap tugas yang berjalan pada instance container.

Peristiwa perubahan status kontainer dan perubahan status tugas berisi dua bidang `version`: satu di bagian utama peristiwa, dan satu di objek `detail` peristiwa. Berikut ini menjelaskan perbedaan antara dua bidang tersebut:

- Bidang `version` dalam bagian utama peristiwa diatur ke `0` pada semua peristiwa. Untuk informasi selengkapnya tentang EventBridge parameter, lihat metadata peristiwa AWS layanan di EventBridge Panduan Pengguna Amazon.
- Bidang `version` dalam objek `detail` peristiwa menggambarkan versi sumber daya terkait. Setiap kali sumber daya berubah status, versi ini bertambah. Karena peristiwa dapat dikirim beberapa kali, bidang ini mengizinkan Anda untuk mengidentifikasi peristiwa duplikat. Peristiwa duplikat memiliki versi yang sama di objek `detail`. Jika Anda mereplikasi instans penampung Amazon ECS dan status tugas dengan EventBridge, Anda dapat membandingkan versi sumber daya yang dilaporkan oleh Amazon ECS APIs dengan versi yang dilaporkan EventBridge untuk sumber daya (di dalam `detail` objek) untuk memverifikasi bahwa versi dalam aliran peristiwa Anda saat ini.

Peristiwa tindakan layanan hanya berisi bidang `version` dalam bagian utama.

Acara tindakan layanan menentukan layanan dalam 2 bidang berbeda:

- Untuk acara yang dihasilkan oleh`create-service`, layanan ada di `serviceName` lapangan.
- Untuk acara yang dihasilkan oleh`update-service`, layanan ada di `service` lapangan.

Jika Anda menggunakan perkakas otomatis untuk acara layanan, Anda perlu membuat kode untuk kedua bidang.

Untuk informasi tentang cara membuat aturan untuk peristiwa tindakan layanan, lihatAcara tindakan layanan Amazon ECS.

Untuk informasi tambahan tentang cara mengintegrasikan Amazon ECS dan EventBridge, lihat Mengintegrasikan Amazon dan EventBridge Amazon ECS.

Acara perubahan status instans penampung Amazon ECS

Skenario berikut menyebabkan peristiwa perubahan status instance container:

Anda memanggil`StartTask`,`RunTask`, atau operasi `StopTask` API, baik secara langsung atau dengan AWS Management Console atau SDKs.

Menempatkan atau menghentikan tugas pada instans kontainer memodifikasi sumber daya yang tersedia pada instans kontainer, seperti CPU, memori, dan port yang tersedia.

Penjadwal layanan Amazon ECS memulai atau menghentikan tugas.

Menempatkan atau menghentikan tugas pada instans kontainer memodifikasi sumber daya yang tersedia pada instans kontainer, seperti CPU, memori, dan port yang tersedia.

Agen penampung Amazon ECS memanggil operasi `SubmitTaskStateChange` API dengan `STOPPED` status tugas dengan status yang diinginkan. `RUNNING`

Agen penampung Amazon ECS memantau status tugas pada instans penampung Anda, dan ia melaporkan perubahan status apa pun. Jika tugas yang seharusnya `RUNNING` dialihkan ke `STOPPED`, maka agen merilis sumber daya yang dialokasikan ke tugas yang berhenti, seperti CPU, memori, dan port yang tersedia.

Anda membatalkan pendaftaran instance container dengan operasi `DeregisterContainerInstance` API, baik secara langsung atau dengan atau. AWS Management Console SDKs

Membatalkan pendaftaran instance container mengubah status instance container dan status koneksi agen container Amazon ECS.

Sebuah tugas dihentikan ketika sebuah EC2 instance dihentikan.

Ketika Anda menghentikan instans kontainer, tugas yang berjalan di dalamnya dialihkan ke status `STOPPED`.

Agen kontainer Amazon ECS mendaftarkan instance kontainer untuk pertama kalinya.

Pertama kali agen penampung Amazon ECS mendaftarkan instance kontainer (saat peluncuran atau saat pertama kali dijalankan secara manual), ini membuat peristiwa perubahan status untuk instance tersebut.

Agen kontainer Amazon ECS menghubungkan atau memutuskan sambungan dari Amazon ECS.

Saat agen penampung Amazon ECS menghubungkan atau memutuskan sambungan dari backend Amazon ECS, agen penampung akan mengubah `agentConnected` status instans penampung.

> **Note**
>
> Agen kontainer Amazon ECS terputus dan menyambung kembali beberapa kali per jam sebagai bagian dari operasi normalnya, sehingga peristiwa koneksi agen harus diharapkan. Peristiwa ini bukan merupakan indikasi bahwa terdapat masalah dengan agen kontainer atau instans kontainer Anda.

Anda memutakhirkan agen penampung Amazon ECS pada sebuah instans.

Detail instans kontainer berisi objek untuk versi agen kontainer. Jika Anda memutakhirkan agen, informasi versi ini berubah dan menghasilkan acara.

Example Peristiwa perubahan status instans kontainer

Peristiwa perubahan status instans kontainer disampaikan dalam format berikut. `detail`Bagian di bawah ini menyerupai ContainerInstanceobjek yang dikembalikan dari operasi DescribeContainerInstancesAPI di Referensi API Amazon Elastic Container Service. Untuk informasi selengkapnya tentang EventBridge parameter, lihat metadata peristiwa AWS layanan di EventBridge Panduan Pengguna Amazon.

```
{
  "version": "0",
  "id": "8952ba83-7be2-4ab5-9c32-6687532d15a2",
  "detail-type": "ECS Container Instance State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315"
  ],
  "detail": {
    "agentConnected": true,
    "attributes": [
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
```

```
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
      },
      {
        "name": "com.amazonaws.ecs.capability.privileged-container"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      }
    ],
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "ec2InstanceId": "i-f3a8506b",
    "registeredResources": [
      {
```

```
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 2048
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 3767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "remainingResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 1988
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
```

```
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "status": "ACTIVE",
    "version": 14801,
    "versionInfo": {
      "agentHash": "aebcbca",
      "agentVersion": "1.13.0",
      "dockerVersion": "DockerVersion: 1.11.2"
    },
    "updatedAt": "2016-12-06T16:41:06.991Z"
  }
}
```

Acara perubahan status tugas Amazon ECS

Skenario berikut menyebabkan peristiwa perubahan status tugas:

Anda memanggil`StartTask`,`RunTask`, atau operasi `StopTask` API, baik secara langsung atau dengan AWS Management Console, AWS CLI, atau SDKs.

> Memulai atau menghentikan tugas membuat sumber daya tugas baru atau mengubah status sumber daya tugas yang ada.

Penjadwal layanan Amazon ECS memulai atau menghentikan tugas.

> Memulai atau menghentikan tugas membuat sumber daya tugas baru atau mengubah status sumber daya tugas yang ada.

Agen kontainer Amazon ECS memanggil operasi `SubmitTaskStateChange` API.

> Untuk jenis EC2 peluncuran Amazon, agen kontainer Amazon ECS memantau status tugas Anda pada instans penampung Anda. Agen kontainer Amazon ECS melaporkan perubahan status apa pun. Perubahan status dapat mencakup perubahan dari `PENDING` ke `RUNNING` atau dari `RUNNING` ke `STOPPED`.

Anda memaksa deregistrasi instance container yang mendasarinya dengan operasi `DeregisterContainerInstance` API dan `force` flag, baik secara langsung atau dengan atau. AWS Management Console SDKs

Membatalkan pendaftaran instance container mengubah status instance container dan status koneksi agen container Amazon ECS. Jika tugas berjalan di instans kontainer, bendera `force` harus diatur untuk mengizinkan pembatalan pendaftaran. Hal ini menghentikan semua tugas pada instans.

Instans kontainer yang mendasarinya dihentikan atau diakhiri.

Saat Anda menghentikan instans kontainer, maka tugas yang berjalan di dalamnya dialihkan ke status `STOPPED`.

Kontainer dalam tugas mengubah status.

Agen kontainer Amazon ECS memantau keadaan kontainer dalam tugas. Misalnya, jika kontainer yang berjalan dalam tugas berhenti, perubahan status kontainer ini menghasilkan peristiwa.

Tugas yang menggunakan penyedia kapasitas Fargate Spot menerima pemberitahuan penghentian.

Saat tugas menggunakan penyedia `FARGATE_SPOT` kapasitas dan dihentikan karena gangguan Spot, peristiwa perubahan status tugas akan dihasilkan.

Example Peristiwa perubahan status tugas

Peristiwa perubahan status tugas disampaikan dalam format berikut. Perhatikan hal berikut tentang bidang:

- Status kesehatan acara tidak tersedia dalam acara perubahan status tugas. Jika Anda memerlukan status kesehatan tugas, Anda dapat menjalankan tugas deskripsikan.
- Saat kontainer Anda menggunakan gambar yang dihosting dengan Amazon ECR, `imageDigest` bidang akan dikembalikan.
- Nilai untuk`createdAt`,,,`connectivityAt`, `pullStartedAt` `startedAt``pullStoppedAt`, dan `updatedAt` bidang adalah stempel waktu string ISO.
- `detail-type`Nilainya adalah “Perubahan Status Tugas ECS”.
- Ketika acara dibuat untuk tugas yang dihentikan, `stoppedCode` bidang `stoppedReason` dan memberikan informasi tambahan tentang mengapa tugas dihentikan (misalnya, “Pengguna dimulai”).

Untuk informasi selengkapnya tentang EventBridge parameter, lihat metadata peristiwa AWS layanan di Referensi EventBridge Acara Amazon.

Untuk informasi tentang cara mengonfigurasi aturan EventBridge peristiwa Amazon yang hanya menangkap peristiwa tugas di mana tugas telah berhenti berjalan karena salah satu container pentingnya telah dihentikan, lihat Mengirim peringatan Amazon Simple Notification Service untuk tugas Amazon ECS menghentikan peristiwa


```
{
    "version": "0",
    "id": "105f6bb1-4da6-c630-4965-35383018cbca",
    "detail-type": "ECS Task State Change",
    "source": "aws.ecs",
    "account": "123456789012",
    "time": "2025-05-06T11:02:34Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/a1173316d40a45dea9"
    ],
    "detail": {
        "attachments": [
            {
                "id": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-0d0eab1bb38d5ca64"
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-0103a2f01bad57d71"
                    },
                    {
                        "name": "macAddress",
                        "value": "0e:50:d1:c1:77:81"
                    },
                    {
                        "name": "privateDnsName",
                        "value": "ip-10-0-1-163.ec2.internal"
                    },
                    {
```

```
                        "name": "privateIPv4Address",
                        "value": "10.0.1.163"
                    }
                ]
            }
        ],
        "attributes": [
            {
                "name": "ecs.cpu-architecture",
                "value": "x86_64"
            }
        ],
        "availabilityZone": "us-east-1b",
        "capacityProviderName": "FARGATE",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/example-cluster",
        "connectivity": "CONNECTED",
        "connectivityAt": "2025-05-06T11:02:17.19Z",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-
cluster/a1173316d40a45dea9/a0a99b87-baa8-4bf6-b9f1-a9a95917a635",
                "lastStatus": "RUNNING",
                "name": "web",
                "image": "nginx",
                "imageDigest":
 "sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66",
                "runtimeId": "a1173316d40a45dea9-0265927825",
                "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
                "networkInterfaces": [
                    {
                        "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                        "privateIpv4Address": "10.0.1.163"
                    }
                ],
                "cpu": "99",
                "memory": "100"
            },
            {
                "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-
cluster/a1173316d40a45dea9/a2010e2d-ba7c-4135-8b79-e0290ff3cd8c",
                "lastStatus": "RUNNING",
                "name": "aws-guardduty-agent-nm40lC",
```

```
                "imageDigest":
 "sha256:bf9197abdf853607e5fa392b4f97ccdd6ca56dd179be3ce8849e552d96582ac8",
                "runtimeId": "a1173316d40a45dea9-2098416933",
                "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
                "networkInterfaces": [
                    {
                        "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                        "privateIpv4Address": "10.0.1.163"
                    }
                ],
                "cpu": "null"
            },
            {
                "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-
cluster/a1173316d40a45dea9/dccf0ca2-d929-471f-a5c3-98006fd4379e",
                "lastStatus": "RUNNING",
                "name": "aws-otel-collector",
                "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.32.0",
                "imageDigest":
 "sha256:7a1b3560655071bcacd66902c20ebe9a69470d5691fe3bd36baace7c2f3c4640",
                "runtimeId": "a1173316d40a45dea9-4027662657",
                "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
                "networkInterfaces": [
                    {
                        "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                        "privateIpv4Address": "10.0.1.163"
                    }
                ],
                "cpu": "0"
            }
        ],
        "cpu": "256",
        "createdAt": "2025-05-06T11:02:13.877Z",
        "desiredStatus": "RUNNING",
        "enableExecuteCommand": false,
        "ephemeralStorage": {
            "sizeInGiB": 20
        },
        "group": "family:webserver",
        "launchType": "FARGATE",
        "lastStatus": "RUNNING",
        "memory": "512",
```

```
        "overrides": {
            "containerOverrides": [
                {
                    "name": "web"
                },
                {
                    "environment": [
                        {
                            "name": "CLUSTER_NAME",
                            "value": "example-cluster"
                        },
                        {
                            "name": "REGION",
                            "value": "us-east-1"
                        },
                        {
                            "name": "HOST_PROC",
                            "value": "/host_proc"
                        },
                        {
                            "name": "AGENT_RUNTIME_ENVIRONMENT",
                            "value": "ecsfargate"
                        },
                        {
                            "name": "STAGE",
                            "value": "prod"
                        }
                    ],
                    "memory": 128,
                    "name": "aws-guardduty-agent-nm40lC"
                },
                {
                    "name": "aws-otel-collector"
                }
            ]
        },
        "platformVersion": "1.4.0",
        "pullStartedAt": "2025-05-06T11:02:24.162Z",
        "pullStoppedAt": "2025-05-06T11:02:33.493Z",
        "startedAt": "2025-05-06T11:02:34.325Z",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/
webserver:5",
```

```
        "updatedAt": "2025-05-06T11:02:34.325Z",
        "version": 3
    }
}
```

Example

Berikut ini adalah contoh peristiwa perubahan status tugas untuk jenis EC2 peluncuran.

```
{
    "version": "0",
    "id": "a65cf262-f104-0dd5-ceda-4b09ba71a441",
    "detail-type": "ECS Task State Change",
    "source": "aws.ecs",
    "account": "123456789012",
    "time": "2025-05-12T13:12:06Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecs:us-east-1:123456789012:task/example/c1ffa94f19a540ed8d9f7e1d2a5d"
    ],
    "detail": {
        "attachments": [
            {
                "id": "52333e3b-b812-41a8-b057-9ed184bbe5e1",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-0d0eab1bb38d5ca64"
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-0ea90f746500773a4"
                    },
                    {
                        "name": "macAddress",
                        "value": "0e:d5:9b:ce:49:fb"
                    },
                    {
                        "name": "privateDnsName",
                        "value": "ip-10-0-1-37.ec2.internal"
                    },
                    {
```

```
                        "name": "privateIPv4Address",
                        "value": "10.0.1.37"
                    }
                ]
            }
        ],
        "attributes": [
            {
                "name": "ecs.cpu-architecture",
                "value": "x86_64"
            }
        ],
        "availabilityZone": "us-east-1b",
        "capacityProviderName": "Infra-ECS-Cluster-example-fa84e0cc-
AsgCapacityProvider-OseQJU9pizmp",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/example",
        "connectivity": "CONNECTED",
        "connectivityAt": "2025-05-12T13:11:44.98Z",
        "containerInstanceArn": "arn:aws:ecs:us-east-1:123456789012:container-instance/
example/d1d84798400f49f3b21cb61610c1e",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example/
c1ffa94f19a540ed8d9f7e1d2a5d3626/197d0994-5367-4a6d-9f9a-f075e4a6",
                "lastStatus": "RUNNING",
                "name": "aws-otel-collector",
                "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.32.0",
                "imageDigest":
 "sha256:7a1b3560655071bcacd66902c20ebe9a69470d5691fe3bd36baace7c2f3c4640",
                "runtimeId":
 "8e926f0ccd8fe2b459926f49584ba6d33a3d9f61398dbabe944ee6a13a8ff3a1",
                "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
                "networkInterfaces": [
                    {
                        "attachmentId": "52333e3b-b812-41a8-b057-9ed184bbe5e1",
                        "privateIpv4Address": "10.0.1.37"
                    }
                ],
                "cpu": "0"
            },
            {
                "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example/
c1ffa94f19a540ed8d9f7e1d2a5d3626/cab39ef0-9c50-459d-844b-b9d51d73d",
```

```
                "lastStatus": "RUNNING",
                "name": "web",
                "image": "nginx",
                "imageDigest":
 "sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66",
                "runtimeId":
 "9f1c73f0094f051541d9e5c2ab1e172d83c4eb5171bcc857c4504b02770ff3b8",
                "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
                "networkInterfaces": [
                    {
                        "attachmentId": "52333e3b-b812-41a8-b057-9ed184bbe5e1",
                        "privateIpv4Address": "10.0.1.37"
                    }
                ],
                "cpu": "99",
                "memory": "100"
            }
        ],
        "cpu": "256",
        "createdAt": "2025-05-12T13:11:44.98Z",
        "desiredStatus": "RUNNING",
        "enableExecuteCommand": false,
        "group": "family:webserver",
        "launchType": "EC2",
        "lastStatus": "RUNNING",
        "memory": "512",
        "overrides": {
            "containerOverrides": [
                {
                    "name": "aws-otel-collector"
                },
                {
                    "name": "web"
                }
            ]
        },
        "pullStartedAt": "2025-05-12T13:11:59.491Z",
        "pullStoppedAt": "2025-05-12T13:12:05.896Z",
        "startedAt": "2025-05-12T13:12:06.053Z",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/
webserver",
```

```
        "updatedAt": "2025-05-12T13:12:06.053Z",
        "version": 4
    }
}
```

Acara tindakan layanan Amazon ECS

Amazon ECS mengirimkan peristiwa tindakan layanan dengan jenis detail ECS Service Action. Tidak seperti peristiwa perubahan status instans kontainer dan tugas, peristiwa tindakan layanan tidak mencakup nomor versi di bidang tanggapan `details`. Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa tindakan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Memulai EventBridge di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Service Action"
    ]
}
```

Amazon ECS mengirimkan acara dengan`INFO`,`WARN`, dan jenis `ERROR` acara. Berikut ini adalah peristiwa tindakan layanan.

Peristiwa tindakan layanan dengan tipe peristiwa **`INFO`**

`SERVICE_STEADY_STATE`

Kondisi layanan ini baik dan pada jumlah tugas yang diinginkan, sehingga mencapai status stabil. Penjadwal layanan melaporkan status secara berkala, sehingga Anda mungkin menerima pesan ini beberapa kali.

`TASKSET_STEADY_STATE`

Tugas yang ditetapkan dalam kondisi baik dan pada jumlah tugas yang diinginkan, sehingga mencapai status stabil.

`CAPACITY_PROVIDER_STEADY_STATE`

Penyedia kapasitas yang terkait dengan layanan mencapai status stabil.

`SERVICE_DESIRED_COUNT_UPDATED`

Ketika penjadwal layanan memperbarui jumlah hitungan yang diinginkan untuk set layanan atau tugas. Peristiwa ini tidak dikirim ketika jumlah yang diinginkan diperbarui secara manual oleh pengguna.

`TASKS_STOPPED`

Layanan telah menghentikan tugas yang sedang berjalan.

`SERVICE_DEPLOYMENT_IN_PROGRESS`

Penyebaran layanan sedang berlangsung. Penyebaran layanan dapat berupa rollback, atau revisi layanan baru.

`SERVICE_DEPLOYMENT_COMPLETED`

Penyebaran layanan dalam kondisi mapan dan selesai. Penyebaran layanan dapat berupa rollback, atau untuk menerapkan revisi layanan yang diperbarui.

Peristiwa tindakan layanan dengan tipe peristiwa **`WARN`**

`SERVICE_TASK_START_IMPAIRED`

Layanan ini tidak dapat berhasil memulai tugas secara konsisten.

`SERVICE_DISCOVERY_INSTANCE_UNHEALTHY`

Layanan yang menggunakan penemuan layanan berisi tugas yang kondisinya tidak baik. Penjadwal layanan mendeteksi bahwa kondisi tugas dalam registri layanan tidak baik.

`VPC_LATTICE_TARGET_UNHEALTHY`

Layanan yang menggunakan VPC Lattice telah mendeteksi salah satu target untuk VPC Lattice tidak sehat.

Peristiwa tindakan layanan dengan tipe peristiwa **`ERROR`**

`SERVICE_DAEMON_PLACEMENT_CONSTRAINT_VIOLATED`

Tugas dalam layanan yang menggunakan strategi penjadwal layanan `DAEMON` tidak lagi memenuhi strategi kendala penempatan untuk layanan.

`ECS_OPERATION_THROTTLED`

Penjadwal layanan telah dibatasi karena batas throttle Amazon ECS API.

`SERVICE_DISCOVERY_OPERATION_THROTTLED`

Penjadwal layanan telah dibatasi karena batas throttle AWS Cloud Map API. Hal ini dapat terjadi pada layanan yang dikonfigurasi untuk menggunakan penemuan layanan.

`SERVICE_TASK_PLACEMENT_FAILURE`

Penjadwal layanan tidak dapat menempatkan tugas. Penyebabnya akan dijelaskan dalam bidang `reason`.

Penyebab umum untuk acara layanan ini dihasilkan adalah karena kurangnya sumber daya di cluster untuk menempatkan tugas. Misalnya, kurangnya kapasitas CPU atau memori pada instans kontainer yang tersedia atau tidak tersedianya instans kontainer. Penyebab umum lainnya adalah ketika agen penampung Amazon ECS terputus pada instance penampung, menyebabkan penjadwal tidak dapat menempatkan tugas.

`SERVICE_TASK_CONFIGURATION_FAILURE`

Penjadwal layanan tidak dapat menempatkan tugas karena kesalahan konfigurasi. Penyebabnya akan dijelaskan dalam bidang `reason`.

Penyebab umum peristiwa layanan ini dihasilkan adalah karena tag diterapkan ke layanan tetapi pengguna atau peran belum memilih format Amazon Resource Name (ARN) baru di Wilayah. Untuk informasi selengkapnya, lihat Nama Sumber Daya Amazon (ARNs) dan IDs. Penyebab umum lainnya adalah Amazon ECS tidak dapat mengambil peran tugas IAM yang disediakan.

`SERVICE_HEALTH_UNKNOWN`

Layanan tidak dapat menggambarkan data kesehatan untuk tugas-tugas.

`SERVICE_DEPLOYMENT_FAILED`

Penyebaran layanan tidak mencapai stabil. Ini terjadi ketika a CloudWatch dipicu atau pemutus sirkuit mendeteksi kegagalan penyebaran layanan.

Example Peristiwa status stabil layanan

Peristiwa status stabil layanan disampaikan dalam format berikut. Untuk informasi selengkapnya tentang EventBridge parameter, lihat Peristiwa EventBridge di Panduan EventBridge Pengguna Amazon.

```
{
    "version": "0",
    "id": "af3c496d-f4a8-65d1-70f4-a69d52e9b584",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:27:22Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:27:22.695Z"
    }
}
```

Example Peristiwa status stabil penyedia kapasitas

Peristiwa status stabil penyedia kapasitas disampaikan dalam format berikut.

```
{
    "version": "0",
    "id": "b9baa007-2f33-0eb1-5760-0d02a572d81f",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:37:00Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "CAPACITY_PROVIDER_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/ASG-tutorial-
capacity-provider"
        ],
        "createdAt": "2019-11-19T19:37:00.807Z"
```

```
    }
}
```

Example Peristiwa gangguan memulai tugas layanan

Peristiwa gangguan memulai tugas layanan disampaikan dalam format berikut.

```
{
    "version": "0",
    "id": "57c9506e-9d21-294c-d2fe-e8738da7e67d",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "WARN",
        "eventName": "SERVICE_TASK_START_IMPAIRED",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:55:38.725Z"
    }
}
```

Example Peristiwa kegagalan penempatan tugas layanan

Peristiwa kegagalan penempatan tugas disampaikan dalam format berikut. Untuk informasi selengkapnya, lihat Acara EventBridge di Panduan EventBridge Pengguna Amazon.

Dalam contoh berikut, tugas mencoba untuk menggunakan penyedia kapasitas `FARGATE_SPOT` tetapi penjadwal layanan tidak dapat memperoleh kapasitas Fargate Spot.

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
```

```
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "ERROR",
        "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/FARGATE_SPOT"
        ],
        "reason": "RESOURCE:FARGATE",
        "createdAt": "2019-11-06T19:09:33.087Z"
    }
}
```

Dalam contoh berikut untuk jenis EC2 peluncuran, tugas dicoba untuk diluncurkan pada Instance Container `2dd1b186f39845a584488d2ef155c131` tetapi penjadwal layanan tidak dapat menempatkan tugas karena CPU tidak mencukupi.

```
{
  "version": "0",
  "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
  "detail-type": "ECS Service Action",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2019-11-19T19:55:38Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
  ],
  "detail": {
    "eventType": "ERROR",
    "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
    "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
    ],
    "reason": "RESOURCE:CPU",
    "createdAt": "2019-11-06T19:09:33.087Z"
  }
}
```

Acara perubahan status penerapan layanan Amazon ECS

Amazon ECS mengirimkan peristiwa perubahan status penerapan layanan dengan jenis detail Perubahan Status Penerapan ECS. Berikut ini adalah pola peristiwa yang digunakan untuk membuat EventBridge aturan untuk peristiwa perubahan status penerapan layanan Amazon ECS. Untuk informasi selengkapnya tentang membuat EventBridge aturan, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Deployment State Change"
    ]
}
```

Amazon ECS mengirimkan acara dengan `INFO` dan jenis `ERROR` acara. Untuk informasi selengkapnya, lihat Acara tindakan layanan Amazon ECS

Berikut adalah peristiwa perubahan status deployment layanan.

`SERVICE_DEPLOYMENT_IN_PROGRESS`

Deployment layanan sedang berlangsung. Peristiwa ini dikirim untuk deployment awal dan deployment rollback.

`SERVICE_DEPLOYMENT_COMPLETED`

Deployment layanan telah selesai. Peristiwa ini dikirim setelah layanan mencapai status stabil setelah deployment.

`SERVICE_DEPLOYMENT_FAILED`

Deployment layanan gagal. Acara ini dikirim untuk layanan dengan logika pemutus sirkuit penyebaran diaktifkan.

Example Peristiwa deployment layanan sedang berlangsung

Peristiwa deployment layanan yang sedang berlangsung disampaikan ketika deployment awal dan rollback dimulai. Perbedaan antara keduanya ada di bidang `reason`. Untuk informasi selengkapnya

tentang EventBridge parameter, lihat metadata peristiwa AWS layanan; di EventBridge Panduan Pengguna Amazon.

Berikut ini menampilkan contoh output untuk permulaan deployment awal.

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3a6EXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment deploymentId in progress."
   }
}
```

Berikut ini menampilkan contoh output untuk permulaan deployment rollback. Bidang `reason` menyediakan ID deployment yang digunakan untuk mematalkan layanan.

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
```

```
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment circuit breaker: rolling back to
 deploymentId deploymentID."
    }
}
```

Example Peristiwa deployment layanan yang telah selesai

Peristiwa status deployment layanan yang telah selesai disampaikan dalam format berikut. Untuk informasi selengkapnya, lihat Menyebarkan layanan Amazon ECS dengan mengganti tugas.

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_COMPLETED",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment deploymentID completed."
    }
}
```

Example Peristiwa deployment layanan yang gagal

Peristiwa deployment layanan yang gagal disampaikan dalam format berikut. Peristiwa status gagal penerapan layanan hanya akan dikirim untuk layanan yang mengaktifkan logika pemutus sirkuit penerapan. Untuk informasi selengkapnya, lihat Menyebarkan layanan Amazon ECS dengan mengganti tugas.

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
```

```
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "ERROR",
         "eventName": "SERVICE_DEPLOYMENT_FAILED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: task failed to start."
    }
}
```

Menangani acara Amazon ECS

Amazon ECS mengirimkan acara setidaknya sekali. Ini berarti Anda mungkin menerima banyak salinan dari acara tertentu. Selain itu, peristiwa mungkin tidak dikirimkan ke listener peristiwa Anda sesuai urutan peristiwa yang terjadi.

Untuk memesan acara dengan benar, `detail` bagian dari setiap acara berisi `version` properti. Setiap kali sumber daya mengubah status, `version` ini bertambah. Peristiwa duplikat memiliki `version` yang sama di objek `detail`. Jika Anda mereplikasi instans penampung Amazon ECS dan status tugas dengan EventBridge, Anda dapat membandingkan versi sumber daya yang dilaporkan oleh Amazon ECS APIs dengan sumber daya yang `version` dilaporkan dalam EventBridge untuk memverifikasi bahwa versi dalam aliran peristiwa Anda saat ini. Peristiwa dengan jumlah properti versi yang lebih tinggi harus diperlakukan sebagai peristiwa yang terjadi setelah peristiwa dengan jumlah versi yang lebih rendah.

Contoh: Menangani peristiwa di fungsi AWS Lambda

Contoh berikut menunjukkan fungsi Lambda yang ditulis dengan Python 3.9 yang menangkap peristiwa perubahan status instance tugas dan kontainer dan menyimpannya ke salah satu dari dua tabel Amazon DynamoDB:

- ECSCtrInstanceState— Menyimpan status terbaru untuk instance kontainer. ID tabel adalah nilai `containerInstanceArn` instans kontainer.

- ECSTaskNegara - Menyimpan status terbaru untuk suatu tugas. ID tabel adalah nilai `taskArn` tugas.

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    elif event["detail-type"] == "ECS Container Instance State Change":
        table_name = "ECSCtrInstanceState"
        id_name =  "containerInstanceArn"
        event_id = event["detail"]["containerInstanceArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
 without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")
```

```
    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
 updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
 ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

Contoh Fargate berikut menunjukkan fungsi Lambda yang ditulis dengan Python 3.9 yang menangkap peristiwa perubahan status tugas dan menyimpannya ke tabel Amazon DynamoDB berikut:

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
```

```
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
        raise ValueError("Function only supports input from events with a source type
of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
```

```
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan

Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan menyediakan semua metrik Wawasan Kontainer, ditambah metrik tugas dan kontainer tambahan.

Container Insights menemukan semua kontainer yang sedang berjalan di cluster dan mengumpulkan data kinerja di setiap lapisan tumpukan kinerja. Data operasional dikumpulkan sebagai log acara performa. Ini adalah entri yang menggunakan skema JSON terstruktur untuk data kardinalitas tinggi untuk dicerna dan disimpan dalam skala besar. Dari data ini, CloudWatch buat metrik agregat tingkat yang lebih tinggi di tingkat klaster, layanan, dan tugas sebagai metrik. CloudWatch Metrik tersebut mencakup pemanfaatan sumber daya seperti CPU, memori, disk, dan jaringan. Metrik tersebut tersedia di dasbor otomatis CloudWatch.

Metrik hanya mencerminkan sumber daya dengan menjalankan tugas selama rentang waktu yang ditentukan. Misalnya, jika Anda memiliki klaster dengan satu layanan di dalamnya namun layanan tersebut tidak memiliki tugas dengan status `RUNNING`, maka tidak akan ada metrik yang dikirimkan ke CloudWatch. Jika Anda memiliki dua layanan dan salah satunya memiliki tugas yang sedang berjalan sementara yang satunya tidak, maka hanya metrik untuk layanan dengan tugas yang sedang berjalan lah yang akan dikirim.

Pada tanggal 2 Desember 2024, AWS merilis Container Insights dengan peningkatan observabilitas untuk Amazon ECS. Versi ini mendukung peningkatan observabilitas untuk cluster Amazon ECS menggunakan jenis peluncuran Amazon dan EC2 Fargate. Setelah Anda mengonfigurasi Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan di Amazon ECS, Container Insights

secara otomatis mengumpulkan telemetri infrastruktur terperinci dari tingkat klaster hingga tingkat penampung di lingkungan Anda dan menampilkan data kinerja penting ini di dasbor yang dikuratori yang menghapus beban berat dalam pengaturan observabilitas. Untuk informasi tentang cara menyiapkan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan, lihat. Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan

Kami menyarankan Anda menggunakan Wawasan Kontainer dengan observabilitas yang ditingkatkan, bukan Wawasan Kontainer karena ini memberikan visibilitas mendetail di lingkungan penampung Anda, sehingga mengurangi waktu rata-rata untuk resolusi. Untuk informasi selengkapnya, lihat Amazon ECS Container Insights dengan metrik observabilitas yang disempurnakan di Panduan Pengguna Amazon. CloudWatch

Acara yang dapat Anda lihat adalah peristiwa yang dikirimkan Amazon ECS ke Amazon EventBridge. Untuk informasi selengkapnya, lihat acara Amazon ECS.

> **Important**
>
> Metrik yang dikumpulkan oleh CloudWatch Container Insights dikenakan biaya sebagai metrik kustom. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatchHarga.

Tentukan kesehatan tugas Amazon ECS menggunakan pemeriksaan kesehatan kontainer

Saat membuat definisi tugas, Anda dapat mengonfigurasi pemeriksaan kesehatan untuk kontainer Anda. Pemeriksaan Kesehatan adalah perintah yang berjalan secara lokal pada wadah dan memvalidasi kesehatan dan ketersediaan aplikasi.

Agen kontainer Amazon ECS hanya memantau dan melaporkan pemeriksaan kesehatan yang ditentukan dalam definisi tugas. Amazon ECS tidak memantau pemeriksaan kesehatan Docker yang disematkan dalam gambar kontainer tetapi tidak ditentukan dalam definisi penampung. Parameter pemeriksaan kondisi yang ditentukan dalam ketentuan kontainer menimpa setiap pemeriksaan kondisi Docker yang ada di citra kontainer.

Ketika pemeriksaan kesehatan didefinisikan dalam definisi tugas, wadah menjalankan proses pemeriksaan kesehatan di dalam wadah, dan kemudian mengevaluasi kode keluar untuk menentukan kesehatan aplikasi.

Pemeriksaan kesehatan terdiri dari parameter berikut:

- Command — Perintah yang dijalankan kontainer untuk menentukan apakah itu sehat. Array string dapat dimulai dengan `CMD` untuk menjalankan argumen perintah secara langsung, atau `CMD-SHELL` untuk menjalankan perintah dengan shell default container. Gunakan `CMD-SHELL` saat Anda membutuhkan fitur shell seperti pipa, pengalihan, rantai perintah, atau ekspansi variabel lingkungan. Misalnya, `CMD-SHELL` memungkinkan Anda untuk menggunakan perintah seperti `curl -f http://localhost/ || exit 1` di mana shell menafsirkan `||` operator. Gunakan `CMD` untuk perintah sederhana yang tidak memerlukan interpretasi shell.
- Interval — Periode waktu (dalam detik) antara setiap pemeriksaan kesehatan.
- Timeout — Periode waktu (dalam hitungan detik) untuk menunggu pemeriksaan kesehatan berhasil sebelum dianggap gagal.
- Mencoba lagi — Berapa kali untuk mencoba kembali pemeriksaan kesehatan yang gagal sebelum wadah dianggap tidak sehat.
- Periode mulai - Masa tenggang opsional untuk menyediakan waktu kontainer untuk bootstrap sebelum pemeriksaan kesehatan gagal dihitung terhadap jumlah maksimum percobaan ulang.

  Jika pemeriksaan kondisi berhasil dalam `startPeriod`, maka kontainer dianggap sehat dan kegagalan berikutnya dihitung terhadap jumlah maksimum percobaan kembali.

Untuk informasi tentang cara menentukan pemeriksaan kesehatan dalam definisi tugas, lihatPemeriksaan kondisi.

Berikut ini menjelaskan kemungkinan nilai status kesehatan untuk sebuah wadah:

- `HEALTHY`—Pemeriksaan kesehatan kontainer telah berhasil lolos.
- `UNHEALTHY`—Pemeriksaan kesehatan kontainer gagal.
- `UNKNOWN`—Pemeriksaan kesehatan kontainer sedang dievaluasi, tidak ada pemeriksaan kesehatan kontainer yang ditentukan, atau Amazon ECS tidak memiliki status kesehatan wadah.

Perintah pemeriksaan kesehatan berjalan di wadah. Oleh karena itu Anda harus menyertakan perintah dalam gambar kontainer.

Pemeriksaan kesehatan terhubung ke aplikasi melalui antarmuka loopback kontainer di `localhost` atau. `127.0.0.1` Kode keluar `0` menunjukkan keberhasilan, dan kode keluar bukan nol menunjukkan kegagalan.

Pertimbangkan hal-hal berikut saat menggunakan pemeriksaan kesehatan kontainer:

- Pemeriksaan kondisi kontainer memerlukan versi 1.17.0 atau yang lebih baru dari agen kontainer Amazon ECS.
- Pemeriksaan kesehatan kontainer didukung untuk tugas Fargate jika Anda menggunakan versi platform Linux atau yang lebih besar `1.1.0` atau versi `1.1.0` platform Windows atau yang lebih besar

Bagaimana Amazon ECS menentukan kesehatan tugas

Wadah yang penting dan memiliki perintah pemeriksaan kesehatan dalam definisi tugas adalah satu-satunya yang dipertimbangkan untuk menentukan kesehatan tugas.

Aturan berikut dievaluasi secara berurutan:

1. Jika status satu wadah penting adalah`UNHEALTHY`, maka status tugasnya adalah`UNHEALTHY`.
2. Jika status satu wadah penting adalah`UNKNOWN`, maka status tugasnya adalah`UNKNOWN`.
3. Jika status semua wadah penting adalah`HEALTHY`, maka status tugasnya adalah`HEALTHY`.

Pertimbangkan contoh kesehatan tugas berikut dengan 2 wadah penting.

Kontainer 1 kesehatan	Kontainer 2 kesehatan	Kesehatan tugas
`UNHEALTHY`	`UNKNOWN`	`UNHEALTHY`
`UNHEALTHY`	`HEALTHY`	`UNHEALTHY`
`HEALTHY`	`UNKNOWN`	`UNKNOWN`
`HEALTHY`	`HEALTHY`	`HEALTHY`

Perhatikan contoh kesehatan tugas berikut dengan 3 kontainer.

Kontainer 1 kesehatan	Kontainer 2 kesehatan	Kontainer 3 kesehatan	Kesehatan tugas
`UNHEALTHY`	`UNKNOWN`	`UNKNOWN`	`UNHEALTHY`

Kontainer 1 kesehatan	Kontainer 2 kesehatan	Kontainer 3 kesehatan	Kesehatan tugas
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

Bagaimana pemeriksaan kesehatan dipengaruhi oleh pemutusan hubungan agen

Jika agen penampung Amazon ECS terputus dari layanan Amazon ECS, ini tidak akan menyebabkan kontainer beralih ke status. `UNHEALTHY` Ini dirancang, untuk memastikan bahwa kontainer tetap berjalan selama agen restart atau tidak tersedianya sementara. Status pemeriksaan kesehatan adalah tanggapan “terakhir terdengar dari” dari agen Amazon ECS, jadi jika wadah dipertimbangkan `HEALTHY` sebelum pemutusan, status itu akan tetap sampai agen terhubung kembali dan pemeriksaan kesehatan lain terjadi. Tidak ada asumsi yang dibuat tentang status pemeriksaan kesehatan kontainer.

Melihat kesehatan wadah Amazon ECS

Anda dapat melihat kesehatan kontainer di konsol, dan menggunakan API `DescribeTasks` sebagai respons. Untuk informasi selengkapnya, lihat DescribeTasksdi Referensi API Amazon Elastic Container Service.

Jika Anda menggunakan logging untuk penampung Anda, misalnya Amazon CloudWatch Logs, Anda dapat mengonfigurasi perintah pemeriksaan kesehatan untuk meneruskan output kesehatan kontainer ke log Anda. Pastikan untuk menggunakan `2&1` untuk menangkap `stderr` informasi `stdout` dan informasi. Contoh berikut digunakan `CMD-SHELL` karena memerlukan fitur shell seperti operator pipe (>>) dan logis OR (||):

```
"command": [
     "CMD-SHELL",
```

```
    "curl -f http://localhost/ >> /proc/1/fd/1 2>&1  || exit 1"
  ],
```

Pantau kesehatan instans kontainer Amazon ECS

Amazon ECS menyediakan pemantauan kesehatan instans kontainer. Anda dapat dengan cepat menentukan apakah Amazon ECS telah mendeteksi masalah apa pun yang mungkin mencegah instance container Anda menjalankan container. Amazon ECS melakukan pemeriksaan otomatis pada setiap instance container yang berjalan dengan versi agen `1.57.0` atau yang lebih baru untuk mengidentifikasi masalah. Untuk informasi selengkapnya tentang memverifikasi versi agen sebuah instance container, lihatMemperbarui agen kontainer Amazon ECS.

Anda harus menggunakan AWS CLI versi `1.22.3` atau yang lebih baru atau AWS CLI versi `2.3.6` atau yang lebih baru. Untuk informasi tentang cara memperbarui AWS CLI, lihat Menginstal atau memperbarui versi terbaru dari AWS CLI Panduan AWS Command Line Interface Pengguna Versi 2.

Untuk melihat kesehatan instance container, jalankan `describe-container-instances` dengan `CONTAINER_INSTANCE_HEALTH` opsi.

Berikut ini adalah nilai yang valid untuk`overallStatus`:

- `OK`
- `IMPAIRED`
- `INSUFFICIENT_DATA`
- `INITIALIZING`

Berikut ini adalah contoh cara menjalankannya`describe-container-instances`.

```
aws ecs describe-container-instances \
     --cluster cluster_name \
     --container-instances 47279cd2cadb41cbaef2dcEXAMPLE \
     --include CONTAINER_INSTANCE_HEALTH
```

Berikut ini adalah contoh objek status kesehatan dalam output.

```
"healthStatus": {
 "overallStatus": "OK",
 "details": [{
```

```
  "type": "CONTAINER_RUNTIME",
  "status": "OK",
  "lastUpdated": "2021-11-10T03:30:26+00:00",
  "lastStatusChange": "2021-11-10T03:26:41+00:00"
 }]
}
```

Masalah kesehatan instans kontainer

Ketika status `overallStatus` apa pun selain`OK`, coba yang berikut ini:

- Tunggu, lalu lari `describe-container-instances`
- Lihat kesehatan instance container Anda di EC2 konsol atau dengan menggunakan CLI.
- Tinjau CloudWatch metrik. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch
- Periksa AWS Health Dashboard untuk melihat apakah ada masalah dengan layanan ini.

Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi

Amazon ECS terintegrasi dengan AWS Distro OpenTelemetry untuk mengumpulkan data jejak dari aplikasi Anda. Amazon ECS menggunakan AWS Distro untuk wadah OpenTelemetry sespan untuk mengumpulkan dan merutekan data penelusuran ke. AWS X-Ray Untuk informasi selengkapnya, lihat Menyiapkan AWS Distro untuk OpenTelemetry Kolektor di Amazon ECS. Anda kemudian dapat menggunakan AWS X-Ray untuk mengidentifikasi kesalahan dan pengecualian, menganalisis kemacetan kinerja dan waktu respons.

Agar AWS Distro for OpenTelemetry Collector dapat mengirim data jejak AWS X-Ray, aplikasi Anda harus dikonfigurasi untuk membuat data jejak. Untuk informasi selengkapnya, lihat Menginstrumentasi aplikasi Anda AWS X-Ray di Panduan AWS X-Ray Pengembang.

Izin IAM yang diperlukan untuk AWS Distro untuk integrasi dengan OpenTelemetry AWS X-Ray

Integrasi Amazon ECS dengan AWS Distro for OpenTelemetry mengharuskan Anda membuat peran tugas dan menentukan peran dalam definisi tugas Anda. Kami menyarankan Anda mengonfigurasi AWS Distro untuk OpenTelemetry sespan untuk merutekan log kontainer ke Log. CloudWatch

> **Important**
>
> Jika Anda juga mengumpulkan metrik aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi.

Buat kebijakan berikut, lalu lampirkan ke peran eksekusi tugas.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "ssm:GetParameters"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Menentukan AWS Distro untuk OpenTelemetry sespan untuk AWS X-Ray integrasi dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pembuatan AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Use trace collection. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Cuplikan definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk integrasi. AWS X-Ray

```
{
 "family": "otel-using-xray",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryXrayRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
```

```
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/aws-otel-emitter",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    },
    "dependsOn": [{
     "containerName": "aws-otel-collector",
     "condition": "START"
    }]
   },
   {
    "name": "aws-otel-collector",
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
    "essential": true,
    "command": [
     "--config=/etc/ecs/otel-instance-metrics-config.yaml"
    ],
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
}
```

Korelasikan kinerja aplikasi Amazon ECS menggunakan metrik aplikasi

Amazon ECS di Fargate mendukung pengumpulan metrik dari aplikasi Anda yang berjalan di Fargate dan mengekspornya ke Amazon atau CloudWatch Amazon Managed Service untuk Prometheus.

Anda dapat menggunakan metadata yang dikumpulkan untuk mengkorelasikan data kinerja aplikasi dengan data infrastruktur yang mendasarinya, sehingga mengurangi waktu rata-rata untuk menyelesaikan masalah.

Amazon ECS menggunakan AWS Distro untuk wadah OpenTelemetry sespan untuk mengumpulkan dan merutekan metrik aplikasi Anda ke tujuan. Pengalaman konsol Amazon ECS menyederhanakan proses menambahkan integrasi ini saat membuat definisi tugas Anda.

Topik

- Mengekspor metrik aplikasi ke Amazon CloudWatch
- Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus

Mengekspor metrik aplikasi ke Amazon CloudWatch

Amazon ECS di Fargate mendukung ekspor metrik aplikasi kustom Anda ke Amazon CloudWatch sebagai metrik khusus. Ini dilakukan dengan menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Konsol Amazon ECS menyederhanakan proses ini dengan menambahkan opsi Gunakan koleksi metrik saat membuat definisi tugas baru. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Metrik aplikasi diekspor ke CloudWatch Log dengan nama grup log `/aws/ecs/application/metrics` dan metrik dapat dilihat di namespace. `ECS/AWSOTel/Application` Aplikasi Anda harus diinstrumentasi dengan OpenTelemetry SDK. Untuk informasi selengkapnya, lihat Pengantar AWS Distro untuk OpenTelemetry di AWS Distro untuk OpenTelemetry dokumentasi.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan Amazon ECS pada integrasi Fargate AWS dengan Distro OpenTelemetry untuk mengirim metrik aplikasi ke Amazon. CloudWatch

- Integrasi ini hanya mengirimkan metrik aplikasi kustom Anda ke CloudWatch. Jika menginginkan metrik tingkat tugas, Anda dapat mengaktifkan Wawasan Kontainer di konfigurasi cluster Amazon

ECS. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.

- AWS Distro untuk OpenTelemetry integrasi didukung untuk beban kerja Amazon ECS yang dihosting di beban kerja Fargate dan Amazon ECS yang dihosting di instans Amazon. EC2 Instance eksternal saat ini tidak didukung.
- CloudWatch mendukung maksimum 30 dimensi per metrik. Secara default, Amazon ECS secara default menyertakan`TaskARN`,,,, `ClusterARN` `LaunchType` `TaskDefinitionFamily`, dan `TaskDefinitionRevision` dimensi ke metrik. 25 dimensi yang tersisa dapat ditentukan oleh aplikasi Anda. Jika lebih dari 30 dimensi dikonfigurasi, tidak CloudWatch dapat menampilkannya. Ketika ini terjadi, metrik aplikasi akan muncul di namespace `ECS/AWSOTel/Application` CloudWatch metrik tetapi tanpa dimensi apa pun. Anda dapat menginstruksikan aplikasi Anda untuk menambahkan dimensi tambahan. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch metrik dengan AWS Distro untuk OpenTelemetry dokumentasi di AWS Distro. OpenTelemetry

Izin IAM yang diperlukan untuk AWS Distro untuk OpenTelemetry integrasi dengan Amazon CloudWatch

Integrasi Amazon ECS dengan AWS Distro untuk OpenTelemetry mengharuskan Anda membuat peran IAM tugas dan menentukan peran dalam definisi tugas Anda. Kami merekomendasikan bahwa AWS Distro untuk OpenTelemetry sespan juga dikonfigurasi untuk merutekan log kontainer ke CloudWatch Log yang memerlukan peran IAM eksekusi tugas dibuat dan ditentukan dalam definisi tugas Anda juga. Konsol Amazon ECS menangani peran IAM eksekusi tugas atas nama Anda, tetapi peran IAM tugas harus dibuat secara manual dan ditambahkan ke definisi tugas Anda. Untuk informasi selengkapnya tentang peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Jika Anda juga mengumpulkan data jejak aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi.
> Jika aplikasi Anda memerlukan izin tambahan, Anda harus menambahkannya ke kebijakan ini. Setiap definisi tugas hanya dapat menentukan satu peran IAM tugas. Misalnya, jika

Anda menggunakan file konfigurasi kustom yang disimpan di Systems Manager, Anda harus menambahkan `ssm:GetParameters` izin ke kebijakan IAM ini.

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari AWSDistroOpenTelemetryPolicyForXray, lalu pilih kebijakan.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

   a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

      IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

   b. Pilih kebijakan yang akan digunakan untuk batas izin.
8. Pilih Berikutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

   Important

   Saat Anda memberi nama peran, perhatikan hal berikut:

   - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.

     Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada

pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.

- Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tag di IAM, lihat Tag untuk AWS Identity and Access Management sumber daya di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Menentukan AWS Distro untuk OpenTelemetry sespan dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pengalaman membuat AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Gunakan koleksi metrik. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda secara manual. Contoh definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk integrasi Amazon. CloudWatch

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [
    {
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
```

```
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-cloudwatch.yaml"
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

Mengekspor metrik aplikasi ke Amazon Managed Service untuk Prometheus

Amazon ECS mendukung ekspor CPU tingkat tugas, memori, jaringan, dan metrik penyimpanan serta metrik aplikasi kustom Anda ke Amazon Managed Service for Prometheus. Ini dilakukan dengan menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda. Konsol Amazon ECS menyederhanakan proses ini dengan menambahkan opsi Gunakan koleksi metrik saat membuat definisi tugas baru. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Metrik diekspor ke Amazon Managed Service untuk Prometheus dan dapat dilihat menggunakan dasbor Grafana Terkelola Amazon. Aplikasi Anda harus diinstrumentasi dengan pustaka Prometheus atau dengan SDK. OpenTelemetry Untuk informasi selengkapnya tentang menginstrumentasi aplikasi Anda dengan OpenTelemetry SDK, lihat Pengantar AWS Distro untuk OpenTelemetry di Distro untuk dokumentasi AWS . OpenTelemetry

Saat menggunakan pustaka Prometheus, aplikasi Anda harus mengekspos titik akhir yang digunakan untuk mengikis `/metrics` data metrik. Untuk informasi selengkapnya tentang menginstrumentasi aplikasi Anda dengan pustaka Prometheus, lihat pustaka klien Prometheus dalam dokumentasi Prometheus.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan Amazon ECS pada integrasi Fargate AWS dengan Distro OpenTelemetry untuk mengirim metrik aplikasi ke Amazon Managed Service untuk Prometheus.

- AWS Distro untuk OpenTelemetry integrasi didukung untuk beban kerja Amazon ECS yang dihosting di beban kerja Fargate dan Amazon ECS yang dihosting di instans Amazon. EC2 Instance eksternal saat ini tidak didukung.
- Secara default, AWS Distro for OpenTelemetry menyertakan semua dimensi tingkat tugas yang tersedia untuk metrik aplikasi Anda saat mengekspor ke Amazon Managed Service untuk Prometheus. Anda juga dapat instrumen aplikasi Anda untuk menambahkan dimensi tambahan. Untuk informasi selengkapnya, lihat Memulai Prometheus Remote Write Exporter for Amazon Managed Service for Prometheus di Distro untuk dokumentasi. AWS OpenTelemetry

Izin IAM yang diperlukan untuk AWS Distro untuk integrasi OpenTelemetry dengan Amazon Managed Service untuk Prometheus

Integrasi Amazon ECS dengan Amazon Managed Service untuk Prometheus menggunakan AWS Distro OpenTelemetry untuk sespan mengharuskan Anda membuat peran IAM tugas dan menentukan peran dalam definisi tugas Anda. Peran IAM tugas ini harus dibuat secara manual menggunakan langkah-langkah di bawah ini sebelum mendaftarkan definisi tugas Anda.

Kami merekomendasikan bahwa AWS Distro untuk OpenTelemetry sespan juga dikonfigurasi untuk merutekan log kontainer ke CloudWatch Log yang memerlukan peran IAM eksekusi tugas dibuat dan ditentukan dalam definisi tugas Anda juga. Konsol Amazon ECS menangani peran IAM eksekusi tugas atas nama Anda, tetapi peran tugas IAM harus dibuat secara manual. Untuk informasi

selengkapnya tentang membuat peran IAM eksekusi tugas, lihatPeran IAM eksekusi tugas Amazon ECS.

> **Important**
>
> Jika Anda juga mengumpulkan data jejak aplikasi menggunakan AWS Distro untuk OpenTelemetry integrasi, pastikan peran IAM tugas Anda juga berisi izin yang diperlukan untuk integrasi tersebut. Untuk informasi selengkapnya, lihat Identifikasi peluang pengoptimalan Amazon ECS menggunakan data pelacakan aplikasi.

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari AmazonPrometheusRemoteWriteAccess, lalu pilih kebijakan.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

   a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

      IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

   b. Pilih kebijakan yang akan digunakan untuk batas izin.
8. Pilih Berikutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

   > **Important**
   >
   > Saat Anda memberi nama peran, perhatikan hal berikut:

- Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.

  Misalnya, jangan membuat peran bernama keduanya **PRODROLE** dan**prodrole**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
- Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tag di IAM, lihat Tag untuk AWS Identity and Access Management sumber daya di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Menentukan AWS Distro untuk OpenTelemetry sespan dalam definisi tugas Anda

Konsol Amazon ECS menyederhanakan pengalaman membuat AWS Distro untuk wadah OpenTelemetry sespan dengan menggunakan opsi Gunakan koleksi metrik. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.

Jika Anda tidak menggunakan konsol Amazon ECS, Anda dapat menambahkan AWS Distro untuk wadah OpenTelemetry sespan ke definisi tugas Anda secara manual. Contoh definisi tugas berikut menunjukkan definisi container untuk menambahkan AWS Distro untuk OpenTelemetry sespan untuk Amazon Managed Service untuk integrasi Prometheus.

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
```

```
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/aws-otel-emitter",
      "awslogs-region": "aws-region",
      "awslogs-stream-prefix": "ecs"
     }
    },
    "dependsOn": [{
     "containerName": "aws-otel-collector",
     "condition": "START"
    }]
   },
   {
    "name": "aws-otel-collector",
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
    "essential": true,
    "command": [
     "--config=/etc/ecs/ecs-amp.yaml"
    ],
    "environment": [{
     "name": "AWS_PROMETHEUS_ENDPOINT",
     "value": "https://aps-workspaces.aws-region.amazonaws.com/workspaces/
ws-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
    }],
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "aws-region",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
```

```
}
```

Log panggilan Amazon ECS API menggunakan AWS CloudTrail

Amazon Elastic Container Service terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk Amazon ECS sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol Amazon ECS dan panggilan kode ke operasi Amazon ECS API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon ECS, alamat IP dari mana permintaan itu dibuat, kapan dibuat, dan detail tambahan.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Anda Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. Wilayah AWS Untuk informasi selengkapnya, lihat Bekerja dengan riwayat CloudTrail Acara di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara CloudTrailDanau.

CloudTrail jalan setapak

> Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan AWS Management Console Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan. AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika

Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. Wilayah AWS Untuk informasi selengkapnya tentang jejak, lihat Membuat jejak untuk Anda Akun AWS dan Membuat jejak untuk organisasi di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga. Untuk informasi tentang harga Amazon S3, lihat Harga Amazon S3.

CloudTrail Penyimpanan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail Lake mengonversi peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC. ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara tingkat lanjut. Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat Bekerja dengan AWS CloudTrail Danau di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih opsi harga yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa, dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Acara manajemen Amazon ECS di CloudTrail

Acara manajemen memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di Anda Akun AWS. Ini juga dikenal sebagai operasi pesawat kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

Amazon Elastic Container Service mencatat semua operasi pesawat kontrol Amazon ECS sebagai peristiwa manajemen. Misalnya, panggilan ke`CreateService`, `RunTask` dan `DeleteCluster` bagian menghasilkan entri dalam file CloudTrail log. Untuk daftar operasi bidang kontrol Amazon Elastic Container Service yang dicatat oleh Amazon ECS CloudTrail, lihat Referensi API Amazon Elastic Container Service.

Contoh acara Amazon ECS

Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang operasi API yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga peristiwa tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail peristiwa yang menunjukkan `CreateService` tindakan.

> **Note**
>
> Contoh ini telah diformat untuk meningkatkan keterbacaan. Dalam file CloudTrail log, semua entri dan peristiwa digabungkan menjadi satu baris. Selain itu, contoh ini terbatas pada satu entri Amazon ECS. Dalam file CloudTrail log nyata, Anda melihat entri dan peristiwa dari beberapa AWS layanan.


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-20T18:32:25Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Mary_Major"
            }
        }
    },
    "eventTime": "2018-06-20T19:04:36Z",
    "eventSource": "ecs.amazonaws.com",
```

```
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "clusterName": "default"
    },
    "responseElements": {
        "cluster": {
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
            "pendingTasksCount": 0,
            "registeredContainerInstancesCount": 0,
            "status": "ACTIVE",
            "runningTasksCount": 0,
            "statistics": [],
            "clusterName": "default",
            "activeServicesCount": 0
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Untuk informasi tentang konten CloudTrail rekaman, lihat konten CloudTrail rekaman di Panduan AWS CloudTrail Pengguna.

Melihat CloudWatch Log Live Tail untuk layanan dan tugas Amazon ECS

CloudWatch Logs Live Tail membantu Anda memecahkan masalah insiden dengan cepat dengan melihat daftar streaming peristiwa log baru saat tertelan. Anda dapat melihat peristiwa tugas dan layanan di konsol Amazon ECS. Pandangan ini memberikan pandangan yang kohesif tentang tugas dan layanan kesehatan Anda.

Setiap tugas dalam layanan Amazon ECS memiliki aliran log khusus untuk setiap kontainer. Jika layanan meningkat, setiap instance tugas memiliki kumpulan aliran lognya sendiri. Konvensi penamaan untuk aliran log mengikuti pola:

```
log-group-prefix/container-name/task-id
```

Satu tugas menulis ke aliran log yang sama selama masa pakainya. Aliran log berisi pesan dari wadah tugas itu dan juga output apa pun dari kode aplikasi Anda. Setiap pesan diberi cap waktu, termasuk log kustom Anda.

> **Note**
>
> Sesi Live Tail dikenakan biaya berdasarkan waktu penggunaan sesi, per menit. Untuk informasi selengkapnya tentang harga, lihat CloudWatch Harga Amazon.

Izin yang diperlukan

Izin berikut diperlukan bagi pengguna IAM konsol untuk memulai dan menghentikan sesi CloudWatch Log Live Tail:

- `logs:DescribeLogGroups`
- `logs:StartLiveTail`
- `logs:StopLiveTail`

Prosedur

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Tentukan sumber daya yang ingin Anda lihat

Sumber Daya	Langkah-langkah
Tugas	a. Pada halaman Clusters, pilih cluster. Halaman detail cluster ditampilkan. b. Pilih tab Tugas. Halaman detail tugas ditampilkan.

Sumber Daya	Langkah-langkah
	c. Pilih tab Log.
Layanan	a. Pada halaman Clusters, pilih cluster. Halaman detail cluster ditampilkan. b. Pilih layanan. Halaman detail layanan ditampilkan. c. Pilih tab Log.

3. Pilih CloudWatch Log Live Tail, lalu pilih Mulai.
4. (Opsional) Untuk memfilter aliran, di bawah Filter, untuk Pilih grup log, pilih grup log.

AWS CLI referensi

Anda juga dapat menggunakan AWS CLI untuk memulai sesi Live Tail untuk CloudWatch Log. Sumber daya berikut memberikan informasi terperinci tentang penggunaan AWS CLI dengan Live Tail:

- start-live-tail referensi perintah - Sintaks perintah lengkap, parameter, dan contoh untuk `aws logs start-live-tail` perintah.
- CloudWatch Panduan pengguna Logs Live Tail - Panduan komprehensif termasuk AWS CLI penggunaan dengan mode khusus cetak dan interaktif.
- StartLiveTail Contoh SDK - Contoh terprogram untuk menggunakan StartLiveTail API dengan berbagai. AWS SDKs

Pantau beban kerja menggunakan metadata Amazon ECS

Anda dapat menggunakan metadata tugas dan kontainer untuk memecahkan masalah beban kerja Anda dan membuat perubahan konfigurasi berdasarkan lingkungan runtime.

Metadata mencakup kategori berikut:

- Atribut tingkat tugas yang memberikan informasi tentang di mana tugas berjalan.
- Atribut tingkat kontainer yang menyediakan ID Docker, nama, dan detail gambar.

  Ini memberikan visibilitas ke dalam wadah.
- Pengaturan jaringan seperti alamat IP, subnet, dan mode jaringan.

  Ini membantu dengan konfigurasi jaringan dan pemecahan masalah.
- Status tugas dan kesehatan

  Ini memungkinkan Anda mengetahui apakah tugas sedang berjalan.

Anda dapat melihat metadata dengan salah satu metode berikut:

- File metadata kontainer

  Dimulai dengan versi 1.15.0 dari agen penampung Amazon ECS, berbagai metadata kontainer tersedia di dalam kontainer Anda atau instans penampung host. Dengan mengaktifkan fitur ini, Anda dapat melakukan kueri informasi tentang tugas, kontainer, dan instans kontainer dari dalam kontainer atau instans kontainer host. File metadata dibuat pada instance host dan dipasang di wadah sebagai volume Docker dan oleh karena itu tidak tersedia saat tugas di-host di Fargate. AWS
- Titik akhir metadata tugas

  Agen kontainer Amazon ECS menyuntikkan variabel lingkungan ke dalam setiap kontainer, yang disebut sebagai titik akhir metadata tugas yang menyediakan berbagai metadata tugas dan statistik Docker ke wadah.
- Introspeksi wadah

  Agen penampung Amazon ECS menyediakan operasi API untuk mengumpulkan detail tentang instance kontainer tempat agen dijalankan dan tugas terkait yang berjalan pada instance tersebut.

Variabel lingkungan Amazon ECS

Variabel lingkungan mengontrol perilaku tugas seperti kapasitas yang digunakan untuk menjalankan tugas. Amazon ECS menetapkan variabel lingkungan berikut untuk tugas Anda:

- `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`- Titik akhir URL HTTP lengkap untuk SDK yang akan digunakan saat membuat permintaan kredensil. Ini termasuk skema dan tuan rumah.

Untuk informasi selengkapnya, lihat Penyedia kredensi kontainer di Panduan AWS SDKs Referensi.

- `ECS_CONTAINER_METADATA_URI_V4`- Alamat metadata tugas versi 4. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.
- `ECS_CONTAINER_METADATA_URI`- Alamat metadata tugas versi 3. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 3.
- `ECS_AGENT_URI`- Alamat dasar untuk berbagai titik akhir yang didukung oleh Fargate. Untuk informasi selengkapnya, lihat:
  - Titik akhir perlindungan skala tugas Amazon ECS
  - Titik akhir injeksi kesalahan Amazon ECS
- `AWS_EXECUTION_ENV`- Informasi tentang jenis peluncuran tugas yang dijalankan.
  - Untuk Fargate, Amazon ECS menetapkan ini ke. `AWS_ECS_FARGATE`
  - Untuk EC2, Amazon ECS menetapkan ini ke`AWS_ECS_EC2`.
- `AWS_DEFAULT_REGION`— Default Wilayah AWS untuk Anda Akun AWS. Ini adalah Wilayah default tempat tugas Anda berjalan.
- `AWS_REGION`— Wilayah tempat tugas berjalan. Jika didefinisikan, nilai ini mengesampingkan. `AWS_DEFAULT_REGION`

Anda dapat melihat variabel lingkungan dalam metadata tugas. Untuk informasi selengkapnya, lihat salah satu topik berikut:

- Fargate - Metadata tugas Amazon ECS tersedia untuk tugas di Fargate
- EC2 - Metadata tugas tersedia untuk tugas Amazon ECS EC2

File metadata wadah Amazon ECS

Dimulai dengan versi 1.15.0 dari agen penampung Amazon ECS, berbagai metadata kontainer tersedia di dalam kontainer Anda atau instans penampung host. Dengan mengaktifkan fitur ini, Anda dapat melakukan kueri informasi tentang tugas, kontainer, dan instans kontainer dari dalam kontainer atau instans kontainer host. File metadata dibuat pada instance host dan dipasang di wadah sebagai volume Docker dan oleh karena itu tidak tersedia saat tugas di-host di Fargate. AWS

File metadata kontainer dibersihkan pada instans host ketika kontainer dibersihkan.
Anda dapat menyesuaikan kapan hal ini terjadi dengan variabel agen kontainer

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`. Untuk informasi selengkapnya, lihat Tugas Amazon ECS otomatis dan pembersihan gambar.

Topik

- Lokasi file metadata kontainer
- Mengaktifkan metadata kontainer Amazon ECS
- Format file metadata wadah Amazon ECS

Lokasi file metadata kontainer

Secara default, file metadata kontainer ditulis ke jalur host dan kontainer berikut.

- Untuk instance Linux:
  - Jalur host: `/var/lib/ecs/data/metadata/`*`cluster_name`*`/`*`task_id`*`/`*`container_name`*`/ecs-container-metadata.json`

    > **Note**
    >
    > Jalur host Linux mengasumsikan bahwa jalur pemasangan direktori data default (`/var/lib/ecs/data`) digunakan saat agen dimulai. Jika Anda tidak menggunakan AMI yang dioptimalkan Amazon ECS (atau `ecs-init` paket untuk memulai dan memelihara agen penampung), pastikan untuk menyetel variabel konfigurasi `ECS_HOST_DATA_DIR` agen ke jalur host tempat file status agen penampung berada. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

  - Jalur kontainer: `/opt/ecs/metadata/`*`random_ID`*`/ecs-container-metadata.json`
- Untuk contoh Windows:
  - Jalur host: `C:\ProgramData\Amazon\ECS\data\metadata\`*`task_id`*`\`*`container_name`*`\ecs-container-metadata.json`
  - Jalur kontainer: `C:\ProgramData\Amazon\ECS\metadata\`*`random_ID`*`\ecs-container-metadata.json`

Namun, untuk memudahkan akses, lokasi file metadata kontainer diatur ke variabel lingkungan `ECS_CONTAINER_METADATA_FILE` di dalam kontainer. Anda dapat membaca isi file dari dalam kontainer dengan perintah berikut:

- Untuk instance Linux:

```
cat $ECS_CONTAINER_METADATA_FILE
```

- Untuk instance Windows (PowerShell):

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

Mengaktifkan metadata kontainer Amazon ECS

Anda dapat mengaktifkan metadata kontainer di tingkat instance kontainer dengan menyetel variabel agen `ECS_ENABLE_CONTAINER_METADATA` kontainer ke. `true` Anda dapat mengatur variabel ini di file konfigurasi `/etc/ecs/ecs.config` dan mulai ulang agen. Anda juga dapat mengaturnya sebagai variabel lingkungan Docker pada saat waktu aktif ketika kontainer agen dimulai. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Jika `ECS_ENABLE_CONTAINER_METADATA` diatur ke `true` ketika agen dimulai, file metadata dibuat untuk setiap kontainer yang dibuat sejak saat itu. Agen penampung Amazon ECS tidak dapat membuat file metadata untuk kontainer yang dibuat sebelum variabel agen `ECS_ENABLE_CONTAINER_METADATA` kontainer disetel ke. `true` Untuk memastikan bahwa semua kontainer menerima file metadata, Anda harus mengatur variabel agen ini pada peluncuran instans kontainer. Berikut ini adalah contoh skrip data pengguna yang akan mengatur variabel ini serta mendaftarkan instans kontainer dengan klaster Anda.

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_ENABLE_CONTAINER_METADATA=true
EOF
```

Format file metadata wadah Amazon ECS

Informasi berikut disimpan dalam file JSON metadata kontainer.

`Cluster`

Nama klaster tempat tugas kontainer sedang berjalan.

`ContainerInstanceARN`

Nama lrngkap Amazon Resource Name (ARN) dari instans kontainer host.

`TaskARN`

Nama lengkap Amazon Resource Name (ARN) dari tugas milik kontainer.

`TaskDefinitionFamily`

Nama famili ketentuan tugas yag digunakan kontainer.

`TaskDefinitionRevision`

Revisi ketentuan tugas yang digunakan kontainer.

`ContainerID`

ID kontainer Docker (dan bukan ID penampung Amazon ECS) untuk penampung.

`ContainerName`

Nama kontainer dari definisi tugas Amazon ECS untuk penampung.

`DockerContainerName`

Nama kontainer yang digunakan daemon Docker untuk kontainer (misalnya, nama yang muncul di output perintah docker ps).

`ImageID`

SHA digest untuk citra Docker digunakan untuk memulai kontainer.

`ImageName`

Nama citra dan tanda untuk citra Docker yang digunakan untuk memulai kontainer.

`PortMappings`

Setiap pemetaan port yang terkait dengan kontainer.

`ContainerPort`

Port pada kontainer yang diekspos.

`HostPort`

Port pada instans kontainer host yang diekspos.

`BindIp`

Alamat IP yang mengikat yang ditetapkan untuk kontainer oleh Docker. Alamat IP ini hanya diterapkan dengan mode jaringan `bridge`, dan hanya dapat diakses dari instans kontainer.

`Protocol`

Protokol jaringan yang digunakan untuk pemetaan port.

`Networks`

Mode jaringan dan alamat IP untuk kontainer.

`NetworkMode`

Mode jaringan untuk tugas milik kontainer.

`IPv4Addresses`

Alamat IP yang terkait dengan kontainer.

> **Important**
>
> Jika tugas Anda menggunakan mode jaringan `awsvpc`, alamat IP kontainer tidak akan dikembalikan. Dalam hal ini, Anda dapat mengambil alamat IP dengan membaca file /etc/host dengan perintah berikut:
>
> ```
> tail -1 /etc/hosts | awk '{print $1}'
> ```

`MetadataFileStatus`

Status file metadata. Bila statusnya `READY`, file metadata baru dan lengkap. Jika file belum siap (misalnya, saat tugas dimulai), versi format file yang terpotong tersedia. Untuk menghindari kemungkinan kondisi pacu saat kontainer dimulai, namun metadata belum ditulis, Anda dapat mengurai file metadata dan menunggu parameter ini untuk diatur ke `READY` sebelum tergantung pada metadata. Hal ini biasanya tersedia dalam waktu kurang dari 1 detik sejak saat kontainer dimulai.

`AvailabilityZone`

Availability Zone instans kontainer host berada di.

`HostPrivateIPv4Address`

Alamat IP privat untuk tugas milik kontainer.

`HostPublicIPv4Address`

Alamat IP publik untuk tugas milik kontainer.

Example File metadata wadah Amazon ECS () **`READY`**

Contoh berikut menunjukkan file metadata kontainer di status `READY`.

```
{
    "Cluster":"arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
    "TaskARN":"arn:aws:ecs:us-east-1:123456789012:task/MyCluster/
b593651c4d6b44a6b2b583f45c957e15",
    "Family":"curltest-container",
    "Revision":"2",
    "DesiredStatus":"RUNNING",
    "KnownStatus":"RUNNING",
    "Limits":
        {
            "CPU":0.25,
            "Memory":512
        },
    "PullStartedAt":"2025-01-17T20:56:17.394610044Z",
    "PullStoppedAt":"2025-01-17T20:56:25.282708213Z",
    "AvailabilityZone":"us-east-1b",
    "LaunchType":"FARGATE",
    "Containers":[
        {
            "DockerId":"b593651c4d6b44a6b2b583f45c957e15-3356213583",
            "Name":"curltest","DockerName":"curltest",
            "Image":"public.ecr.aws/amazonlinux/amazonlinux:latest",

 "ImageID":"sha256:7f371357694782356b65c7fd60dd1ca124c47bd5ed1b1ffe7c0e17f562898367",
            "Labels":
                {
                    "com.amazonaws.ecs.cluster":"arn:aws:ecs:us-
east-1:123456789012:cluster/MyCluster",
                    "com.amazonaws.ecs.container-name":"curltest",
                    "com.amazonaws.ecs.task-arn":"arn:aws:ecs:us-
east-1:123456789012:task/MyCluster/b593651c4d6b44a6b2b583f45c957e15",
```

```
                    "com.amazonaws.ecs.task-definition-family":"curltest-
container","com.amazonaws.ecs.task-definition-version":"2"
                },
            "DesiredStatus":"RUNNING",
            "KnownStatus":"RUNNING",
            "Limits":
                {
                    "CPU":2
                },
            "CreatedAt":"2025-01-17T20:56:26.180347056Z",
            "StartedAt":"2025-01-17T20:56:26.180347056Z",
            "Type":"NORMAL",
            "LogDriver":"awslogs",
            "LogOptions":
                {
                    "awslogs-create-group":"true",
                    "awslogs-group":"/ecs/curltest-container",
                    "awslogs-region":"us-east-1",
                    "awslogs-stream":"ecs/curltest/b593651c4d6b44a6b2b583f45c957e15"
                       },
            "ContainerARN":"arn:aws:ecs:us-east-1:123456789012:container/MyCluster/
b593651c4d6b44a6b2b583f45c957e15/934575e8-5bdb-478f-b763-2341a85b690e",
            "Networks":[
                {
                    "NetworkMode":"awsvpc",
                    "IPv4Addresses":["10.0.1.58"]
                }
            ],
            "Snapshotter":"overlayfs"
        }
    ],
    "ClockDrift":
        {
            "ClockErrorBound":0.487801,"ReferenceTimestamp":"2025-01-17T20:56:02Z",
            "ClockSynchronizationStatus":"SYNCHRONIZED"
        },
    "FaultInjectionEnabled":false
}
```

Example File metadata kontainer Amazon ECS tidak lengkap (belum) **READY**

Contoh berikut menunjukkan file metadata kontainer yang belum mencapai status READY. Informasi dalam file terbatas untuk beberapa parameter yang diketahui dari ketentuan tugas. File metadata kontainer harus siap dalam 1 detik setelah kontainer dimulai.

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
}
```

Metadata tugas tersedia untuk tugas Amazon ECS EC2

Agen penampung Amazon ECS menyediakan metode untuk mengambil berbagai metadata tugas dan statistik Docker. Hal Ini disebut sebagai titik akhir metadata tugas. Versi berikut tersedia:

- Titik akhir metadata tugas versi 4 — Menyediakan berbagai metadata dan statistik Docker ke kontainer. Juga dapat menyediakan data tingkat jaringan. Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.39.0` penampung Amazon ECS. Untuk instans Amazon EC2 Windows yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.
- Titik akhir metadata tugas versi 3 — Menyediakan berbagai metadata dan statistik Docker ke kontainer. Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.21.0` penampung Amazon ECS. Untuk instans Amazon EC2 Windows yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 3.
- Titik akhir metadata tugas versi 2 — Tersedia untuk tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.17.0` penampung Amazon ECS. Untuk instans Amazon EC2 Windows yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 2.

Jika tugas Amazon ECS di-host di Amazon EC2, Anda juga dapat mengakses metadata host tugas menggunakan titik akhir Layanan Metadata Instans (IMDS). Perintah berikut, ketika dijalankan dari dalam instance hosting tugas, mencantumkan ID dari instance host.

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

Informasi yang dapat Anda peroleh dari titik akhir dibagi ke dalam kategori seperti`instance-id`. Untuk informasi selengkapnya tentang berbagai kategori metadata instans host yang dapat Anda peroleh menggunakan titik akhir, lihat Kategori metadata instans.

Titik akhir metadata tugas Amazon ECS versi 4

Agen kontainer Amazon ECS menyuntikkan variabel lingkungan ke dalam setiap kontainer, yang disebut sebagai titik akhir metadata tugas yang menyediakan berbagai metadata tugas dan statistik Docker ke wadah.

Metadata tugas dan statistik tingkat jaringan dikirim ke CloudWatch Wawasan Kontainer dan dapat dilihat di. AWS Management Console Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.

> **Note**
>
> Amazon ECS menyediakan versi sebelumnya dari titik akhir metadata tugas. Untuk menghindari kebutuhan membuat versi titik akhir metadata tugas di masa mendatang, metadata tambahan dapat ditambahkan ke output versi 4. Kami tidak akan menghapus metadata yang ada atau mengubah nama bidang metadata.

Variabel lingkungan disuntikkan secara default ke dalam wadah tugas Amazon ECS yang diluncurkan di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.39.0` penampung Amazon ECS. Untuk instans Amazon EC2 Windows yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi. `1.54.0` Untuk informasi selengkapnya, lihat Manajemen instans penampung Amazon ECS Linux.

> **Note**
>
> Anda dapat menambahkan dukungan untuk fitur ini di EC2 instans Amazon menggunakan versi lama agen penampung Amazon ECS dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

Untuk keluaran contoh metadata tugas, lihat contoh metadata tugas Amazon ECS v4.

Jalur titik akhir metadata tugas versi 4

Jalur titik akhir metadata tugas berikut tersedia untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}`

Jalur ini mengembalikan metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}/task`

Jalur ini mengembalikan metadata untuk tugas, termasuk daftar wadah IDs dan nama untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS V4 Respons JSON.

`${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`

Jalur ini mengembalikan metadata untuk tugas termasuk dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`. Setiap kesalahan yang diterima saat mengambil metadata tanda akan disertakan dalam kolom `Errors` sebagai responsnya.

> **Note**
>
> `Errors`Bidang ini hanya dalam respons untuk tugas yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya versi agen `1.50.0` penampung. Untuk instans Amazon EC2 Windows yang menggunakan mode `awsvpc` jaringan, agen penampung Amazon ECS harus setidaknya versi `1.54.0`
> Titik akhir ini membutuhkan `ecs.ListTagsForResource` izin.

> **Important**
>
> Saat menggunakan `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags` titik akhir, ketahuilah bahwa setiap panggilan membuat hingga dua permintaan API `ecs:ListTagsForResource` (satu untuk tag instance kontainer dan satu untuk tag tugas) dan bahwa setiap wadah sespan dalam tugas dapat melakukan panggilan ini atas nama Anda. Panggilan endpoint yang sering dapat mengakibatkan pelambatan API. Pertimbangkan untuk menerapkan strategi caching atau batching untuk mengurangi frekuensi panggilan, terutama dalam aplikasi dengan lalu lintas tinggi, dan men-debug masalah pelambatan API yang digunakan. AWS CloudTrail Untuk informasi tentang batas pembatasan untuk `ListTagsForResource` API, lihat Meminta pembatasan untuk Amazon ECS API di Referensi API Amazon Elastic Container Service. Untuk informasi selengkapnya tentang men-debug panggilan Amazon ECS API menggunakan AWS CloudTrail, lihat. Log panggilan Amazon ECS API menggunakan AWS CloudTrail

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

Jalur ini mengembalikan statistik Docker untuk kontainer tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya versi `1.43.0` agen kontainer, akan ada statistik tarif jaringan tambahan yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

Jalan ini mengembalikan statistik Docker untuk semua kontainer yang terkait dengan tugas. Hal ini dapat digunakan oleh kontainer sidecar untuk mengekstrak metrik jaringan. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di instans Amazon EC2 Linux yang menjalankan setidaknya versi `1.43.0` agen kontainer, akan ada statistik tarif jaringan tambahan yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

Metadata tugas Amazon ECS V4 Respons JSON

Informasi berikut dikembalikan dari respons JSON (`${ECS_CONTAINER_METADATA_URI_V4}/task`) titik akhir metadata tugas. Hal ini termasuk metadata yang terkait dengan tugas selain metadata untuk setiap kontainer dalam tugas.

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`ServiceName`

Nama layanan tempat tugas itu berada. ServiceName akan muncul untuk instance penampung Amazon EC2 dan Amazon ECS Anywhere jika tugas dikaitkan dengan layanan.

> **Note**
>
> `ServiceName`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`VPCID`

ID VPC dari instans EC2 penampung Amazon. Bidang ini hanya muncul untuk EC2 instance Amazon.

> **Note**
>
> `VPCID`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`TaskARN`

Nama Sumber Daya Amazon (ARN) dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPUs) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 (Windows).

`LaunchType`

Jenis peluncuran yang digunakan tugas. Saat menggunakan penyedia kapasitas cluster, ini menunjukkan apakah tugas tersebut menggunakan Fargate atau EC2 infrastruktur.

> **Note**
>
> `LaunchType`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux versi agen container atau yang lebih baru (Linux) `1.45.0` atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

Saat Anda menggunakan Fargate, id adalah hex 32 digit diikuti dengan angka 10 digit.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 dari manifes gambar. Ini adalah intisari yang dapat digunakan untuk menarik gambar menggunakan format`repository-url/image@sha256:digest`.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`LogDriver`

Driver log yang digunakan wadah.

> **Note**
>
> `LogDriver`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`LogOptions`

Opsi driver log yang ditentukan untuk wadah.

> **Note**
>
> `LogOptions`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`ContainerARN`

Nama Sumber Daya Amazon (ARN) dari wadah.

> **Note**
>
> `ContainerARN`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`RestartCount`

Berapa kali wadah telah dimulai ulang.

> **Note**
>
> `RestartCount`Metadata hanya disertakan jika kebijakan restart diaktifkan untuk penampung. Untuk informasi selengkapnya, lihat Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v4

Contoh berikut menampilkan output contoh dari masing-masing titik akhir metadata tugas.

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah contoh output dari tugas yang berjalan sebagai bagian dari layanan (MyService).

```
{
```

```
    "DockerId": "ea32192c8553fbff06c9340478a2ff089b2bb5646fb718b4ee206641c9086d66",
    "Name": "curl",
    "DockerName": "ecs-curltest-24-curl-cca48e8dcadd97805600",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/8f03e41243824aea923aca126495f665",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "24"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-02T00:15:07.620912337Z",
    "StartedAt": "2020-10-02T00:15:08.062559351Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/metadata",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/8f03e41243824aea923aca126495f665"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/0206b271-
b33f-47ab-86c6-a0ba208a70a9",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.100"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0e:9e:32:c7:48:85",
            "IPv4SubnetCIDRBlock": "10.0.2.0/24",
            "PrivateDNSName": "ip-10-0-2-100.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "10.0.2.1/24"
        }
```

```
    ]
}
```

Respons metadata tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task`, metadata dari tugas yang berisi bagian kontainer selain metada untuk masing-masing kontainer dalam tugas dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
```

```
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
```

```
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Tugas contoh dengan respons metadata tanda

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`, metadata dari tugas yang berisi tugas dan tanda instans kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
```

```
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "TaskTags": {
        "tag-use": "task-metadata-endpoint-test"
    },
    "ContainerInstanceTags":{
        "tag_key":"tag_value"
    },
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
```

```
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
```

```
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Tugas contoh dengan tanda yang menggambarkan respons metadata kesalahan

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`, metadata dari tugas yang berisi tugas dan tanda instans kontainer dikembalikan kepada Anda. Jika terjadi kesalahan saat mengambil data penandaan, kesalahan dikembalikan dalam bentuk respons. Berikut ini adalah contoh keluaran ketika peran IAM yang terkait dengan instance container tidak memiliki `ecs:ListTagsForResource` izin yang diizinkan.

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "Errors": [
        {
            "ErrorField": "ContainerInstanceTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:container-
instance/default/2dd1b186f39845a584488d2ef155c131",
            "StatusCode": 400,
            "RequestId": "cd597ef0-272b-4643-9bd2-1de469870fa6",
```

```
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
        },
        {
            "ErrorField": "TaskTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3",
            "StatusCode": 400,
            "RequestId": "862c5986-6cd2-4aa6-87cc-70be395531e1",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3"
        }
    ],
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
```

```
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
```

```
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

Respons statistik kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/stats`, metrik jaringan untuk kontainer dikembalikan kepada Anda. Untuk tugas Amazon ECS yang menggunakan mode `bridge` jaringan `awsvpc` atau yang dihosting di EC2 instans Amazon yang menjalankan setidaknya versi agen `1.43.0` penampung, akan ada statistik tarif jaringan tambahan yang disertakan dalam respons. Untuk semua tugas lainnya, respons hanya akan mencakup statistik jaringan kumulatif.

Berikut ini adalah contoh output dari tugas Amazon ECS di Amazon EC2 yang menggunakan mode `bridge` jaringan.

```
{
    "read": "2020-10-02T00:51:13.410254284Z",
    "preread": "2020-10-02T00:51:12.406202398Z",
    "pids_stats": {
        "current": 3
    },
    "blkio_stats": {
        "io_service_bytes_recursive": [

        ],
        "io_serviced_recursive": [
```

```
        ],
        "io_queue_recursive": [

        ],
        "io_service_time_recursive": [

        ],
        "io_wait_time_recursive": [

        ],
        "io_merged_recursive": [

        ],
        "io_time_recursive": [

        ],
        "sectors_recursive": [

        ]
    },
    "num_procs": 0,
    "storage_stats": {

    },
    "cpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13939680000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "precpu_stats": {
```

```
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13937670000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "memory_stats": {
        "usage": 1806336,
        "max_usage": 6299648,
        "stats": {
            "active_anon": 606208,
            "active_file": 0,
            "cache": 0,
            "dirty": 0,
            "hierarchical_memory_limit": 134217728,
            "hierarchical_memsw_limit": 268435456,
            "inactive_anon": 0,
            "inactive_file": 0,
            "mapped_file": 0,
            "pgfault": 4185,
            "pgmajfault": 0,
            "pgpgin": 2926,
            "pgpgout": 2778,
            "rss": 606208,
            "rss_huge": 0,
            "total_active_anon": 606208,
            "total_active_file": 0,
            "total_cache": 0,
            "total_dirty": 0,
            "total_inactive_anon": 0,
            "total_inactive_file": 0,
            "total_mapped_file": 0,
            "total_pgfault": 4185,
```

```
            "total_pgmajfault": 0,
            "total_pgpgin": 2926,
            "total_pgpgout": 2778,
            "total_rss": 606208,
            "total_rss_huge": 0,
            "total_unevictable": 0,
            "total_writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "limit": 134217728
    },
    "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
    "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
    "networks": {
        "eth0": {
            "rx_bytes": 84,
            "rx_packets": 2,
            "rx_errors": 0,
            "rx_dropped": 0,
            "tx_bytes": 84,
            "tx_packets": 2,
            "tx_errors": 0,
            "tx_dropped": 0
        }
    },
    "network_rate_stats": {
        "rx_bytes_per_sec": 0,
        "tx_bytes_per_sec": 0
    }
}
```


Respons statistik tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task/stats`, metrik jaringan dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.


```
{
    "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854": {
        "read": "2020-10-02T00:51:32.51467703Z",
        "preread": "2020-10-02T00:51:31.50860463Z",
        "pids_stats": {
```

```
            "current": 1
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13977820000000,
            "online_cpus": 2,
```

```
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 532480,
            "max_usage": 6279168,
            "stats": {
                "active_anon": 40960,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 9223372036854771712,
                "hierarchical_memsw_limit": 9223372036854771712,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 2033,
                "pgmajfault": 0,
                "pgpgin": 1734,
                "pgpgout": 1724,
                "rss": 40960,
                "rss_huge": 0,
                "total_active_anon": 40960,
```

```
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 2033,
                "total_pgmajfault": 0,
                "total_pgpgin": 1734,
                "total_pgpgout": 1724,
                "total_rss": 40960,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 4073377792
        },
        "name": "/ecs-curltest-26-internalecspause-a6bcc3dbadfacfe85300",
        "id": "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    },
    "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af": {
        "read": "2020-10-02T00:51:32.512771349Z",
        "preread": "2020-10-02T00:51:31.510597736Z",
        "pids_stats": {
            "current": 3
        },
```

```
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 379075681,
                "percpu_usage": [
                    191355275,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13977800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
```

```
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 378825197,
                "percpu_usage": [
                    191104791,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 1814528,
            "max_usage": 6299648,
            "stats": {
                "active_anon": 606208,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 134217728,
                "hierarchical_memsw_limit": 268435456,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 5377,
                "pgmajfault": 0,
                "pgpgin": 3613,
                "pgpgout": 3465,
                "rss": 606208,
                "rss_huge": 0,
                "total_active_anon": 606208,
                "total_active_file": 0,
                "total_cache": 0,
```

```
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 5377,
                "total_pgmajfault": 0,
                "total_pgpgin": 3613,
                "total_pgpgout": 3465,
                "total_rss": 606208,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 134217728
        },
        "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
        "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    }
}
```

Titik akhir metadata tugas Amazon ECS versi 3

> **Important**
>
> Titik akhir metadata tugas versi 3 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.
>
> Jika Anda menggunakan tugas Amazon ECS yang dihosting AWS Fargate, lihat titik akhir metadata tugas Amazon ECS versi 3 untuk tugas di Fargate.

Dimulai dengan versi 1.21.0 dari agen kontainer Amazon ECS, agen menyuntikkan variabel lingkungan yang dipanggil `ECS_CONTAINER_METADATA_URI` ke setiap kontainer dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 3, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas. Untuk tugas yang menggunakan mode jaringan `bridge`, metrik jaringan tersedia ketika melakukan kueri terhadap titik akhir `/stats`.

Fitur titik akhir metadata tugas versi 3 diaktifkan secara default untuk tugas yang menggunakan jenis peluncuran Fargate pada platform versi v1.3.0 atau yang lebih baru dan tugas yang menggunakan jenis EC2 peluncuran dan diluncurkan di infrastruktur Amazon EC2 Linux yang menjalankan setidaknya versi 1.21.0 dari agen penampung Amazon ECS atau di EC2 infrastruktur Amazon Windows yang menjalankan setidaknya versi agen penampung Amazon ECS dan menggunakan mode `1.54.0` jaringan. `awsvpc` Untuk informasi selengkapnya, lihat Manajemen instans penampung Amazon ECS Linux.

Anda dapat menambahkan dukungan untuk fitur ini pada instans kontainer yang lebih lama dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

> **Important**
>
> Untuk tugas yang menggunakan tipe peluncuran Fargate dan versi platform sebelum v1.3.0, titik akhir metadata tugas versi 2 didukung. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 2.

Jalur titik akhir Metadata Tugas versi 3

Titik akhir metadata tugas berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI}`

Jalur ini mengembalikan JSON metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI}/task`

Jalur ini mengembalikan metadata JSON untuk tugas, termasuk daftar wadah IDs dan nama untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v3 respons JSON.

`${ECS_CONTAINER_METADATA_URI}/taskWithTags`

Jalur ini mengembalikan metadata pada tugas yang disertakan dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`.

`${ECS_CONTAINER_METADATA_URI}/stats`

Jalur ini mengembalikan JSON statistik Docker pada kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`${ECS_CONTAINER_METADATA_URI}/task/stats`

Jalur ini mengembalikan JSON statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Metadata tugas Amazon ECS v3 respons JSON

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPUs) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 dari manifes gambar. Ini adalah intisari yang dapat digunakan untuk menarik gambar menggunakan format`repository-url/image@sha256:digest`.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Layanan Sinkronisasi Waktu Amazon untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan EC2 Pengguna Amazon untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v3

Contoh berikut menampilkan output sampel dari titik akhir metadata tugas.

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
    "Name": "nginx-curl",
    "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
    "Image": "nrdlngr/nginx-curl",
    "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 512,
        "Memory": 512
    },
    "CreatedAt": "2018-02-01T20:55:10.554941919Z",
    "StartedAt": "2018-02-01T20:55:11.064236631Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.106"
```

```
            ]
        }
    ]
}
```

Respons metadata tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI}/task`, metadata dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
```

```
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
```

```
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Titik akhir metadata tugas Amazon ECS versi 2

> **Important**
>
> Titik akhir metadata tugas versi 2 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.

Dimulai dengan versi 1.17.0 dari agen penampung Amazon ECS, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS.

Semua kontainer milik tugas yang diluncurkan dengan mode `awsvpc` jaringan menerima IPv4 alamat lokal dalam rentang alamat link-lokal yang telah ditentukan. Saat kontainer menanyakan titik akhir metadata, agen penampung Amazon ECS dapat menentukan tugas mana yang dimiliki penampung berdasarkan alamat IP uniknya, serta metadata serta statistik untuk tugas tersebut dikembalikan.

Mengaktifkan metadata tugas

Fitur metadata tugas versi 2 diaktifkan secara default untuk hal berikut:

- Tugas menggunakan jenis peluncuran Fargate yang menggunakan platform versi v1.1.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.
- Tugas menggunakan jenis EC2 peluncuran yang juga menggunakan mode `awsvpc` jaringan dan diluncurkan pada infrastruktur Amazon EC2 Linux yang menjalankan setidaknya versi 1.17.0 dari agen penampung Amazon ECS atau di infrastruktur Amazon EC2 Windows yang menjalankan setidaknya versi agen `1.54.0` penampung Amazon ECS. Untuk informasi selengkapnya, lihat Manajemen instans penampung Amazon ECS Linux.

Anda dapat menambahkan dukungan untuk fitur ini pada instans kontainer yang lebih lama dengan memperbarui agen ke versi terbaru. Untuk informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

Jalur titik akhir metadata tugas

Titik akhir API berikut tersedia untuk kontainer:

`169.254.170.2/v2/metadata`

Titik akhir ini mengembalikan metadata JSON untuk tugas, termasuk daftar penampung IDs dan nama untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Respons JSON metadata tugas.

`169.254.170.2/v2/metadata/<container-id>`

Titik akhir ini mengembalikan JSON metadata untuk ID kontainer Docker ditentukan.

`169.254.170.2/v2/metadata/taskWithTags`

Jalur ini mengembalikan metadata pada tugas yang disertakan dalam titik akhir `/task` selain tugas dan tanda instans kontainer yang dapat diambil menggunakan API `ListTagsForResource`.

`169.254.170.2/v2/stats`

Titik akhir ini mengembalikan JSON statistik Docker pada semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`169.254.170.2/v2/stats/<container-id>`

Titik akhir ini mengembalikan JSON statistik Docker untuk ID kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Respons JSON metadata tugas

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`169.254.170.2/v2/metadata`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPUs) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 dari manifes gambar. Ini adalah intisari yang dapat digunakan untuk menarik gambar menggunakan format`repository-url/image@sha256:digest`.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Layanan Sinkronisasi Waktu Amazon untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan EC2 Pengguna Amazon untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Respons metadata tugas contoh

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
```

```
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
```

```
}
```

Metadata tugas Amazon ECS tersedia untuk tugas di Fargate

Amazon ECS di Fargate menyediakan metode untuk mengambil berbagai metadata, metrik jaringan, dan statistik Docker tentang container Anda dan tugas yang menjadi bagiannya. Ini disebut sebagai titik akhir metadata tugas. Versi titik akhir metadata tugas berikut tersedia untuk Amazon ECS pada tugas Fargate:

- Titik akhir metadata tugas versi 4 — Tersedia untuk tugas yang menggunakan platform versi 1.4.0 atau yang lebih baru.
- Titik akhir metadata tugas versi 3 — Tersedia untuk tugas yang menggunakan platform versi 1.1.0 atau yang lebih baru.

Semua kontainer milik tugas yang diluncurkan dengan mode `awsvpc` jaringan menerima IPv4 alamat lokal dalam rentang alamat link-lokal yang telah ditentukan. Ketika sebuah kontainer melakukan kueri terhadap titik akhir metadata, agen kontainer dapat menentukan tugas yang menjadi milik kontainer berdasarkan alamat IP unik, serta metadata dan statistik untuk tugas tersebut dikembalikan.

Topik

- Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate
- Titik akhir metadata tugas Amazon ECS versi 3 untuk tugas di Fargate

Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate

> **Important**
>
> Jika Anda menggunakan tugas Amazon ECS yang dihosting di EC2 instans Amazon, lihat titik akhir metadata tugas Amazon ECS.

Dimulai dengan versi platform Fargate`1.4.0`, variabel lingkungan bernama `ECS_CONTAINER_METADATA_URI_V4` disuntikkan ke setiap wadah dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 4, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas.

Titik akhir metadata tugas versi 4 berfungsi seperti titik akhir versi 3 tetapi menyediakan metadata jaringan tambahan untuk kontainer dan tugas Anda. Metrik jaringan tambahan juga tersedia saat melakukan kueri terhadap titik akhir `/stats`.

Titik akhir metadata tugas aktif secara default untuk semua tugas Amazon ECS yang dijalankan pada versi platform yang menggunakan versi platform atau AWS Fargate yang lebih baru. `1.4.0`

> **Note**
>
> Untuk menghindari kebutuhan membuat versi titik akhir metadata tugas di masa mendatang, metadata tambahan dapat ditambahkan ke output versi 4. Kami tidak akan menghapus metadata yang ada atau mengubah nama bidang metadata.

Jalur titik akhir metadata tugas Fargate versi 4

Titik akhir metadata tugas berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI_V4}`

Jalur ini mengembalikan metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI_V4}/task`

Jalur ini mengembalikan metadata untuk tugas, termasuk daftar wadah IDs dan nama untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v4 respons JSON untuk tugas di Fargate.

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

Jalur ini mengembalikan statistik Docker untuk kontainer Docker. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

> **Note**
>
> Tugas Amazon ECS AWS Fargate mengharuskan penampung berjalan selama ~1 detik sebelum mengembalikan statistik penampung.

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

Jalur ini mengembalikan statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

> **Note**
>
> Tugas Amazon ECS AWS Fargate mengharuskan penampung berjalan selama ~1 detik sebelum mengembalikan statistik penampung.

Metadata tugas Amazon ECS v4 respons JSON untuk tugas di Fargate

Metadata berikut dikembalikan dalam respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI_V4}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`VPCID`

ID VPC dari instans EC2 penampung Amazon. Bidang ini hanya muncul untuk EC2 instance Amazon.

> **Note**
>
> `VPCID`Metadata hanya disertakan saat menggunakan versi `1.63.1` agen penampung Amazon ECS atau yang lebih baru.

`TaskARN`

Nama Sumber Daya Amazon (ARN) dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas seperti CPU (dinyatakan dalam vCPUs) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 (Windows).

`LaunchType`

Jenis peluncuran yang digunakan tugas. Saat menggunakan penyedia kapasitas cluster, ini menunjukkan apakah tugas tersebut menggunakan Fargate atau EC2 infrastruktur.

> **Note**
>
> `LaunchType`Metadata ini hanya disertakan saat menggunakan Amazon ECS Linux versi agen container atau yang lebih baru (Linux) `1.45.0` atau 1.0.0 atau yang lebih baru (Windows).

`EphemeralStorageMetrics`

Ukuran cadangan dan penggunaan penyimpanan sementara saat ini dari tugas ini.

> **Note**
>
> Fargate mencadangkan ruang pada disk. Ruang ini hanya digunakan oleh Fargate. Anda tidak akan dikenai biaya untuk ruang tersebut. Ruang ini tidak akan ditampilkan dalam metrik ini. Namun demikian, Anda dapat melihat penyimpanan tambahan ini di alat lain seperti `df`.

`Utilized`

Penggunaan penyimpanan sementara saat ini (dalam MiB) dari tugas ini.

`Reserved`

Penyimpanan sementara yang dicadangkan (dalam MiB) dari tugas ini. Ukuran penyimpanan sementara tidak dapat diubah dalam tugas yang sedang berjalan. Anda dapat menentukan `ephermalStorage` objek dalam definisi tugas Anda untuk mengubah jumlah penyimpanan sementara. `ephermalStorage`Ini ditentukan dalam GiB, bukan MiB. The `ephermalStorage` and the hanya `EphemeralStorageMetrics` tersedia di platform Fargate Linux versi 1.4.0 atau yang lebih baru.

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

Saat Anda menggunakan Fargate, id adalah hex 32 digit diikuti dengan angka 10 digit.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 dari manifes gambar. Ini adalah intisari yang dapat digunakan untuk menarik gambar menggunakan format`repository-url/image@sha256:digest`.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`LogDriver`

Driver log yang digunakan wadah.

> **Note**
>
> `LogDriver`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`LogOptions`

Opsi driver log yang ditentukan untuk wadah.

> **Note**
>
> `LogOptions`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`ContainerARN`

Nama Sumber Daya Amazon (ARN) dari wadah.

> **Note**
>
> `ContainerARN`Metadata ini hanya disertakan saat menggunakan versi `1.45.0` agen penampung Amazon ECS Linux atau yang lebih baru.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`Snapshotter`

snapshotterYang digunakan oleh containerd untuk mengunduh gambar kontainer ini. Nilai yang valid adalah`overlayfs`, yang merupakan default, dan`soci`, digunakan saat lazy loading dengan indeks SOCI. Parameter ini hanya tersedia untuk tugas-tugas yang berjalan pada versi platform Linux`1.4.0`.

`RestartCount`

Berapa kali wadah telah dimulai ulang.

> **Note**
>
> `RestartCount`Metadata hanya disertakan jika kebijakan restart diaktifkan untuk penampung. Untuk informasi selengkapnya, lihat Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Kemampuan ini menggunakan Layanan Sinkronisasi Waktu Amazon untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan EC2 Pengguna Amazon untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v4 untuk tugas di Fargate

Contoh berikut menunjukkan contoh keluaran dari titik akhir metadata tugas untuk tugas Amazon ECS yang dijalankan. AWS Fargate

Dari wadah, Anda dapat menggunakan curl diikuti oleh titik akhir data meta tugas untuk menanyakan titik akhir misalnya. `curl ${ECS_CONTAINER_METADATA_URI_V4}/task`

Respons metadata kontainer contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}`, hanya metadata kontainer tersebut yang dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
    "DockerId": "cd189a933e5849daa93386466019ab50-2495160603",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:25f3695bedfb454a50f12d127839a68ad3caf91e451c1da073db34c542c4d2cb",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/default/
cd189a933e5849daa93386466019ab50",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "2"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-08T20:09:11.44527186Z",
    "StartedAt": "2020-10-08T20:09:11.44527186Z",
    "Type": "NORMAL",
```

```
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "192.0.2.3"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0a:de:f6:10:51:e5",
            "IPv4SubnetCIDRBlock": "192.0.2.0/24",
            "DomainNameServers": [
                "192.0.2.2"
            ],
            "DomainNameSearchList": [
                "us-west-2.compute.internal"
            ],
            "PrivateDNSName": "ip-10-0-0-222.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "192.0.2.0/24"
        }
    ],
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/05966557-
f16c-49cb-9352-24b3a0dcd0e1",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/containerlogs",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/cd189a933e5849daa93386466019ab50"
    },
    "LogDriver": "awslogs",
    "Snapshotter": "overlayfs"
}
```

Contoh metadata tugas Amazon ECS v4 untuk tugas di Fargate

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task`, metadata dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
  "Cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/MyEmptyCluster",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyEmptyCluster/
bfa2636268144d039771334145e490c5",
  "Family": "sample-fargate",
  "Revision": "5",
```

```
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": {
    "CPU": 0.25,
    "Memory": 512
  },
  "PullStartedAt": "2023-07-21T15:45:33.532811081Z",
  "PullStoppedAt": "2023-07-21T15:45:38.541068435Z",
  "AvailabilityZone": "us-east-1d",
  "Containers": [
    {
      "DockerId": "bfa2636268144d039771334145e490c5-1117626119",
      "Name": "curl-image",
      "DockerName": "curl-image",
      "Image": "curlimages/curl",
      "ImageID":
 "sha256:daf3f46a2639c1613b25e85c9ee4193af8a1d538f92483d67f9a3d7f21721827",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "curl-image",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 128 },
      "CreatedAt": "2023-07-21T15:45:44.91368314Z",
      "StartedAt": "2023-07-21T15:45:44.91368314Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
```

```
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/da6cccf7-1178-400c-afdf-7536173ee209",
      "Snapshotter": "overlayfs"
    },
    {
      "DockerId": "bfa2636268144d039771334145e490c5-3681984407",
      "Name": "fargate-app",
      "DockerName": "fargate-app",
      "Image": "public.ecr.aws/docker/library/httpd:latest",
      "ImageID":
 "sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5ececa02",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "fargate-app",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 2 },
      "CreatedAt": "2023-07-21T15:45:44.954460255Z",
      "StartedAt": "2023-07-21T15:45:44.954460255Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/f65b461d-aa09-4acb-a579-9785c0530cbc",
      "Snapshotter": "overlayfs"
    }
```

```
  ],
  "LaunchType": "FARGATE",
  "ClockDrift": {
    "ClockErrorBound": 0.446931,
    "ReferenceTimestamp": "2023-07-21T16:09:17Z",
    "ClockSynchronizationStatus": "SYNCHRONIZED"
  },
  "EphemeralStorageMetrics": {
    "Utilized": 261,
    "Reserved": 20496
  }
}
```

Respons statistik tugas contoh

Ketika melakukan kueri terhadap titik akhir `${ECS_CONTAINER_METADATA_URI_V4}/task/stats`, metrik jaringan dari tugas yang berisi bagian kontainer dikembalikan kepada Anda. Berikut ini adalah output contoh.

```
{
  "3d1f891cded94dc795608466cce8ddcf-464223573": {
    "read": "2020-10-08T21:24:44.938937019Z",
    "preread": "2020-10-08T21:24:34.938633969Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 638976
```

```
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 638976
        }
      ],
      "io_serviced_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 12
```

```
        }
      ],
      "io_queue_recursive": [],
      "io_service_time_recursive": [],
      "io_wait_time_recursive": [],
      "io_merged_recursive": [],
      "io_time_recursive": [],
      "sectors_recursive": []
    },
    "num_procs": 0,
    "storage_stats": {},
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 1137691504,
        "percpu_usage": [
          696479228,
          441212276,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9393210000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "precpu_stats": {
      "cpu_usage": {
```

```
        "total_usage": 1136624601,
        "percpu_usage": [
          695639662,
          440984939,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9373330000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "memory_stats": {
      "usage": 6504448,
      "max_usage": 8458240,
      "stats": {
        "active_anon": 1675264,
        "active_file": 557056,
        "cache": 651264,
        "dirty": 0,
        "hierarchical_memory_limit": 536870912,
        "hierarchical_memsw_limit": 9223372036854772000,
        "inactive_anon": 0,
        "inactive_file": 3088384,
        "mapped_file": 430080,
        "pgfault": 11034,
        "pgmajfault": 5,
```

```
        "pgpgin": 8436,
        "pgpgout": 7137,
        "rss": 4669440,
        "rss_huge": 0,
        "total_active_anon": 1675264,
        "total_active_file": 557056,
        "total_cache": 651264,
        "total_dirty": 0,
        "total_inactive_anon": 0,
        "total_inactive_file": 3088384,
        "total_mapped_file": 430080,
        "total_pgfault": 11034,
        "total_pgmajfault": 5,
        "total_pgpgin": 8436,
        "total_pgpgout": 7137,
        "total_rss": 4669440,
        "total_rss_huge": 0,
        "total_unevictable": 0,
        "total_writeback": 0,
        "unevictable": 0,
        "writeback": 0
      },
      "limit": 9223372036854772000
    },
    "name": "curltest",
    "id": "3d1f891cded94dc795608466cce8ddcf-464223573",
    "networks": {
      "eth1": {
        "rx_bytes": 2398415937,
        "rx_packets": 1898631,
        "rx_errors": 0,
        "rx_dropped": 0,
        "tx_bytes": 1259037719,
        "tx_packets": 428002,
        "tx_errors": 0,
        "tx_dropped": 0
      }
    },
    "network_rate_stats": {
      "rx_bytes_per_sec": 43.298687872232854,
      "tx_bytes_per_sec": 215.39347269466413
    }
  }
```

```
}
```

Titik akhir metadata tugas Amazon ECS versi 3 untuk tugas di Fargate

> **Important**
>
> Titik akhir metadata tugas versi 3 tidak lagi dipertahankan secara aktif. Kami menyarankan Anda memperbarui titik akhir metadata tugas versi 4 untuk mendapatkan informasi titik akhir metadata terbaru. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4 untuk tugas di Fargate”.

Dimulai dengan versi platform Fargate`1.1.0`, variabel lingkungan bernama `ECS_CONTAINER_METADATA_URI` disuntikkan ke setiap wadah dalam tugas. Ketika Anda melakukan kueri terhadap titik akhir metadata tugas versi 3, berbagai metadata tugas dan Statistik Docker tersedia untuk tugas.

Fitur titik akhir metadata tugas diaktifkan secara default untuk tugas Amazon ECS yang dihosting di Fargate yang menggunakan versi platform atau yang lebih baru. `1.1.0` Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.

Jalur titik akhir metadata tugas untuk tugas di Fargate

Titik akhir API berikut tersedia untuk kontainer:

`${ECS_CONTAINER_METADATA_URI}`

Jalur ini mengembalikan JSON metadata untuk kontainer.

`${ECS_CONTAINER_METADATA_URI}/task`

Jalur ini mengembalikan metadata JSON untuk tugas, termasuk daftar wadah IDs dan nama untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang respons titik akhir ini, lihat Metadata tugas Amazon ECS v3 Respons JSON untuk tugas di Fargate.

`${ECS_CONTAINER_METADATA_URI}/stats`

Jalur ini mengembalikan JSON statistik Docker untuk kontainer Docker tertentu. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

`${ECS_CONTAINER_METADATA_URI}/task/stats`

Jalur ini mengembalikan JSON statistik Docker untuk semua kontainer yang terkait dengan tugas. Untuk informasi selengkapnya tentang masing-masing statistik yang dikembalikan, lihat ContainerStatsdi dokumentasi Docker API.

Metadata tugas Amazon ECS v3 Respons JSON untuk tugas di Fargate

Informasi berikut dikembalikan dari respons JSON titik akhir metadata tugas (`${ECS_CONTAINER_METADATA_URI}/task`).

`Cluster`

Nama Sumber Daya Amazon (ARN) atau nama pendek dari cluster Amazon ECS tempat tugas tersebut berada.

`TaskARN`

Nama Sumber Daya Amazon (ARN) dari tugas yang menjadi milik wadah tersebut.

`Family`

Keluarga definisi tugas Amazon ECS untuk tugas tersebut.

`Revision`

Revisi definisi tugas Amazon ECS untuk tugas tersebut.

`DesiredStatus`

Status yang diinginkan untuk tugas dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk tugas dari Amazon ECS.

`Limits`

Batas sumber daya yang ditentukan pada tingkat tugas, seperti CPU (dinyatakan dalam vCPUs) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`PullStartedAt`

Stempel waktu saat penarikan gambar kontainer pertama dimulai.

`PullStoppedAt`

Stempel waktu ketika penarikan gambar kontainer terakhir selesai.

`AvailabilityZone`

Availability Zone tempat tugas berada.

> **Note**
>
> Metadata Availability Zone hanya tersedia untuk tugas Fargate menggunakan platform versi 1.4 atau yang lebih baru (Linux) atau 1.0.0 atau yang lebih baru (Windows).

`Containers`

Daftar metadata kontainer untuk setiap kontainer yang terkait dengan tugas.

`DockerId`

ID Docker untuk wadah.

`Name`

Nama wadah seperti yang ditentukan dalam definisi tugas.

`DockerName`

Nama wadah yang dipasok ke Docker. Agen penampung Amazon ECS menghasilkan nama unik untuk penampung untuk menghindari tabrakan nama saat beberapa salinan dari definisi tugas yang sama dijalankan pada satu instance.

`Image`

Gambar untuk wadah.

`ImageID`

Intisari SHA-256 dari manifes gambar. Ini adalah intisari yang dapat digunakan untuk menarik gambar menggunakan format`repository-url/image@sha256:digest`.

`Ports`

Port apa pun yang terbuka untuk wadah. Parameter ini dihilangkan jika tidak ada port yang terbuka.

`Labels`

Label apa pun diterapkan pada wadah. Parameter ini dihilangkan jika tidak ada label yang diterapkan.

`DesiredStatus`

Status yang diinginkan untuk wadah dari Amazon ECS.

`KnownStatus`

Status yang diketahui untuk wadah dari Amazon ECS.

`ExitCode`

Kode keluar untuk wadah. Parameter ini dihilangkan jika wadah belum keluar.

`Limits`

Batas sumber daya yang ditentukan pada tingkat kontainer, seperti CPU (dinyatakan dalam unit CPU) dan memori. Parameter ini dihilangkan jika tidak ada batasan sumber daya yang ditentukan.

`CreatedAt`

Cap waktu untuk saat wadah dibuat. Parameter ini dihilangkan jika wadah belum dibuat.

`StartedAt`

Cap waktu untuk saat wadah dimulai. Parameter ini dihilangkan jika wadah belum dimulai.

`FinishedAt`

Cap waktu untuk saat wadah berhenti. Parameter ini dihilangkan jika wadah belum berhenti.

`Type`

Jenis wadahnya. Wadah yang ditentukan dalam definisi tugas Anda adalah tipe`NORMAL`. Anda dapat mengabaikan jenis penampung lain, yang digunakan untuk penyediaan sumber daya tugas internal oleh agen penampung Amazon ECS.

`Networks`

Informasi jaringan untuk wadah, seperti mode jaringan dan alamat IP. Parameter ini dihilangkan jika tidak ada informasi jaringan yang didefinisikan.

`ClockDrift`

Informasi tentang perbedaan antara waktu referensi dan waktu sistem. Ini berlaku untuk sistem operasi Linux. Kemampuan ini menggunakan Layanan Sinkronisasi Waktu Amazon untuk mengukur akurasi jam dan memberikan kesalahan jam yang terikat pada kontainer. Untuk informasi selengkapnya, lihat Mengatur waktu untuk instans Linux Anda di Panduan EC2 Pengguna Amazon untuk instans Linux.

`ReferenceTime`

Dasar akurasi jam. Amazon ECS menggunakan standar global Coordinated Universal Time (UTC) melalui NTP, misalnya. `2021-09-07T16:57:44Z`

`ClockErrorBound`

Ukuran kesalahan jam, didefinisikan sebagai offset ke UTC. Kesalahan ini adalah perbedaan milidetik antara waktu referensi dan waktu sistem.

`ClockSynchronizationStatus`

Menunjukkan apakah upaya sinkronisasi terbaru antara waktu sistem dan waktu referensi berhasil.

Nilai yang valid adalah `SYNCHRONIZED` dan `NOT_SYNCHRONIZED`.

`ExecutionStoppedAt`

Cap waktu untuk saat tugas `DesiredStatus` pindah ke`STOPPED`. Ini terjadi ketika wadah penting bergerak ke`STOPPED`.

Contoh metadata tugas Amazon ECS v3 untuk tugas di Fargate

Respons JSON berikut adalah untuk tugas kontainer tunggal.

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
```

```
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
```

```
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Introspeksi wadah Amazon ECS

Agen penampung Amazon ECS menyediakan operasi API untuk mengumpulkan detail tentang instance kontainer tempat agen dijalankan dan tugas terkait yang berjalan pada instance tersebut. Anda dapat menggunakan curl perintah dari dalam instance container untuk menanyakan agen kontainer Amazon ECS (port 51678) dan mengembalikan metadata instans kontainer atau informasi tugas.

> **Important**
>
> Instance container Anda harus memiliki peran IAM yang memungkinkan akses ke Amazon ECS untuk mengambil metadata. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.

Untuk melihat metadata instans kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut. Metadata mencakup ID instans penampung, klaster Amazon ECS tempat instance container terdaftar, dan informasi versi agen penampung Amazon ECS.

```
curl -s http://localhost:51678/v1/metadata | python3 -mjson.tool
```

Output:

```
{
    "Cluster": "cluster_name",
    "ContainerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/cluster_name/container_instance_id",
    "Version": "Amazon ECS Agent - v1.30.0 (02ff320c)"
}
```

Untuk melihat informasi tentang semua tugas yang berjalan pada instans kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut:

```
curl http://localhost:51678/v1/tasks
```

Output:

```
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/example5-58ff-46c9-
ae05-543f8example",
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Family": "hello_world",
            "Version": "8",
            "Containers": [
                {
          "DockerId":
 "9581a69a761a557fbfce1d0f6745e4af5b9dbfb86b6b2c5c4df156f1a5932ff1",
          "DockerName": "ecs-hello_world-8-mysql-fcae8ac8f9f1d89d8301",
          "Name": "mysql",
          "CreatedAt": "2023-10-08T20:09:11.44527186Z",
          "StartedAt": "2023-10-08T20:09:11.44527186Z",
          "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        },
        {
          "DockerId":
 "bf25c5c5b2d4dba68846c7236e75b6915e1e778d31611e3c6a06831e39814a15",
          "DockerName": "ecs-hello_world-8-wordpress-e8bfddf9b488dff36c00",
          "Name": "wordpress"
        }
    ]
```

```
}
    ]
}
```

Anda dapat melihat informasi untuk tugas tertentu yang berjalan pada instans kontainer. Untuk menentukan tugas atau kontainer tertentu, tambahkan salah satu dari berikut pada permintaan:

- ARN tugas (`?taskarn=`*`task_arn`*)
- ID Docker untuk kontainer (`?dockerid=`*`docker_id`*)

Untuk mendapatkan informasi tugas dengan Docker ID kontainer, masuk ke instans kontainer Anda melalui SSH dan jalankan perintah berikut.

> **Note**
>
> Agen penampung Amazon ECS sebelum versi 1.14.2 memerlukan wadah Docker lengkap IDs untuk API introspeksi, bukan versi pendek yang ditampilkan. docker ps Anda bisa mendapatkan ID Docker lengkap untuk kontainer dengan menjalankan perintah docker ps --no-trunc pada instans kontainer.

```
curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

Output:

```
{
    "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId":
 "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx",
            "CreatedAt": "2023-10-08T20:09:11.44527186Z",
            "StartedAt": "2023-10-08T20:09:11.44527186Z",
            "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
```

```
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

Identifikasi perilaku yang tidak sah menggunakan Runtime Monitoring

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Runtime Monitoring in GuardDuty melindungi beban kerja yang berjalan di Fargate dan instance EC2 container dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensil yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDutyRuntime Monitoring di Panduan GuardDuty Pengguna.

Administrator keamanan Anda mengaktifkan Runtime Monitoring untuk satu atau beberapa akun di AWS Organizations for GuardDuty. Mereka juga memilih apakah GuardDuty secara otomatis menyebarkan agen GuardDuty keamanan saat Anda menggunakan Fargate. Semua cluster Anda secara otomatis dilindungi, dan GuardDuty mengelola agen keamanan atas nama Anda.

Anda juga dapat mengonfigurasi agen GuardDuty keamanan secara manual dalam kasus berikut:

- Anda menggunakan instance EC2 kontainer
- Anda memerlukan kontrol granular untuk mengaktifkan Runtime Monitoring di tingkat cluster

Untuk menggunakan Runtime Monitoring, Anda harus mengonfigurasi cluster yang dilindungi, dan menginstal serta mengelola agen GuardDuty keamanan pada instance EC2 container Anda.

Bagaimana Runtime Monitoring bekerja dengan Amazon ECS

Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang memantau aktivitas beban kerja Amazon ECS untuk bagaimana aplikasi meminta, mendapatkan akses, dan mengkonsumsi sumber daya sistem yang mendasarinya.

Untuk tugas Fargate, agen GuardDuty keamanan berjalan sebagai wadah sespan untuk setiap tugas.

Untuk instance EC2 kontainer, agen GuardDuty keamanan berjalan sebagai proses pada instance.

Agen GuardDuty keamanan mengumpulkan data dari sumber daya berikut, dan kemudian mengirimkan data GuardDuty untuk diproses. Anda dapat melihat temuan di GuardDuty konsol. Anda juga dapat mengirimnya ke pihak lain Layanan AWS seperti AWS Security Hub, atau vendor keamanan pihak ketiga untuk agregasi dan remediasi. Untuk informasi tentang cara melihat dan mengelola temuan, lihat Mengelola GuardDuty temuan Amazon di Panduan GuardDuty Pengguna Amazon.

- Tanggapan dari panggilan Amazon ECS API berikut:
  - DescribeClusters

    Parameter respons menyertakan tag Runtime Monitoring (saat tag disetel) saat Anda menggunakan `--include TAGS` opsi.
  - DescribeTasks

    Untuk tipe peluncuran Fargate, parameter respons termasuk wadah GuardDuty sespan.
  - ListAccountSettings

    Parameter respons mencakup pengaturan akun Runtime Monitoring, yang ditetapkan oleh administrator keamanan Anda.
- Data introspeksi agen kontainer. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.
- Titik akhir metadata tugas untuk jenis peluncuran:
  - Titik akhir metadata tugas Amazon ECS versi 4
  - Titik akhir metadata tugas Amazon ECS versi 4 untuk tugas di Fargate

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Runtime Monitoring:

- Runtime Monitoring memiliki biaya yang terkait dengannya. Untuk informasi selengkapnya, lihat GuardDuty Harga Amazon.
- Runtime Monitoring tidak didukung di Amazon ECS Anywhere.
- Runtime Monitoring tidak didukung untuk sistem operasi Windows.
- Saat Anda menggunakan Amazon ECS Exec di Fargate, Anda harus menentukan nama penampung karena agen GuardDuty keamanan berjalan sebagai wadah sespan.
- Anda tidak dapat menggunakan Amazon ECS Exec pada wadah GuardDuty sespan agen keamanan.
- Pengguna IAM yang mengontrol Runtime Monitoring di tingkat cluster, harus memiliki izin IAM yang sesuai untuk penandaan. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.
- Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan, yang disimpan dalam repositori pribadi Amazon ECR, atas nama Anda.

Pemanfaatan Sumber Daya

Tag yang Anda tambahkan ke klaster dihitung terhadap kuota tag cluster.

Kontainer sespan GuardDuty agen tidak dihitung terhadap kontainer per kuota definisi tugas.

Seperti kebanyakan perangkat lunak keamanan, ada sedikit biaya overhead untuk GuardDuty. Untuk informasi tentang batas memori Fargate, lihat batas CPU dan memori di GuardDuty Panduan Pengguna. Untuk informasi tentang batas EC2 memori Amazon, lihat CPU dan batas memori untuk GuardDuty agen.

Pemantauan Runtime untuk beban kerja Amazon ECS Fargate

Jika Anda menggunakan instance EC2 container, Anda harus mengkonfigurasi Runtime Monitoring secara manual. Untuk informasi selengkapnya, lihat Runtime Monitoring untuk EC2 beban kerja di Amazon ECS.

Anda dapat GuardDuty mengelola agen keamanan pada instance kontainer Anda. Opsi ini hanya tersedia untuk Fargate. Opsi ini (manajemen GuardDuty agen) tersedia di GuardDuty

Saat Anda menggunakan manajemen GuardDuty agen GuardDuty , lakukan operasi berikut:

- Membuat titik akhir VPC untuk GuardDuty setiap VPC yang menghosting cluster.

- Mengambil, dan menginstal agen GuardDuty keamanan terbaru sebagai wadah sespan pada semua tugas Fargate mandiri baru, dan penerapan layanan baru.

  Penyebaran layanan baru terjadi saat pertama kali Anda meluncurkan layanan, atau saat Anda memperbarui layanan yang ada dengan opsi force new deployment.

Mengaktifkan Runtime Monitoring untuk Amazon ECS

Anda dapat mengonfigurasi GuardDuty untuk mengelola agen keamanan secara otomatis untuk semua cluster Fargate Anda.

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring:

- Versi platform Fargate harus `1.4.0` atau lebih baru untuk Linux.
- Peran dan izin IAM untuk Amazon ECS:
  - Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan atas nama Anda. Untuk mengetahui informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
  - Anda mengontrol Runtime Monitoring untuk klaster dengan tag yang telah ditentukan sebelumnya. Jika kebijakan akses Anda membatasi akses berdasarkan tag, Anda harus memberikan izin eksplisit kepada pengguna IAM Anda untuk menandai klaster. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.
- Menghubungkan ke repositori Amazon ECR:

  Agen GuardDuty keamanan disimpan dalam repositori Amazon ECR. Setiap tugas mandiri dan layanan harus memiliki akses ke repositori. Anda dapat menggunakan salah satu opsi berikut:
  - Untuk tugas di subnet publik, Anda dapat menggunakan alamat IP publik untuk tugas tersebut, atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - Untuk tugas di subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT), atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan.

    Untuk informasi selengkapnya, lihat Subnet pribadi dan gateway NAT.

- Anda harus memiliki `AWSServiceRoleForAmazonGuardDuty` peran untuk GuardDuty. Untuk informasi selengkapnya, lihat Izin peran terkait layanan GuardDuty di Panduan Pengguna Amazon GuardDuty.
- File apa pun yang ingin Anda lindungi dengan Runtime Monitoring harus dapat diakses oleh pengguna root. Jika Anda mengubah izin file secara manual, Anda harus mengaturnya ke`755`.

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring pada instance container: EC2

- Anda harus menggunakan versi `20230929` atau yang lebih baru dari Amazon ECS-AMI.
- Anda harus menjalankan agen Amazon ECS ke versi `1.77` atau yang lebih baru pada instance container.
- Anda harus menggunakan versi kernel `5.10` atau yang lebih baru.
- Untuk informasi tentang sistem operasi dan arsitektur Linux yang didukung, lihat Model operasi dan beban kerja mana yang didukung GuardDuty Runtime Monitoring.
- Anda dapat menggunakan Systems Manager untuk mengelola instance container Anda. Untuk informasi selengkapnya, lihat Menyiapkan Systems Manager untuk EC2 instance di Panduan AWS Systems Manager Session Manager Pengguna.

Prosedur

Anda mengaktifkan Runtime Monitoring di GuardDuty. Untuk informasi tentang cara mengaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Menambahkan Runtime Monitoring ke tugas Amazon ECS Fargate yang ada

Saat Anda mengaktifkan Runtime Monitoring, semua tugas mandiri baru, dan penerapan layanan baru di cluster akan terlindungi secara otomatis. Untuk mempertahankan kendala kekekalan, tugas yang ada tidak terpengaruh.

Prasyarat

1. Aktifkan Runtime Monitoring. Untuk informasi selengkapnya, lihat Mengaktifkan Runtime Monitoring untuk Amazon ECS.
2. Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan atas nama Anda. Untuk mengetahui informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Prosedur

- Untuk segera melindungi tugas, Anda perlu melakukan salah satu tindakan berikut:
  - Untuk tugas mandiri, hentikan tugas, lalu mulailah. Untuk informasi selengkapnya, lihat Menghentikan tugas Amazon ECS dan Menjalankan aplikasi sebagai tugas Amazon ECS
  - Untuk tugas yang merupakan bagian dari layanan, perbarui layanan dengan opsi “paksa penerapan baru”. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Menghapus Runtime Monitoring dari klaster Amazon ECS

Anda mungkin ingin mengecualikan klaster tertentu dari perlindungan, misalnya cluster yang Anda gunakan untuk pengujian. Hal ini menyebabkan GuardDuty untuk melakukan operasi berikut pada sumber daya di cluster:

- Tidak lagi menyebarkan agen GuardDuty keamanan ke tugas Fargate mandiri baru, atau penerapan layanan baru.

  Untuk mempertahankan kendala kekekalan, tugas dan penerapan yang ada dengan Runtime Monitoring diaktifkan tidak terpengaruh.
- Hentikan penagihan dan tidak lagi menerima acara run time untuk tugas.

Prosedur

Lakukan langkah-langkah berikut untuk menghapus Runtime Monitoring dari cluster.

1. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `false` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API. Gunakan nilai berikut untuk tag.

   > **Note**
   >
   > Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

   Kunci =`GuardDutyManaged`, Nilai = `false`

2. Hapus titik akhir GuardDuty VPC untuk cluster. Untuk informasi selengkapnya tentang cara menghapus titik akhir VPC, lihat Menghapus titik akhir antarmuka di Panduan Pengguna.AWS PrivateLink

Menghapus Runtime Monitoring untuk Amazon ECS dari akun

Saat Anda tidak lagi ingin menggunakan Runtime Monitoring, nonaktifkan fitur di GuardDuty. Untuk informasi tentang cara menonaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

GuardDuty melakukan operasi berikut:

- Menghapus titik akhir VPC GuardDuty untuk setiap VPC yang menghosting cluster.
- Tidak lagi menyebarkan agen GuardDuty keamanan ke tugas Fargate mandiri baru, atau penerapan layanan baru.

  Untuk mempertahankan kendala kekekalan, tugas dan penerapan yang ada tidak terpengaruh sampai dihentikan, direplikasi, atau diskalakan.
- Menghentikan penagihan dan tidak lagi menerima acara run time untuk tugas.

Runtime Monitoring untuk EC2 beban kerja di Amazon ECS

Gunakan opsi ini saat Anda menggunakan EC2 instance untuk kapasitas Anda, atau saat Anda memerlukan kontrol granular Runtime Monitoring di tingkat cluster di Fargate.

Anda menyediakan kluster untuk Runtime Monitoring dengan menambahkan tag yang telah ditentukan sebelumnya.

Untuk instance EC2 kontainer, Anda mengunduh, menginstal, dan mengelola agen GuardDuty keamanan.

Untuk Fargate, GuardDuty mengelola agen keamanan atas nama Anda.

Mengaktifkan Runtime Monitoring untuk Amazon ECS

Anda dapat mengaktifkan Runtime Monitoring untuk cluster dengan EC2 instance, atau saat Anda memerlukan kontrol granular Runtime Monitoring di tingkat cluster di Fargate.

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring:

- Versi platform Fargate harus `1.4.0` atau lebih baru untuk Linux.
- Peran dan izin IAM untuk Amazon ECS:
  - Tugas Fargate harus menggunakan peran eksekusi tugas. Peran ini memberikan izin tugas untuk mengambil, memperbarui, dan mengelola agen GuardDuty keamanan atas nama Anda. Untuk mengetahui informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.
  - Anda mengontrol Runtime Monitoring untuk klaster dengan tag yang telah ditentukan sebelumnya. Jika kebijakan akses Anda membatasi akses berdasarkan tag, Anda harus memberikan izin eksplisit kepada pengguna IAM Anda untuk menandai klaster. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.
- Menghubungkan ke repositori Amazon ECR:

  Agen GuardDuty keamanan disimpan dalam repositori Amazon ECR. Setiap tugas mandiri dan layanan harus memiliki akses ke repositori. Anda dapat menggunakan salah satu opsi berikut:
  - Untuk tugas di subnet publik, Anda dapat menggunakan alamat IP publik untuk tugas tersebut, atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan. Untuk informasi selengkapnya, lihat Antarmuka Amazon ECR VPC endpoint AWS PrivateLink() di Panduan Pengguna Amazon Elastic Container Registry.
  - Untuk tugas di subnet pribadi, Anda dapat menggunakan gateway Network Address Translation (NAT), atau membuat titik akhir VPC untuk Amazon ECR di subnet tempat tugas berjalan.

    Untuk informasi selengkapnya, lihat Subnet pribadi dan gateway NAT.
- Anda harus memiliki `AWSServiceRoleForAmazonGuardDuty` peran untuk GuardDuty. Untuk informasi selengkapnya, lihat Izin peran terkait layanan GuardDuty di Panduan Pengguna Amazon GuardDuty.
- File apa pun yang ingin Anda lindungi dengan Runtime Monitoring harus dapat diakses oleh pengguna root. Jika Anda mengubah izin file secara manual, Anda harus mengaturnya ke`755`.

Berikut ini adalah prasyarat untuk menggunakan Runtime Monitoring pada instance container: EC2

- Anda harus menggunakan versi `20230929` atau yang lebih baru dari Amazon ECS-AMI.
- Anda harus menjalankan agen Amazon ECS ke versi `1.77` atau yang lebih baru pada instance container.
- Anda harus menggunakan versi kernel `5.10` atau yang lebih baru.

- Untuk informasi tentang sistem operasi dan arsitektur Linux yang didukung, lihat Model operasi dan beban kerja mana yang didukung GuardDuty Runtime Monitoring.
- Anda dapat menggunakan Systems Manager untuk mengelola instance container Anda. Untuk informasi selengkapnya, lihat Menyiapkan Systems Manager untuk EC2 instance di Panduan AWS Systems Manager Session Manager Pengguna.

Anda mengaktifkan Runtime Monitoring di GuardDuty. Untuk informasi tentang cara mengaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Menambahkan Runtime Monitoring pada klaster Amazon ECS

Konfigurasikan Runtime Monitoring untuk cluster, lalu instal agen GuardDuty keamanan pada instance EC2 container Anda.

Prasyarat

1. Aktifkan Runtime Monitoring. Untuk informasi selengkapnya, lihat Mengaktifkan Runtime Monitoring untuk Amazon ECS.
2. Anda mengontrol Runtime Monitoring untuk klaster dengan tag yang telah ditentukan sebelumnya. Jika kebijakan akses Anda membatasi akses berdasarkan tag, Anda harus memberikan izin eksplisit kepada pengguna IAM Anda untuk menandai klaster. Untuk informasi selengkapnya, lihat tutorial IAM: Menentukan izin untuk mengakses AWS sumber daya berdasarkan tag di Panduan Pengguna IAM.

Prosedur

Lakukan operasi berikut untuk menambahkan Runtime Monitoring ke cluster.

1. Buat titik akhir VPC GuardDuty untuk setiap VPC cluster. Untuk informasi selengkapnya, lihat Membuat titik akhir Amazon VPC secara manual di Panduan Pengguna. GuardDuty
2. Konfigurasikan instance EC2 kontainer.

    a. Perbarui agen Amazon ECS ke versi `1.77` atau yang lebih baru pada instance EC2 kontainer di cluster. Untuk mengetahui informasi selengkapnya, lihat Memperbarui agen kontainer Amazon ECS.

    b. Instal agen GuardDuty keamanan pada instance EC2 kontainer di cluster. Untuk informasi selengkapnya, lihat Mengelola agen keamanan di EC2 instans Amazon secara manual di Panduan GuardDuty Pengguna.

Semua tugas baru dan yang sudah ada, dan penerapan segera dilindungi karena agen GuardDuty keamanan berjalan sebagai proses pada instance EC2 container.

3. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `true` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API. Gunakan nilai berikut untuk tag.

> **Note**
>
> Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

Kunci =`GuardDutyManaged`, Nilai = `true`

Menambahkan Runtime Monitoring ke tugas Amazon ECS yang ada

Saat Anda mengaktifkan Runtime Monitoring, semua tugas mandiri baru, dan penerapan layanan baru di cluster akan terlindungi secara otomatis. Untuk mempertahankan kendala kekekalan, tugas yang ada tidak terpengaruh.

Prasyarat

- Aktifkan Runtime Monitoring. Untuk informasi selengkapnya, lihat Mengaktifkan Runtime Monitoring untuk Amazon ECS.

Prosedur

- Untuk segera melindungi tugas, Anda perlu melakukan salah satu tindakan berikut:
  - Untuk tugas mandiri, hentikan tugas, lalu mulailah. Untuk informasi selengkapnya, lihat Menghentikan tugas Amazon ECS dan Menjalankan aplikasi sebagai tugas Amazon ECS.
  - Untuk tugas yang merupakan bagian dari layanan, perbarui layanan dengan opsi “paksa penerapan baru”. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS.

Menghapus Runtime Monitoring dari klaster Amazon ECS

Anda dapat menghapus Runtime Monitoring dari sebuah cluster. Hal ini GuardDuty menyebabkan berhenti memantau semua sumber daya di cluster.

Untuk menghapus Runtime Monitoring dari sebuah cluster

1. Gunakan konsol Amazon ECS atau AWS CLI untuk menyetel kunci `GuardDutyManaged` tag pada cluster. `false` Untuk informasi selengkapnya, lihat Memperbarui klaster atau Bekerja dengan tag menggunakan CLI atau API.

   > **Note**
   >
   > Kunci dan Nilai peka huruf besar/kecil dan harus sama persis dengan string.

   Kunci =`GuardDutyManaged`, Nilai = `false`
2. Copot pemasangan agen GuardDuty keamanan pada instance EC2 kontainer Anda di cluster.

   Untuk informasi selengkapnya, lihat Menghapus instalan agen keamanan secara manual di Panduan GuardDuty Pengguna.
3. Hapus titik akhir GuardDuty VPC untuk setiap VPC cluster. Untuk informasi selengkapnya tentang cara menghapus titik akhir VPC, lihat Menghapus titik akhir antarmuka di Panduan Pengguna.AWS PrivateLink

Memperbarui agen GuardDuty keamanan di instans penampung Amazon ECS Anda

Untuk informasi tentang cara memperbarui agen GuardDuty keamanan pada instans EC2 penampung Anda, lihat Memperbarui agen GuardDuty keamanan di Panduan GuardDuty Pengguna Amazon.

Menghapus Runtime Monitoring untuk Amazon ECS dari akun

Saat Anda tidak lagi ingin menggunakan Runtime Monitoring, nonaktifkan fitur di GuardDuty. Untuk informasi tentang cara menonaktifkan fitur, lihat Mengaktifkan Pemantauan Runtime di GuardDuty Panduan Pengguna Amazon.

Hapus Runtime Monitoring dari semua cluster. Untuk informasi selengkapnya, lihat Menghapus Runtime Monitoring dari klaster Amazon ECS.

Pemecahan Masalah Pemantauan Runtime

Anda mungkin perlu memecahkan masalah atau memverifikasi bahwa Runtime Monitoring diaktifkan dan berjalan pada tugas dan container Anda.

Topik

- Bagaimana cara mengetahui apakah Runtime Monitoring aktif di akun saya?
- Bagaimana saya bisa tahu apakah Runtime Monitoring aktif di cluster?
- Bagaimana saya bisa tahu apakah agen GuardDuty keamanan menjalankan tugas Fargate?
- Bagaimana saya bisa tahu apakah agen GuardDuty keamanan berjalan pada instance EC2 container?
- Apa yang terjadi ketika tidak ada peran eksekusi tugas untuk tugas yang berjalan di cluster?
- Bagaimana saya bisa tahu apakah saya memiliki izin yang benar untuk menandai cluster untuk Runtime Monitoring?
- Apa yang terjadi ketika tidak ada koneksi Amazon ECR?
- Bagaimana cara mengatasi kesalahan memori pada tugas Fargate saya setelah mengaktifkan Runtime Monitoring?

Bagaimana cara mengetahui apakah Runtime Monitoring aktif di akun saya?

Di konsol Amazon ECS, informasinya ada di halaman Pengaturan Akun.

Anda juga dapat menjalankan `list-account-settings` dengan `effective-settings` opsi.

```
aws ecs list-account-settings --effective-settings
```

Output

Pengaturan dengan nama diatur ke `guardDutyActivate` dan nilai ditetapkan untuk on menunjukkan bahwa akun dikonfigurasi. Anda harus memeriksa dengan GuardDuty administrator Anda untuk melihat apakah pengelolaannya otomatis atau manual.

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": "aws-managed"
    }
}
```

Bagaimana saya bisa tahu apakah Runtime Monitoring aktif di cluster?

Anda dapat meninjau statistik cakupan di GuardDuty konsol. Ini termasuk informasi untuk sumber daya Amazon ECS yang terkait dengan akun Anda sendiri atau akun anggota Anda adalah persentase cluster sehat di semua cluster yang dipilih. Wilayah AWS Ini termasuk cakupan untuk cluster yang menggunakan Fargate EC2 dan tipe peluncuran. Untuk informasi selengkapnya, lihat Meninjau statistik cakupan di Panduan GuardDuty Pengguna Amazon.

Bagaimana saya bisa tahu apakah agen GuardDuty keamanan menjalankan tugas Fargate?

Agen GuardDuty keamanan berjalan sebagai wadah sespan untuk tugas Fargate.

Di konsol Amazon ECS, sespan ditampilkan di bawah Kontainer di halaman Detail tugas.

Anda dapat menjalankan `describe-tasks` dan mencari wadah dengan nama yang disetel ke `aws-gd-agent` dan LastStatus disetel ke. `RUNNING`

Contoh berikut menunjukkan output untuk cluster default untuk tugas`aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE`.

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

Output

Wadah bernama `gd-agent` ada di `RUNNING` negara bagian.

```
"containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/4df26bb4-
f057-467b-a079-96167EXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/0b69d5c0-
d655-4695-98cd-5d2d5EXAMPLE",
        "lastStatus": "RUNNING",
        "healthStatus": "UNKNOWN",
        "memory": "string",
        "name": "aws-gd-agent"
      }
    ]
```

Bagaimana saya bisa tahu apakah agen GuardDuty keamanan berjalan pada instance EC2 container?

Jalankan perintah berikut untuk melihat status:

```
sudo systemctl status amazon-guardduty-agent
```

File log berada di lokasi berikut:

```
/var/log/amzn-guardduty-agent
```

Apa yang terjadi ketika tidak ada peran eksekusi tugas untuk tugas yang berjalan di cluster?

Untuk tugas Fargate, tugas dimulai tanpa wadah sespan agen GuardDuty keamanan. GuardDuty Dasbor akan menunjukkan bahwa tugas tersebut tidak memiliki perlindungan di dasbor statistik cakupan.

Bagaimana saya bisa tahu apakah saya memiliki izin yang benar untuk menandai cluster untuk Runtime Monitoring?

Untuk menandai cluster, Anda harus memiliki `ecs:TagResource` tindakan untuk keduanya `CreateCluster` dan`UpdateCluster`.

Berikut ini adalah cuplikan dari contoh kebijakan.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "ecs:CreateAction" : "CreateCluster",
             "ecs:CreateAction" : "UpdateCluster",
          }
       }
    }
```

```
  ]
}
```

Apa yang terjadi ketika tidak ada koneksi Amazon ECR?

Untuk tugas Fargate, tugas dimulai tanpa wadah sespan agen GuardDuty keamanan. GuardDuty Dasbor akan menunjukkan bahwa tugas tersebut tidak memiliki perlindungan di dasbor statistik cakupan.

Bagaimana cara mengatasi kesalahan memori pada tugas Fargate saya setelah mengaktifkan Runtime Monitoring?

Agen GuardDuty keamanan adalah proses yang ringan. Namun, prosesnya masih mengkonsumsi sumber daya sesuai dengan ukuran beban kerja. Sebaiknya gunakan alat pelacak sumber daya kontainer, seperti Amazon CloudWatch Container Insights untuk mementaskan GuardDuty penerapan di klaster Anda. Alat-alat ini membantu Anda menemukan profil konsumsi agen GuardDuty keamanan untuk aplikasi Anda. Anda kemudian dapat menyesuaikan ukuran tugas Fargate Anda, jika diperlukan, untuk menghindari potensi keluar dari kondisi memori.

Pantau wadah Amazon ECS dengan ECS Exec

Dengan Amazon ECS Exec, Anda dapat langsung berinteraksi dengan kontainer tanpa perlu terlebih dahulu berinteraksi dengan sistem operasi kontainer host, membuka port masuk, atau mengelola kunci SSH. Anda dapat menggunakan ECS Exec untuk menjalankan perintah di atau mendapatkan shell ke wadah yang berjalan pada EC2 instance Amazon atau di. AWS Fargate Hal tersebut mempermudah pengumpulan informasi diagnostik dan memecahkan masalah kesalahan dengan cepat. Misalnya, dalam konteks pengembangan, Anda dapat menggunakan ECS Exec untuk berinteraksi dengan mudah dengan berbagai proses dalam wadah Anda dan memecahkan masalah aplikasi Anda. Dan dalam skenario produksi, Anda dapat menggunakannya untuk mendapatkan akses break-glass ke kontainer Anda untuk men-debug masalah.

Anda dapat menjalankan perintah dalam wadah Linux atau Windows yang sedang berjalan menggunakan ECS Exec dari Amazon ECS API, AWS Command Line Interface (AWS CLI) AWS SDKs, atau AWS Copilot CLI. Untuk detail tentang penggunaan ECS Exec, serta panduan video, menggunakan Copilot AWS CLI, lihat dokumentasi Copilot. GitHub

Anda juga dapat menggunakan ECS Exec untuk mempertahankan kebijakan kontrol akses yang lebih ketat. Dengan mengaktifkan fitur ini secara selektif, Anda dapat mengendalikan siapa yang dapat menjalankan perintah dan ditugas mana mereka dapat menjalankan perintah tersebut. Dengan

log dari setiap perintah dan outputnya, Anda dapat menggunakan ECS Exec untuk melihat tugas mana yang dijalankan dan Anda dapat menggunakan CloudTrail untuk mengaudit siapa yang mengakses wadah.

Pertimbangan

Untuk topik ini, Anda harus terbiasa dengan aspek-aspek berikut yang terlibat dengan penggunaan ECS Exec:

- ECS Exec saat ini tidak didukung menggunakan file. AWS Management Console

  Halaman detail cluster menampilkan logging dan kunci terkelola AWS KMS pelanggan yang digunakan untuk mengenkripsi data antara klien lokal dan penampung.
- ECS Exec mungkin tidak berfungsi seperti yang diharapkan saat berjalan pada sistem operasi yang tidak didukung oleh Systems Manager. Untuk informasi tentang sistem operasi yang didukung, lihat Jenis sistem operasi di Panduan AWS Systems Manager Pengguna.
- ECS Exec didukung untuk tugas-tugas yang berjalan pada infrastruktur berikut:
  - Wadah Linux di Amazon EC2 pada AMI yang dioptimalkan Amazon ECS, termasuk Bottlerocket
  - Wadah Linux dan Windows pada instance eksternal (Amazon ECS Anywhere)
  - Wadah Linux dan Windows di AWS Fargate
  - Kontainer Windows di Amazon EC2 pada Windows Amazon ECS-dioptimalkan berikut AMIs (dengan versi agen kontainer `1.56` atau yang lebih baru):
    - Amazon ECS yang dioptimalkan Windows Server 2022 AMI Penuh
    - Server Windows 2022 AMI Inti yang dioptimalkan Amazon ECS
    - Amazon ECS yang dioptimalkan Windows Server 2019 AMI Penuh
    - Amazon ECS yang dioptimalkan Windows Server 2019 Core AMI
    - Amazon ECS yang dioptimalkan Windows Server 20H2 Core AMI
- Jika Anda mengonfigurasi proxy HTTP untuk tugas Anda, setel variabel `NO_PROXY` lingkungan `"NO_PROXY=169.254.169.254,169.254.170.2"` agar dapat melewati proxy EC2 misalnya metadata dan lalu lintas peran IAM. Jika Anda tidak mengonfigurasi variabel `NO_PROXY` lingkungan, mungkin ada kegagalan saat mengambil metadata instance atau kredensyal peran IAM dari titik akhir metadata di dalam wadah. Mengatur variabel `NO_PROXY` lingkungan seperti yang disarankan menyaring metadata dan lalu lintas IAM sehingga permintaan untuk `169.254.169.254 and 169.254.170.2` tidak melalui proxy. `HTTP`
- ECS Exec dan Amazon VPC

  - Jika Anda menggunakan antarmuka titik akhir Amazon VPC dengan Amazon ECS, Anda harus membuat antarmuka titik akhir Amazon VPC untuk Systems Manager Session Manager (). `ssmmessages` Untuk informasi selengkapnya tentang titik akhir VPC Systems Manager, lihat Menggunakan AWS PrivateLink untuk menyiapkan titik akhir VPC untuk Session Manager di Panduan Pengguna.AWS Systems Manager
  - Jika Anda menggunakan antarmuka titik akhir Amazon VPC dengan Amazon ECS, dan Anda menggunakan AWS KMS key untuk enkripsi, maka Anda harus membuat antarmuka titik akhir Amazon VPC untuk. AWS KMS keyUntuk informasi selengkapnya, lihat Menghubungkan ke AWS KMS key melalui titik akhir VPC di Panduan Pengembang.AWS Key Management Service
  - Bila Anda memiliki tugas yang berjalan di EC2 instans Amazon, gunakan mode `awsvpc` jaringan. Jika Anda tidak memiliki akses internet (seperti tidak dikonfigurasi untuk menggunakan gateway NAT), Anda harus membuat antarmuka titik akhir Amazon VPC untuk Systems Manager Session Manager (). `ssmmessages` Untuk informasi selengkapnya tentang pertimbangan mode `awsvpc` jaringan, lihat Pertimbangan. Untuk informasi selengkapnya tentang titik akhir VPC Systems Manager, lihat Menggunakan AWS PrivateLink untuk menyiapkan titik akhir VPC untuk Session Manager di Panduan Pengguna.AWS Systems Manager
- ECS Exec dan SSM
  - Ketika pengguna menjalankan perintah pada wadah menggunakan ECS Exec, perintah ini dijalankan sebagai pengguna. `root` Agen SSM dan proses turunannya berjalan sebagai root bahkan ketika Anda menentukan ID pengguna untuk penampung.
  - Agen SSM mengharuskan sistem file kontainer dapat ditulis untuk membuat direktori dan file yang diperlukan. Oleh karena itu, membuat sistem file root hanya-baca menggunakan parameter ketentuan tugas `readonlyRootFilesystem`, atau metode lainnya, tidak didukung.
  - Meskipun memulai sesi SSM di luar `execute-command` tindakan dimungkinkan, ini mengakibatkan sesi tidak dicatat dan dihitung terhadap batas sesi. Kami menyarankan untuk membatasi akses ini dengan menolak `ssm:start-session` tindakan menggunakan kebijakan IAM. Untuk informasi selengkapnya, lihat Membatasi akses ke tindakan Memulai Sesi.
- Fitur berikut berjalan sebagai wadah sespan. Oleh karena itu, Anda harus menentukan nama kontainer untuk menjalankan perintah.
  - Pemantauan Runtime
  - Layanan Connect
- Pengguna dapat menjalankan semua perintah yang tersedia dalam konteks kontainer. Tindakan berikut mungkin mengakibatkan proses tanpa induk dan zombie: mengakhiri proses utama

kontainer, mengakhiri agen perintah, dan menghapus dependensi. Untuk membersihkan proses zombie, sebaiknya tambahkan flag `initProcessEnabled` untuk ketentuan tugas Anda.

- ECS Exec menggunakan beberapa CPU dan memori. Anda akan ingin mengakomodasinya saat menentukan alokasi sumber daya CPU dan memori dalam ketentuan tugas Anda.
- Anda harus menggunakan AWS CLI versi `1.22.3` atau yang lebih baru atau AWS CLI versi `2.3.6` atau yang lebih baru. Untuk informasi tentang cara memperbarui AWS CLI, lihat Menginstal atau memperbarui versi terbaru dari AWS CLI Panduan AWS Command Line Interface Pengguna Versi 2.
- Anda hanya dapat memiliki satu sesi ECS Exec per namespace ID proses (PID). Jika Anda berbagi namespace PID dalam tugas, Anda hanya dapat memulai sesi ECS Exec ke dalam satu wadah.
- Sesi ECS Exec memiliki waktu tunggu idle 20 menit. Nilai ini tidak dapat diubah.
- Anda tidak dapat mengaktifkan ECS Exec untuk tugas yang ada. Itu hanya dapat dihidupkan untuk tugas-tugas baru.
- Anda tidak dapat menggunakan ECS Exec saat digunakan `run-task` untuk meluncurkan tugas di klaster yang menggunakan penskalaan terkelola dengan penempatan asinkron (meluncurkan tugas tanpa instance).
- Anda tidak dapat menjalankan ECS Exec terhadap kontainer Microsoft Nano Server.

Prasyarat

Sebelum Anda mulai menggunakan ECS Exec, pastikan Anda telah menyelesaikan tindakan ini:

- Instal dan konfigurasikan AWS CLI. Untuk informasi selengkapnya, lihat Memulai dengan AWS CLI.
- Instal plugin Session Manager untuk file AWS CLI. Untuk informasi selengkapnya, lihat Instal plugin Pengelola Sesi untuk AWS CLI.
- Anda harus menggunakan peran tugas dengan izin yang sesuai untuk ECS Exec. Untuk informasi selengkapnya, lihat Peran IAM Tugas.
- ECS Exec memiliki persyaratan versi tergantung pada apakah tugas Anda di-host di Amazon EC2 atau: AWS Fargate
  - Jika Anda menggunakan Amazon EC2, Anda harus menggunakan AMI Amazon ECS yang dioptimalkan yang dirilis setelah 20 Januari 2021, dengan versi agen 1.50.2 atau lebih tinggi. Untuk informasi selengkapnya, lihat Amazon ECS dioptimalkan AMIs.

- Jika Anda menggunakan AWS Fargate, Anda harus menggunakan versi platform `1.4.0` atau lebih tinggi (Linux) atau `1.0.0` (Windows). Untuk informasi selengkapnya, lihat Versi platform AWS Fargate.

Arsitektur

ECS Exec menggunakan Manajer Sesi AWS Systems Manager (SSM) untuk membuat koneksi dengan wadah yang sedang berjalan dan menggunakan kebijakan AWS Identity and Access Management (IAM) untuk mengontrol akses ke perintah yang sedang berjalan dalam wadah yang sedang berjalan. Hal ini dimungkinkan dengan mengikat pemasangan biner SSM agent yang diperlukan ke dalam kontainer. Amazon ECS atau AWS Fargate agen bertanggung jawab untuk memulai agen inti SSM di dalam wadah bersama kode aplikasi Anda. Untuk informasi selengkapnya, lihat Systems Manager Session Manager.

Anda dapat mengaudit pengguna mana yang mengakses container menggunakan `ExecuteCommand` event in AWS CloudTrail dan mencatat setiap perintah (dan outputnya) ke Amazon S3 atau Amazon CloudWatch Logs. Untuk mengenkripsi data antara klien lokal dan wadah dengan kunci enkripsi Anda sendiri, Anda harus memberikan kunci AWS Key Management Service (AWS KMS).

Menggunakan ECS Exec

Perubahan ketentuan tugas opsional

Jika Anda menyetel parameter definisi tugas `initProcessEnabled` ke`true`, ini memulai proses init di dalam wadah. Ini menghapus proses anak agen SSM zombie yang ditemukan. Contoh disediakan seperti berikut.

```
{
    "taskRoleArn": "ecsTaskRole",
    "networkMode": "awsvpc",
    "requiresCompatibilities": [
        "EC2",
        "FARGATE"
    ],
    "executionRoleArn": "ecsTaskExecutionRole",
    "memory": ".5 gb",
    "cpu": ".25 vcpu",
    "containerDefinitions": [
```

```
        {
            "name": "amazon-linux",
            "image": "amazonlinux:latest",
            "essential": true,
            "command": ["sleep","3600"],
            "linuxParameters": {
                "initProcessEnabled": true
            }
        }
    ],
    "family": "ecs-exec-task"
}
```

Mengaktifkan ECS Exec untuk tugas dan layanan Anda

Anda dapat mengaktifkan fitur ECS Exec untuk layanan dan tugas mandiri Anda dengan menentukan `--enable-execute-command` tanda saat menggunakan salah satu AWS CLI perintah berikut: `create-service`,,, `update-service`atau. `start-task` `run-task`

Misalnya, jika Anda menjalankan perintah berikut, fitur ECS Exec diaktifkan untuk layanan yang baru dibuat yang berjalan di Fargate. Untuk informasi selengkapnya tentang membuat layanan, lihat buat-layanan.

```
aws ecs create-service \
    --cluster cluster-name \
    --task-definition task-definition-name \
    --enable-execute-command \
    --service-name service-name \
    --launch-type FARGATE \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
 \
    --desired-count 1
```

Setelah Anda mengaktifkan ECS Exec untuk tugas, Anda dapat menjalankan perintah berikut untuk mengonfirmasi tugas siap digunakan. Jika properti `lastStatus` dari `ExecuteCommandAgent` terdaftar sebagai `RUNNING` dan properti `enableExecuteCommand` diatur ke `true`, maka tugas Anda sudah siap.

```
aws ecs describe-tasks \
    --cluster cluster-name \
```

```
    --tasks task-id
```

Potongan output berikut adalah contoh apa yang mungkin Anda lihat.

```
{
    "tasks": [
        {
            ...
            "containers": [
                {
                    ...
                    "managedAgents": [
                        {
                            "lastStartedAt": "2021-03-01T14:49:44.574000-06:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                    ]
                }
            ],
            ...
            "enableExecuteCommand": true,
            ...
        }
    ]
}
```

Menjalankan perintah menggunakan ECS Exec

Setelah Anda telah mengonfirmasi bahwa `ExecuteCommandAgent` sedang berjalan, Anda dapat membuka shell interaktif pada kontainer Anda dengan menggunakan perintah berikut. Jika tugas Anda berisi beberapa kontainer, Anda harus menentukan nama kontainer menggunakan `--container` bendera. Amazon ECS hanya mendukung memulai sesi interaktif, jadi Anda harus menggunakan bendera. `--interactive`

Perintah berikut akan menjalankan `/bin/sh` perintah interaktif terhadap wadah bernama *`container-name`* untuk tugas dengan ID dari*`task-id`*.

*`task-id`*Itu adalah Nama Sumber Daya Amazon (ARN) dari tugas tersebut.

```
aws ecs execute-command --cluster cluster-name \
    --task task-id \
```

```
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

Logging menggunakan ECS Exec

Mengaktifkan login dalam tugas dan layanan Anda

> **Important**
>
> Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga. Amazon ECS juga menyediakan metrik pemantauan yang disediakan tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch.

Amazon ECS menyediakan konfigurasi default untuk perintah logging yang dijalankan menggunakan ECS Exec. Defaultnya adalah mengirim CloudWatch log ke Log menggunakan driver `awslogs` log yang dikonfigurasi dalam definisi tugas Anda. Jika Anda ingin memberikan konfigurasi khusus, AWS CLI mendukung `--configuration` bendera untuk kedua `update-cluster` perintah `create-cluster` dan. Gambar kontainer memerlukan `script` dan diinstal `cat` agar log perintah diunggah dengan benar ke Amazon S3 CloudWatch atau Log. Untuk informasi selengkapnya tentang pembuatan klaster, lihat buat-klaster.

> **Note**
>
> Konfigurasi ini hanya menangani pencatatan sesi `execute-command`. Konfigurasi tersebut tidak memengaruhi pencatatan aplikasi Anda.

Contoh berikut membuat klaster dan kemudian mencatat output ke CloudWatch Log LogGroup bernama `cloudwatch-log-group-name` dan bucket Amazon S3 Anda bernama. `s3-bucket-name`

Anda harus menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi grup log saat Anda `true` menyetel `CloudWatchEncryptionEnabled` opsi. Untuk informasi tentang cara mengenkripsi grup log, lihat Mengenkripsi data CloudWatch log di Log menggunakan AWS Key Management Service, di Amazon CloudWatch Logs Panduan Pengguna.

```
aws ecs create-cluster \
```

```
    --cluster-name cluster-name \
    --configuration executeCommandConfiguration="{ \
        kmsKeyId=string, \
        logging=OVERRIDE, \
        logConfiguration={ \
            cloudWatchLogGroupName=cloudwatch-log-group-name, \
            cloudWatchEncryptionEnabled=true, \
            s3BucketName=s3-bucket-name, \
            s3EncryptionEnabled=true, \
            s3KeyPrefix=demo \
        } \
    }"
```

`logging`Properti menentukan perilaku kemampuan logging ECS Exec:

- `NONE`: logging dimatikan.
- `DEFAULT`: log dikirim ke `awslogs` driver yang dikonfigurasi. Jika driver tidak dikonfigurasi, maka tidak ada log yang disimpan.
- `OVERRIDE`: log dikirim ke CloudWatch Log Amazon yang disediakan LogGroup, bucket Amazon S3, atau keduanya.

Izin IAM diperlukan untuk Amazon CloudWatch Log atau Amazon S3 Logging

Untuk mengaktifkan logging, peran tugas Amazon ECS yang direferensikan dalam definisi tugas Anda harus memiliki izin tambahan. Izin tambahan ini dapat ditambahkan sebagai kebijakan untuk peran tugas. Mereka berbeda tergantung pada apakah Anda mengarahkan log Anda ke Amazon CloudWatch Log atau Amazon S3.

Amazon CloudWatch Logs

Kebijakan contoh berikut menambahkan izin Amazon CloudWatch Logs yang diperlukan.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:CreateLogStream",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:arn:aws:: 111122223333:log-group:/aws/
ecs/cloudwatch-log-group-name:*"
        }
   ]
}
```

Amazon S3

Kebijakan contoh berikut menambahkan izin Amazon S3 yang diperlukan.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
           "Effect": "Allow",
           "Action": [
               "s3:GetEncryptionConfiguration"
           ],
           "Resource": "arn:aws:s3:::s3-bucket-name"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name/*"
        }
    ]
}
```

Izin IAM diperlukan untuk enkripsi menggunakan milik Anda sendiri AWS KMS key (kunci KMS)

Secara default, data yang ditransfer antara klien lokal Anda dan penampung menggunakan enkripsi TLS 1.2 yang AWS menyediakan. Untuk mengenkripsi data lebih lanjut menggunakan kunci KMS Anda sendiri, Anda harus membuat kunci KMS dan menambahkan `kms:Decrypt` izin ke peran IAM tugas Anda. Izin ini digunakan oleh kontainer Anda untuk mendekripsi data. Untuk informasi selengkapnya tentang membuat kunci KMS, lihat Membuat kunci.

Anda menambahkan kebijakan sebaris berikut ke peran IAM tugas yang memerlukan izin. AWS KMS Untuk informasi selengkapnya, lihat Izin ECS Exec.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource":
 "arn:aws:kms:arn:aws::111122223333:key/12345678-1234-1234-1234-123456789012"
        }
    ]
}
```

Agar data dienkripsi menggunakan kunci KMS Anda sendiri, pengguna atau grup yang menggunakan `execute-command` tindakan harus diberikan izin. `kms:GenerateDataKey`

Contoh kebijakan berikut untuk pengguna atau grup Anda berisi izin yang diperlukan untuk menggunakan kunci KMS Anda sendiri. Anda harus menentukan Nama Sumber Daya Amazon (ARN) dari kunci KMS Anda.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:arn:aws::111122223333:key/key"
        }
    ]
}
```

Menggunakan kebijakan IAM untuk membatasi akses ke ECS Exec

Anda membatasi akses pengguna ke tindakan API perintah eksekusi dengan menggunakan satu atau beberapa kunci kondisi kebijakan IAM berikut:

- `aws:ResourceTag/clusterTagKey`
- `ecs:ResourceTag/clusterTagKey`
- `aws:ResourceTag/taskTagKey`
- `ecs:ResourceTag/taskTagKey`
- `ecs:container-name`
- `ecs:cluster`
- `ecs:task`
- `ecs:enable-execute-command`

Dengan contoh berikut kebijakan IAM, pengguna dapat menjalankan perintah dalam kontainer yang berjalan dalam tugas dengan tag yang memiliki `environment` kunci dan `development` nilai dan dalam klaster yang diberi `cluster-name` nama.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ExecuteCommand",
                "ecs:DescribeTasks"
            ],
            "Resource": [
                  "arn:aws:ecs:arn:aws::111122223333:task/cluster-name/*",
                  "arn:aws:ecs:arn:aws::111122223333:cluster/cluster-name"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:ResourceTag/environment": "development"
                }
            }
        }
    ]
}
```

Dengan contoh kebijakan IAM berikut, pengguna tidak dapat menggunakan `execute-command` API saat nama kontainer. `production-app`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ecs:ExecuteCommand"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "ecs:container-name": "production-app"
                }
            }
        }
    ]
}
```

Dengan kebijakan IAM berikut, pengguna hanya dapat meluncurkan tugas ketika ECS Exec dimatikan.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:enable-execute-command": "false"
                }
            }
        }
    ]
}
```

> **Note**
>
> Karena tindakan API `execute-command` hanya berisi sumber daya tugas dan klaster dalam sebuah permintaan, maka yang dievaluasi adalah tanda klaster dan tugas.

Untuk informasi selengkapnya tentang kunci kondisi kebijakan IAM, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Membatasi akses ke tindakan Memulai Sesi

Meskipun memulai sesi SSM di wadah Anda di luar ECS Exec dimungkinkan, ini berpotensi mengakibatkan sesi tidak dicatat. Sesi yang dimulai di luar ECS Exec juga dihitung terhadap kuota sesi. Sebaiknya batasi akses ini dengan menolak `ssm:start-session` tindakan secara langsung untuk tugas Amazon ECS Anda menggunakan kebijakan IAM. Anda dapat menolak akses ke semua tugas Amazon ECS atau ke tugas tertentu berdasarkan tag yang digunakan.

Berikut ini adalah contoh kebijakan IAM yang menolak akses ke `ssm:start-session` tindakan untuk tugas di semua Wilayah dengan nama cluster tertentu. Anda dapat menyertakan wildcard dengan *`cluster-name`* secara opsional.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": [
            "arn:aws:ecs:arn:aws:;: 111122223333:task/cluster-name/*"
            ]
        }
    ]
}
```

Berikut ini adalah contoh kebijakan IAM yang menolak akses ke `ssm:start-session` tindakan pada sumber daya di semua Wilayah yang ditandai dengan kunci tag `Task-Tag-Key` dan nilai tag. `Exec-Task`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ecs:*:*:task/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
                }
            }
        }
    ]
}
```

Untuk bantuan terkait masalah apa pun yang mungkin Anda hadapi saat menggunakan Amazon ECS Exec, lihat Memecahkan masalah dengan Exec.

AWS Compute Optimizer rekomendasi untuk Amazon ECS

AWS Compute Optimizer menghasilkan rekomendasi untuk tugas Amazon ECS dan ukuran wadah. Untuk informasi selengkapnya, lihat Apa itu AWS Compute Optimizer? dalam Panduan Pengguna AWS Compute Optimizer .

Rekomendasi ukuran tugas dan wadah untuk layanan Amazon ECS di AWS Fargate

AWS Compute Optimizer menghasilkan rekomendasi untuk layanan Amazon ECS di AWS Fargate. AWS Compute Optimizer merekomendasikan CPU tugas dan ukuran memori tugas dan CPU kontainer, memori kontainer, dan ukuran reservasi memori kontainer. Rekomendasi ini ditampilkan di halaman berikut dari konsol Compute Optimizer.

- Rekomendasi untuk layanan Amazon ECS di halaman Fargate
- Layanan Amazon ECS di halaman detail Fargate

Untuk informasi selengkapnya, lihat Melihat rekomendasi untuk layanan Amazon ECS di Fargate di Panduan Pengguna AWS Compute Optimizer .

Pemecahan masalah Amazon ECS

Anda mungkin perlu memecahkan masalah dengan penyeimbang beban, tugas, layanan, atau instance kontainer Anda. Bab ini membantu Anda menemukan informasi diagnostik dari agen penampung Amazon ECS, daemon Docker pada instance container, dan log peristiwa layanan di konsol Amazon ECS.

Untuk informasi tentang tugas yang dihentikan, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya
Mengatasi kesalahan tugas yang dihentikan.	Melihat Amazon ECS menghentikan kesalahan tugas
Lihat kesalahan tugas yang dihentikan.	Mengatasi kesalahan tugas Amazon ECS yang dihentikan
Tinjau kode kesalahan tugas yang dihentikan.	Amazon ECS menghentikan pesan kesalahan tugas
Tinjau kesalahan CannotPullContainer tugas.	CannotPullContainer kesalahan tugas di Amazon ECS
Lihat permintaan peran IAM tugas.	Melihat permintaan peran IAM untuk tugas Amazon ECS

Untuk informasi tentang kesalahan layanan, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya
Lihat pesan acara layanan.	Melihat pesan acara layanan Amazon ECS
Tinjau pesan acara layanan.	Pesan acara layanan Amazon ECS

Tindakan	Pelajari selengkapnya	
Tinjau masalah penyeimbang beban.	Memecahkan masalah penyeimbang beban layanan di Amazon ECS	
Tinjau masalah penskalaan otomatis layanan.	Pemecahan masalah penskalaan otomatis layanan di Amazon ECS	

Untuk informasi tentang kesalahan definisi tugas, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya	
Mengatasi kesalahan memori definisi tugas.	Memecahkan masalah kesalahan kesalahan CPU atau memori Amazon ECS definisi tugas yang tidak valid	

Untuk informasi tentang kesalahan agen Amazon ECS, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya	
Lihat log agen kontainer Amazon ECS.	Melihat log agen kontainer Amazon ECS	
Pelajari cara mengumpulkan log Amazon ECS.	Mengumpulkan log kontainer dengan kolektor log Amazon ECS	
Ambil detail diagnostik dengan agen Amazon ECS.	Ambil detail diagnostik Amazon ECS dengan introspeksi agen	

Untuk informasi tentang kesalahan Docker, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya
Gunakan diagnostik Docker.	Diagnostik Docker di Amazon ECS
Aktifkan mode debug Docker.	Mengonfigurasi keluaran verbose dari daemon Docker di Amazon ECS
Memecahkan masalah kesalahan API Docker 500.	Memecahkan masalah Docker di Amazon `API error (500): devmapper` ECS

Untuk informasi tentang kesalahan ECS Exec dan Amazon ECS Anywhere, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya
Memecahkan masalah ECS Exec.	Memecahkan masalah Amazon ECS Exec
Memecahkan masalah Amazon ECS Anywhere.	Memecahkan masalah Amazon ECS Anywhere

Untuk informasi tentang masalah pembatasan, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya
Pelajari tentang kuota pelambatan Fargate.	AWS Fargate kuota pelambatan
Pelajari praktik terbaik untuk pelambatan Amazon ECS.	Menangani masalah pelambatan Amazon ECS

Untuk informasi tentang error API, lihat salah satu dari berikut ini.

Tindakan	Pelajari selengkapnya	
Mengatasi kesalahan API.	Alasan kegagalan Amazon ECS API	

Mengatasi kesalahan tugas Amazon ECS yang dihentikan

Ketika tugas Anda gagal dimulai, Anda melihat pesan kesalahan di konsol dan di parameter `describe-tasks` output (`stoppedReason`dan`stoppedCode`).

Anda dapat melihat tugas yang dihentikan di konsol selama satu jam. Untuk melihat tugas yang dihentikan, Anda harus mengubah opsi filter. Untuk informasi selengkapnya, lihat Melihat Amazon ECS menghentikan kesalahan tugas.

Halaman-halaman berikut memberikan informasi tentang tugas yang dihentikan.

- Pelajari tentang perubahan pada pesan galat tugas yang dihentikan.

  Amazon ECS menghentikan pembaruan pesan kesalahan tugas
- Lihat tugas Anda yang dihentikan sehingga Anda bisa mendapatkan informasi tentang penyebabnya.

  Melihat Amazon ECS menghentikan kesalahan tugas
- Pelajari tentang pesan kesalahan tugas yang dihentikan dan kemungkinan alasan kesalahan tersebut.

  Amazon ECS menghentikan pesan kesalahan tugas
- Pelajari cara memverifikasi konektivitas tugas yang dihentikan dan memperbaiki kesalahan.

  Memverifikasi Amazon ECS menghentikan konektivitas tugas

Amazon ECS menghentikan pembaruan pesan kesalahan tugas

Mulai 14 Juni 2024 tim Amazon ECS mengubah pesan kesalahan tugas yang dihentikan seperti yang dijelaskan dalam tabel berikut. Tidak `stopCode` akan berubah. Jika aplikasi Anda bergantung pada string pesan kesalahan yang tepat, Anda harus memperbarui aplikasi Anda dengan string baru. Untuk bantuan dengan pertanyaan atau masalah, hubungi AWS Dukungan.

> **Note**
>
> Kami menyarankan Anda untuk tidak mengandalkan pesan kesalahan untuk otomatisasi Anda, karena pesan kesalahan dapat berubah.

CannotPullContainerError

Pesan kesalahan lama	Pesan kesalahan baru	
CannotPullContainerError: Respons kesalahan dari daemon: akses tarik ditolak untuk*repository* , repositori tidak ada atau mungkin memerlukan 'docker login': ditolak: Pengguna: *roleARN*	• CannotPullContainerError: Tugas tidak dapat menarik gambar. Periksa apakah peran memiliki izin untuk menarik gambar dari registri. Respons kesalahan dari daemon: tarik akses ditolak untuk*repository* , repositori tidak ada atau mungkin memerlukan 'docker login': denied: User: *roleARN* is not authorized to perform: ecr: BatchGetImage on resource: *image* karena tidak ada kebijakan berbasis identitas yang mengizinkan ecr: action. BatchGetImage • CannotPullContainerError: Tugas tidak dapat menarik gambar. Periksa apakah gambar itu ada. Respons kesalahan dari daemon: akses tarik ditolak untuk*repository* , repositori tidak ada atau	

Pesan kesalahan lama	Pesan kesalahan baru	
	mungkin memerlukan 'docker login': ditolak: akses yang diminta ke sumber daya ditolak.	
CannotPullContainerError: Tanggapan kesalahan dari daemon: Dapatkan*imageURI*: net/http: permintaan dibatalkan sambil menunggu koneksi	CannotPullContainerError: Tugas tidak dapat menarik gambar. Periksa konfigurasi jaringan Anda. Respons kesalahan dari daemon: Dapatkan: net/http*image*: permintaan dibatalkan sambil menunggu koneksi (Klien.Timeout terlampaui saat menunggu header)	
CannotPullContainerError<ip><port>: ref pull telah dicoba ulang 5 kali: gagal menyalin:: gagal buka httpReadSeeker: gagal melakukan permintaan: Get: dial tcp*registry-uri* : timeout i/o	CannotPullContainerError: Tugas tidak dapat menarik *image-uri* dari registri*registry-uri* . Ada masalah koneksi antara tugas dan registri. <port>Periksa konfigurasi jaringan tugas Anda.: gagal menyalin: httpReadSeeker: gagal buka: gagal melakukan permintaan: Get*registry-uri* : dial tcp<ip>: timeout i/o	

ResourceNotFoundException

Pesan kesalahan lama	Pesan kesalahan baru	
Mengambil data rahasia dari AWS Secrets Manager	ResourceNotFoundException: Tugas tidak dapat	

Pesan kesalahan lama	Pesan kesalahan baru	
dalam *region* region: secret: *sercretARN* ResourceNotFoundException: Secrets Manager tidak dapat menemukan rahasia yang ditentukan.	mengambil rahasia dengan ARN *sercretARN* 'dari. AWS Secrets Manager Periksa apakah rahasia ada di Wilayah yang ditentukan. ResourceNotFoundException: Mengambil data rahasia dari AWS Secrets Manager dalam wilayah*region*: rahasia: *sercretARN* ResourceNotFoundException: Secrets Manager tidak dapat menemukan rahasia yang ditentukan.	

ResourceInitializationError

Pesan kesalahan lama	Pesan kesalahan baru	
ResourceInitializationError <ip><port>: tidak dapat menarik rahasia atau autentikasi registri: pengambilan sumber daya eksekusi gagal: tidak dapat mengambil autentikasi registri ecr: panggilan layanan telah dicoba ulang 3 kali:: permintaan kirim gagal disebabkan oleh RequestError: Posting "https://api.ecr.us-east-1.amazonaws.com/": dial tcp:: batas waktu. i/o Silakan periksa konfigurasi jaringan tugas Anda.	ResourceInitializationError: tidak dapat menarik rahasia atau autentikasi registri: Tugas tidak dapat menarik autentikasi registri dari Amazon ECR: Ada masalah koneksi antara tugas dan Amazon ECR. Periksa konfigurasi jaringan tugas Anda. RequestError: permintaan kirim gagal disebabkan oleh: Posting "https://api.ecr.us-east-1.amazonaws.com“: dial tcp<ip>:<port>: batas waktu i/o	

Pesan kesalahan lama	Pesan kesalahan baru	
ResourceInitializationError: tidak dapat menarik rahasia atau autentikasi registri: pengambilan sumber daya eksekusi gagal: tidak dapat mengambil rahasia dari ssm: panggilan layanan telah dicoba ulang 5 kali:: konteks permintaan dibatalkan disebabkan oleh RequestCanceled: batas waktu konteks terlampaui	ResourceInitializationError: tidak dapat menarik rahasia atau autentikasi registri: tidak dapat mengambil rahasia dari ssm: Tugas tidak dapat menarik rahasia. AWS Systems Manager Ada masalah koneksi antara tugas dan AWS Systems Manager Parameter Store. Periksa konfigurasi jaringan tugas Anda. RequestCanceled: konteks permintaan dibatalkan disebabkan oleh: batas waktu konteks terlampaui	
ResourceInitializationError: gagal mengunduh file env: perintah unduhan file: aliran kesalahan tidak kosong:: konteks permintaan dibatalkan disebabkan oleh RequestCanceled: batas waktu konteks terlampaui	ResourceInitializationError: gagal mengunduh file env: Tugas tidak dapat mengunduh file variabel lingkungan dari Amazon S3. Ada masalah koneksi antara tugas dan Amazon S3. Periksa konfigurasi jaringan tugas Anda. panggilan layanan telah dicoba ulang 5 kali:: konteks permintaan dibatalkan disebabkan oleh RequestCanceled: batas waktu konteks terlampaui	

Pesan kesalahan lama	Pesan kesalahan baru	
ResourceInitializationError: gagal memvalidasi argumen logger: :signal:killed	ResourceInitializationError: gagal memvalidasi argumen logger: Tugas tidak dapat menemukan grup CloudWatch log Amazon yang ditentukan dalam definisi tugas. Ada masalah koneksi antara tugas dan Amazon CloudWatch. Periksa konfigurasi jaringan Anda.: sinyal: mati	
ResourceInitializationError: tidak dapat menarik rahasia atau autentikasi registri: perintah tarik gagal:: sinyal: terbunuh	ResourceInitializationError: tidak dapat menarik rahasia atau autentikasi registri: Periksa konfigurasi jaringan tugas Anda.: sinyal: mati	

Melihat Amazon ECS menghentikan kesalahan tugas

Jika Anda kesulitan memulai tugas, tugas Anda mungkin berhenti karena kesalahan aplikasi atau konfigurasi. Contohnya, Anda menjalankan tugas dan tugas menampilkan status `PENDING` dan kemudian menghilang.

Jika tugas Anda dibuat oleh layanan Amazon ECS, tindakan yang dilakukan Amazon ECS untuk mempertahankan layanan dipublikasikan dalam acara layanan. Anda dapat melihat peristiwa di AWS Management Console, AWS CLI AWS SDKs, Amazon ECS API, atau alat yang menggunakan SDKs dan API. Peristiwa ini termasuk Amazon ECS menghentikan dan menggantikan tugas karena kontainer dalam tugas telah berhenti berjalan, atau telah gagal terlalu banyak pemeriksaan kesehatan dari Elastic Load Balancing.

Jika tugas Anda berjalan pada instance container di Amazon EC2 atau komputer eksternal, Anda juga dapat melihat log runtime container dan Amazon ECS Agent. Log ini ada di host Amazon EC2 instance atau komputer eksternal. Untuk informasi selengkapnya, lihat Melihat log agen kontainer Amazon ECS.

Prosedur

Console

AWS Management Console

Langkah-langkah berikut dapat digunakan untuk memeriksa tugas yang dihentikan untuk kesalahan menggunakan konsol. Untuk melihat tugas yang dihentikan, Anda harus mengubah opsi filter.

Tugas yang dihentikan hanya muncul di konsol selama 1 jam.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pada *name* halaman Cluster:, pilih tab Tugas.
5. Konfigurasikan filter untuk menampilkan tugas yang dihentikan. Untuk Filter status yang diinginkan, pilih Berhenti.

   Opsi Berhenti menampilkan tugas yang dihentikan dan Status apa pun yang diinginkan menampilkan semua tugas Anda.
6. Pilih tugas yang dihentikan untuk diperiksa.
7. Di baris untuk tugas yang dihentikan, di kolom Status Terakhir, pilih Berhenti.

   Jendela pop-up menampilkan alasan berhenti.

AWS CLI

1. Daftar tugas yang berhenti dalam sebuah klaster. Output berisi Amazon Resource Name (ARN) dari tugas, yang Anda butuhkan untuk menjelaskan tugas tersebut.

   ```
   aws ecs list-tasks \
        --cluster cluster_name \
        --desired-status STOPPED \
        --region region
   ```
2. Jelaskan tugas yang dihentikan untuk mengambil informasi. Untuk informasi selengkapnya, lihat mendeskripsikan tugas di Referensi. AWS Command Line Interface

```
aws ecs describe-tasks \
     --cluster cluster_name \
     --tasks arn:aws:ecs:region:account_id:task/cluster_name/task_ID \
     --region region
```

Gunakan parameter output berikut.

- `stopCode`- Kode berhenti menunjukkan mengapa tugas dihentikan, misalnya ResourceInitializationError
- `StoppedReason`- Alasan tugas berhenti.
- `reason`(dalam `containers` struktur) - Alasannya memberikan rincian tambahan tentang wadah yang dihentikan.

Langkah selanjutnya

Lihat tugas Anda yang dihentikan sehingga Anda bisa mendapatkan informasi tentang penyebabnya. Untuk informasi selengkapnya, lihat Amazon ECS menghentikan pesan kesalahan tugas.

Amazon ECS menghentikan pesan kesalahan tugas

Berikut ini adalah kemungkinan pesan kesalahan yang mungkin Anda terima ketika tugas Anda berhenti secara tak terduga.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Kode kesalahan tugas yang dihentikan memiliki kategori yang terkait dengannya, misalnya "ResourceInitializationError”. Untuk mendapatkan informasi lebih lanjut tentang setiap kategori, lihat berikut ini:

Kategori	Pelajari selengkapnya	
TaskFailedToStart	Memecahkan masalah kesalahan Amazon ECS TaskFailedToStart	

Kategori	Pelajari selengkapnya
ResourceInitializationError	Memecahkan masalah kesalahan Amazon ECS ResourceInitializationError
ResourceNotFoundException	Memecahkan masalah kesalahan Amazon ECS ResourceNotFoundException
SpotInterruptionError	Memecahkan masalah kesalahan Amazon ECS SpotInterruption
InternalError	Memecahkan masalah kesalahan Amazon ECS InternalError
OutOfMemoryError	Memecahkan masalah kesalahan Amazon ECS OutOfMemoryError
ContainerRuntimeError	Memecahkan masalah kesalahan Amazon ECS ContainerRuntimeError
ContainerRuntimeTimeoutError	Memecahkan masalah kesalahan Amazon ECS ContainerRuntimeTimeoutError
CannotStartContainerError	Memecahkan masalah kesalahan Amazon ECS CannotStartContainerError
CannotStopContainerError	Memecahkan masalah kesalahan Amazon ECS CannotStopContainerError

Kategori	Pelajari selengkapnya	
CannotInspectContainerError	Memecahkan masalah kesalahan Amazon ECS CannotInspectContainerError	
CannotCreateVolumeError	Memecahkan masalah kesalahan Amazon ECS CannotCreateVolumeError	
CannotPullContainer	CannotPullContainer kesalahan tugas di Amazon ECS	

Memecahkan masalah kesalahan Amazon ECS TaskFailedToStart

Berikut ini adalah beberapa pesan `TaskFailedToStart` kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

EC2 Kesalahan tak terduga saat mencoba Membuat Antarmuka Jaringan dengan penetapan IP publik diaktifkan di subnet '***subnet-id***

Ini terjadi ketika tugas Fargate yang menggunakan mode `awsvpc` jaringan dan berjalan di subnet dengan alamat IP publik, dan subnet tidak memiliki cukup alamat IP.

Jumlah alamat IP yang tersedia tersedia di halaman detail subnet di EC2 konsol Amazon, atau dengan menggunakan`describe-subnets`. Untuk informasi selengkapnya, lihat Melihat subnet Anda di Panduan Pengguna Amazon VPC.

Untuk memperbaiki masalah ini, Anda dapat membuat subnet baru untuk menjalankan tugas Anda.

InternalError: ***<reason>***

Kesalahan ini terjadi ketika lampiran ENI diminta. Amazon secara EC2 asinkron menangani penyediaan ENI. Proses penyediaan membutuhkan waktu. Amazon ECS memiliki batas waktu jika ada waktu tunggu yang lama atau kegagalan yang tidak dilaporkan. Ada kalanya ENI disediakan,

tetapi laporan tersebut datang ke Amazon ECS setelah batas waktu kegagalan. Dalam kasus ini, Amazon ECS melihat kegagalan tugas yang dilaporkan dengan ENI yang sedang digunakan.

Definisi tugas yang dipilih tidak kompatibel dengan strategi komputasi yang dipilih

Kesalahan ini terjadi ketika Anda memilih definisi tugas dengan tipe peluncuran yang tidak cocok dengan tipe kapasitas cluster. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS. Anda harus memilih definisi tugas yang cocok dengan penyedia kapasitas yang ditetapkan ke klaster Anda.

Tidak dapat melampirkan antarmuka jaringan ke indeks perangkat yang tidak digunakan

Kesalahan ini terjadi ketika Ketika menggunakan jenis `awsvpc` jaringan dan tidak ada cukup CPU/memory untuk tugas. Pertama, periksa CPU untuk instance. Untuk informasi selengkapnya, lihat spesifikasi jenis EC2 instans Amazon di jenis EC2 instans Amazon. Ambil nilai CPU untuk instance dan kalikan dengan jumlah ENIs untuk instance. Gunakan nilai itu e dalam definisi tugas.

AGEN

Instans kontainer tempat Anda mencoba untuk meluncurkan tugas memiliki agen yang saat ini terputus. Untuk mencegah waktu tunggu diperpanjang untuk penempatan tugas, permintaan ditolak.

Untuk informasi tentang cara memecahkan masalah agen yang terputus, lihat Bagaimana cara memecahkan masalah agen Amazon ECS yang terputus.

Memecahkan masalah kesalahan Amazon ECS ResourceInitializationError

Berikut ini adalah beberapa pesan `ResourceInitialization` kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Kesalahan

- Tugas tidak dapat menarik otentikasi registri dari Amazon ECR. Ada masalah koneksi antara tugas dan Amazon ECR. Periksa konfigurasi jaringan tugas Anda.
- Tugas tidak dapat mengunduh file variabel lingkungan dari Amazon S3. Ada masalah koneksi antara tugas dan Amazon S3. Periksa konfigurasi jaringan tugas Anda.
- Tugas tidak dapat menarik rahasia dari AWS Systems Manager Parameter Store. Periksa koneksi jaringan Anda antara tugas dan AWS Systems Manager.

- Tugas tidak dapat menarik rahasia dari AWS Secrets Manager. Ada masalah koneksi antara tugas dan Secrets Manager. Periksa konfigurasi jaringan tugas Anda.
- Tugas tidak dapat menarik rahasia dari Secrets Manager. Tugas tidak dapat mengambil rahasia dengan ARN secretARN " dari Secrets Manager. Periksa apakah rahasia ada di Wilayah yang ditentukan.
- perintah tarik gagal: tidak dapat menarik rahasia atau autentikasi registri Periksa konfigurasi jaringan tugas Anda.
- Tugas tidak dapat menemukan grup CloudWatch log Amazon yang ditentukan dalam definisi tugas. Ada masalah koneksi antara tugas dan Amazon CloudWatch. Periksa konfigurasi jaringan Anda.
- gagal menginisialisasi driver logging
- gagal memanggil perintah EFS utils untuk menyiapkan volume EFS

Tugas tidak dapat menarik otentikasi registri dari Amazon ECR. Ada masalah koneksi antara tugas dan Amazon ECR. Periksa konfigurasi jaringan tugas Anda.

Kesalahan ini menunjukkan bahwa tugas tidak dapat terhubung ke Amazon ECR.

Periksa koneksi antara tugas dan Amazon ECR. Untuk informasi, lihat Memverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat mengunduh file variabel lingkungan dari Amazon S3. Ada masalah koneksi antara tugas dan Amazon S3. Periksa konfigurasi jaringan tugas Anda.

Kesalahan ini terjadi ketika tugas Anda tidak dapat mengunduh file lingkungan Anda dari Amazon S3.

Periksa koneksi antara tugas dan titik akhir VPC Amazon S3. Untuk informasi, lihat Memverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat menarik rahasia dari AWS Systems Manager Parameter Store. Periksa koneksi jaringan Anda antara tugas dan AWS Systems Manager.

Kesalahan ini terjadi ketika tugas Anda tidak dapat menarik gambar yang ditentukan dalam definisi tugas menggunakan kredensi di Systems Manager.

Periksa koneksi antara tugas dan titik akhir VPC Systems Manager. Untuk informasi, lihat Memverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat menarik rahasia dari AWS Secrets Manager. Ada masalah koneksi antara tugas dan Secrets Manager. Periksa konfigurasi jaringan tugas Anda.

Kesalahan ini terjadi ketika tugas Anda tidak dapat menarik gambar yang ditentukan dalam definisi tugas menggunakan kredensil di Secrets Manager.

Kesalahan menunjukkan bahwa ada masalah konektivitas jaringan antara titik akhir VPC Systems Manager dan tugas.

Untuk informasi tentang cara memverifikasi konektivitas antara tugas dan titik akhir, lihatMemverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat menarik rahasia dari Secrets Manager. Tugas tidak dapat mengambil rahasia dengan ARN ***secretARN*** " dari Secrets Manager. Periksa apakah rahasia ada di Wilayah yang ditentukan.

Kesalahan ini terjadi ketika tugas Anda tidak dapat menarik gambar yang ditentukan dalam definisi tugas menggunakan kredensil di Secrets Manager.

Masalah ini disebabkan oleh salah satu alasan berikut:

Penyebab kesalahan..	Lakukan ini...	
Masalah konektivitas jaringan antara titik akhir VPC Secrets Manager dan tugas. Masalahnya adalah masalah jaringan ketika Anda melihat salah satu string berikut dalam pesan kesalahan: • panggil tcp • panggil udp • <ip>:<port>: i/o batas waktu • net/http: Batas waktu jabat tangan TLS • baca: waktu koneksi habis	Verifikasi konektivitas antara tugas dan titik akhir Secrets Manager. Untuk informasi selengkapnya, lihat Memverifikasi Amazon ECS menghentikan konektivitas tugas.	

Penyebab kesalahan..	Lakukan ini...	
• Client.Timeout terlampaui sambil menunggu header • net/http: permintaa n dibatalkan sambil menunggu koneksi • sinyal: terbunuh • batas waktu konteks terlampaui		
Peran eksekusi tugas yang ditentukan dalam definisi tugas tidak memiliki izin untuk Secrets Manager.	Tambahkan izin yang diperlukan untuk Secrets Manager ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Izin Secrets Manager atau Systems Manager.	
Rahasia ARN tidak ada	Periksa apakah ARN ada di Secrets Manager. Untuk informasi tentang melihat gambar Anda, lihat Menemukan rahasia di Secrets Manager di Panduan Pengembang Secrets Manager.	

perintah tarik gagal: tidak dapat menarik rahasia atau autentikasi registri Periksa konfigurasi jaringan tugas Anda.

Kesalahan ini terjadi ketika tugas Anda tidak dapat terhubung ke Amazon ECR, Systems Manager, atau Secrets Manager. Ini karena kesalahan konfigurasi di jaringan Anda.

Untuk memperbaiki masalah ini, verifikasi keterkaitan antara tugas dan Amazon ECR. Anda juga perlu memeriksa konektivitas antara tugas Anda dan layanan yang menyimpan rahasia Anda

(Systems Manager, atau Secrets Manager). Untuk informasi selengkapnya, lihat Memverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat menemukan grup CloudWatch log Amazon yang ditentukan dalam definisi tugas. Ada masalah koneksi antara tugas dan Amazon CloudWatch. Periksa konfigurasi jaringan Anda.

Kesalahan ini terjadi ketika tugas Anda gagal menemukan grup CloudWatch log yang Anda tentukan dalam definisi tugas.

Kesalahan menunjukkan bahwa ada masalah konektivitas jaringan antara titik akhir CloudWatch VPC dan tugas.

Untuk informasi tentang cara memverifikasi konektivitas antara tugas dan titik akhir, lihatMemverifikasi Amazon ECS menghentikan konektivitas tugas.

gagal menginisialisasi driver logging

Kesalahan ini terjadi ketika tugas Anda gagal menemukan grup CloudWatch log yang Anda tentukan dalam definisi tugas.

Kesalahan menunjukkan bahwa CloudWatch grup dalam definisi tugas tidak ada.

Gunakan langkah-langkah berikut untuk menemukan yang hilang CloudWatch.

1. Jalankan perintah berikut untuk mendapatkan informasi definisi tugas.

   ```
   aws ecs describe-task-definition \
       --task-definition task-def-name
   ```

   Lihatlah output untuk setiap kontainer dan perhatikan `awslogs-group` nilainya.

   ```
   "logConfiguration": {
                   "logDriver": "awslogs",
                   "options": {
                       "awslogs-group": "/ecs/example-group",
                       "awslogs-create-group": "true",
                       "awslogs-region": "us-east-1",
                       "awslogs-stream-prefix": "ecs"
                   },
   ```

2. Verifikasi bahwa grup tersebut tersedia CloudWatch untuk informasi selengkapnya, lihat Bekerja dengan grup log dan aliran log di Panduan Pengguna CloudWatch Log Amazon.

Masalahnya adalah grup yang ditentukan dalam definisi tugas salah, atau grup log tidak ada.

3. Perbaiki masalah.

Masalahnya adalah...	Lakukan ini...
Grup log yang salah ditentukan dalam definisi tugas.	Perbarui definisi tugas untuk menyertakan konfigura si grup log dalam definisi container. Untuk informasi tentang memperbarui definisi tugas, lihat Memperbar ui definisi tugas Amazon ECS menggunakan konsol atau RegisterTaskDefini tiondi Referensi API Amazon Elastic Container Service.
Grup log tidak ada di CloudWatch	Buat grup log. Untuk informasi selengkapnya, lihat Membuat grup CloudWatc h log di Log di Panduan Pengguna CloudWatch Log Amazon.

gagal memanggil perintah EFS utils untuk menyiapkan volume EFS

Masalah berikut mungkin mencegah Anda memasang volume Amazon EFS pada permintaan Anda:

- Sistem file Amazon EFS tidak dikonfigurasi dengan benar.
- Tugas tidak memiliki izin yang diperlukan.
- Ada masalah yang terkait dengan konfigurasi jaringan dan VPC.

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihat Mengapa saya tidak dapat memasang volume Amazon EFS saya pada AWS Fargate tugas saya di AWS re:post.

Memecahkan masalah kesalahan Amazon ECS ResourceNotFoundException

Berikut ini adalah beberapa pesan `ResourceNotFoundException` kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Tugas tidak dapat mengambil rahasia dengan ARN ***sercretARN*** 'dari. AWS Secrets Manager Periksa apakah rahasia ada di Wilayah yang ditentukan.

Kesalahan ini terjadi ketika tugas tidak dapat mengambil rahasia dari Secrets Manager. Ini berarti bahwa rahasia yang ditentukan dalam definisi tugas (dan terkandung dalam pesan kesalahan) tidak ada di Secrets Manager.

Wilayah ada dalam pesan kesalahan.

Mengambil data rahasia dari AWS Secrets Manager dalam region*region*: secret: *sercretARN* ResourceNotFoundException: Secrets Manager tidak dapat menemukan rahasia yang ditentukan.

Untuk informasi tentang menemukan rahasia, lihat Menemukan rahasia AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna.

Gunakan tabel berikut untuk menentukan dan mengatasi kesalahan.

Isu	Tindakan	
Rahasianya ada di Wilayah yang berbeda dari definisi tugas.	a. Buat rahasia di Wilayah yang sama dengan tugas. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia. b. Perbarui definisi tugas dengan rahasia baru. Untuk informasi selengkapnya, lihat Memperbarui definisi tugas Amazon ECS menggunakan konsol atau RegisterTaskDefinitiondi	

Isu	Tindakan	
	Referensi API Amazon Elastic Container Service.	
Definisi tugas memiliki ARN rahasia yang salah. Rahasia yang benar ada di Secrets Manager.	Perbarui definisi tugas dengan rahasia yang benar. Untuk informasi selengkapnya, lihat Memperbarui definisi tugas Amazon ECS menggunakan konsol atau RegisterTaskDefinitiondi Referensi API Amazon Elastic Container Service.	
Rahasianya sudah tidak ada lagi.	a. Buat rahasia di Wilayah yang sama dengan tugas. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia. b. Perbarui definisi tugas dengan rahasia baru. Untuk informasi selengkapnya, lihat Memperbarui definisi tugas Amazon ECS menggunakan konsol atau RegisterTaskDefinitiondi Referensi API Amazon Elastic Container Service.	

Memecahkan masalah kesalahan Amazon ECS SpotInterruption

`SpotInterruption`Kesalahan memiliki alasan berbeda untuk Fargate dan tipe EC2 peluncuran.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Jenis peluncuran Fargate

`SpotInterruption`Kesalahan terjadi ketika tidak ada kapasitas Fargate Spot atau ketika Fargate mengambil kembali kapasitas Spot.

Anda dapat menjalankan tugas di beberapa Availability Zone untuk memungkinkan kapasitas yang lebih besar.

EC2 jenis peluncuran

Kesalahan ini terjadi ketika tidak ada Instans Spot yang tersedia atau EC2 mengambil kembali kapasitas Instans Spot.

Anda dapat menjalankan instans di beberapa Availability Zone untuk memungkinkan kapasitas yang lebih besar.

Memecahkan masalah kesalahan Amazon ECS InternalError

Berlaku untuk: Jenis peluncuran Fargate

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

`InternalError`Kesalahan saat agen menemukan kesalahan internal yang tidak terduga dan tidak terkait runtime.

Kesalahan ini hanya terjadi jika menggunakan versi platform `1.4` atau yang lebih baru.

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihatAmazon ECS menghentikan pesan kesalahan tugas.

Memecahkan masalah kesalahan Amazon ECS OutOfMemoryError

Berikut ini adalah beberapa pesan OutOfMemoryError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

kontainer mati karena penggunaan memori

Kesalahan ini terjadi ketika wadah keluar karena proses dalam wadah menghabiskan lebih banyak memori daripada yang dialokasikan dalam definisi tugas, atau karena kendala host atau sistem operasi.

Memecahkan masalah kesalahan Amazon ECS ContainerRuntimeError

Berikut ini adalah beberapa pesan ContainerRuntimeError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

ContainerRuntimeError

Kesalahan ini terjadi ketika agen menerima kesalahan tak terduga dari `containerd` untuk operasi khusus runtime. Kesalahan ini biasanya disebabkan oleh kegagalan internal pada agen atau `containerd` runtime.

Kesalahan ini hanya terjadi jika Anda menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihat Mengapa tugas Amazon ECS saya Dihentikan di AWS re:Post.

Memecahkan masalah kesalahan Amazon ECS ContainerRuntimeTimeoutError

Berikut ini adalah beberapa pesan ContainerRuntimeTimeoutError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Tidak dapat beralih ke berjalan; habis waktu setelah menunggu 1m atau kesalahan batas waktu Docker

Kesalahan ini terjadi ketika sebuah kontainer tidak dapat melakukan transisi ke `RUNNING` atau kondisi `STOPPED` dalam periode timeout. Alasan dan nilai timeout yang disediakan dalam kesalahan pesan.

Memecahkan masalah kesalahan Amazon ECS CannotStartContainerError

Berikut ini adalah beberapa pesan CannotStartContainerError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

gagal mendapatkan status kontainer: ***<reason>***

Kesalahan ini terjadi ketika kontainer tidak dapat dimulai.

Jika kontainer Anda mencoba untuk melebihi memori yang ditentukan di sini, kontainer akan dihentikan. Tingkatkan memori yang disajikan ke wadah. Ini adalah `memory` parameter dalam definisi tugas. Untuk informasi selengkapnya, lihat the section called “Memori”.

Memecahkan masalah kesalahan Amazon ECS CannotStopContainerError

Berikut ini adalah beberapa pesan CannotStopContainerError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

CannotStopContainerError

Kesalahan ini terjadi ketika kontainer tidak dapat dihentikan.

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihat Mengapa tugas Amazon ECS saya Dihentikan di AWS re:Post.

Memecahkan masalah kesalahan Amazon ECS CannotInspectContainerError

Berikut ini adalah beberapa pesan CannotInspectContainerError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

CannotInspectContainerError

Kesalahan ini terjadi ketika agen kontainer tidak dapat menjelaskan kontainer melalui waktu aktif kontainer.

Saat menggunakan versi platform `1.3` atau sebelumnya, agen Amazon ECS mengembalikan alasan dari Docker.

Saat menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows), agen Fargate mengembalikan alasannya. `containerd`

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihat Mengapa tugas Amazon ECS saya Dihentikan di AWS re:Post.

Memecahkan masalah kesalahan Amazon ECS CannotCreateVolumeError

Berikut ini adalah beberapa pesan CannotCreateVolumeError kesalahan dan tindakan yang dapat Anda ambil untuk memperbaiki kesalahan.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

CannotCreateVolumeError

Kesalahan ini terjadi ketika agen tidak dapat membuat volume pemasangan yang ditentukan dalam ketentuan tugas.

Kesalahan ini hanya terjadi jika Anda menggunakan versi platform `1.4.0` atau yang lebih baru (Linux) `1.0.0` atau yang lebih baru (Windows).

Untuk informasi tentang cara men-debug dan memperbaiki masalah ini, lihat Mengapa tugas Amazon ECS saya Dihentikan di AWS re:Post.

CannotPullContainer kesalahan tugas di Amazon ECS

Kesalahan berikut menunjukkan bahwa tugas gagal dimulai karena Amazon ECS tidak dapat mengambil gambar kontainer yang ditentukan.

> **Note**
>
> Versi platform 1.4 Fargate memotong pesan kesalahan yang panjang.

Untuk memeriksa tugas yang dihentikan untuk pesan kesalahan menggunakan AWS Management Console, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Kesalahan

- Tugas tidak dapat menarik gambar. Periksa apakah peran memiliki izin untuk menarik gambar dari registri.
- Tugas tidak dapat menarik 'image-name' dari repositori Amazon ECR ". repository URI Ada masalah koneksi antara tugas dan Amazon ECR. Periksa konfigurasi jaringan tugas Anda.
- Tugas tidak dapat menarik gambar. Periksa konfigurasi jaringan Anda

- CannotPullContainerError: tarik manifes gambar telah dicoba ulang 5 kali: gagal menyelesaikan ref
- Kesalahan API (500): Dapatkan https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: permintaan dibatalkan sambil menunggu koneksi
- Kesalahan API
- tulis/var/lib/docker/tmp/GetImageBlob111111111: tidak ada ruang tersisa di perangkat
- ERROR: toomanyrequests: Terlalu Banyak Permintaan atau Anda telah mencapai batas tingkat tarik Anda.
- Tanggapan kesalahan dari daemon: Dapatkanurl: net/http: permintaan dibatalkan sambil menunggu koneksi
- ref pull telah dicoba ulang 1 kali: gagal menyalin:: gagal buka httpReaderSeeker: kode status tak terduga
- akses tarik ditolak
- perintah tarik gagal: panik: kesalahan runtime: alamat memori tidak valid atau dereferensi pointer nil
- kesalahan menarik gambar conf/error menarik konfigurasi gambar
- Konteks dibatalkan

Tugas tidak dapat menarik gambar. Periksa apakah peran memiliki izin untuk menarik gambar dari registri.

Kesalahan ini menunjukkan bahwa tugas tidak dapat menarik gambar yang ditentukan dalam definisi tugas karena masalah izin.

Untuk menyelesaikan masalah ini:

1. Periksa apakah gambar ada di*`repository`*. Untuk informasi tentang melihat gambar Anda, lihat Melihat detail gambar di Amazon ECR di Panduan Pengguna Amazon Elastic Container Registry.
2. Verifikasi bahwa *`role-arn`* memiliki izin yang benar untuk menarik gambar.

   Untuk informasi tentang cara memperbarui peran, lihat Memperbarui izin untuk peran di Panduan AWS Identity and Access Management Penggunaan.

   Tugas menggunakan salah satu peran berikut:

   - Untuk tugas dengan tipe peluncuran Fargate, ini adalah peran eksekusi tugas. Untuk informasi tentang izin tambahan untuk Amazon ECR,. Tugas Fargate menarik gambar Amazon ECR melalui izin titik akhir antarmuka

- Untuk tugas dengan tipe EC2 peluncuran, ini adalah peran instance container. Untuk informasi tentang izin tambahan untuk Amazon ECR,. Izin Amazon ECR

Tugas tidak dapat menarik '*image-name*' dari repositori Amazon ECR ''. *repository URI* Ada masalah koneksi antara tugas dan Amazon ECR. Periksa konfigurasi jaringan tugas Anda.

Kesalahan ini menunjukkan bahwa tugas tidak dapat terhubung ke Amazon ECR. Periksa koneksi ke *repository URI* repositori.

Untuk informasi tentang cara memverifikasi dan menyelesaikan masalah, lihatMemverifikasi Amazon ECS menghentikan konektivitas tugas.

Tugas tidak dapat menarik gambar. Periksa konfigurasi jaringan Anda

Kesalahan ini menunjukkan bahwa tugas tidak dapat terhubung ke Amazon ECR.

Untuk informasi tentang cara memverifikasi dan menyelesaikan masalah, lihatMemverifikasi Amazon ECS menghentikan konektivitas tugas.

CannotPullContainerError: tarik manifes gambar telah dicoba ulang 5 kali: gagal menyelesaikan ref

Kesalahan ini menunjukkan bahwa tugas tidak dapat menarik gambar.

Untuk mengatasi ini, Anda dapat:

- Verifikasi bahwa gambar yang ditentukan dalam definisi tugas cocok dengan gambar di repositori.
- Amazon ECS memaksa stabilitas versi gambar. Jika gambar asli tidak lagi tersedia, Anda mendapatkan kesalahan ini. Tag gambar adalah bagian dari menegakkan perilaku ini. Ubah gambar dalam definisi tugas dari menggunakan:latest sebagai tag ke versi tertentu. Untuk informasi selengkapnya, lihat Resolusi gambar kontainer.

Untuk informasi tentang cara memverifikasi dan menyelesaikan masalah, lihatMemverifikasi Amazon ECS menghentikan konektivitas tugas.

Kesalahan API (500): Dapatkan https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: permintaan dibatalkan sambil menunggu koneksi

Kesalahan ini menunjukkan bahwa koneksi habis waktu, karena rute ke internet tidak ada.

Untuk mengatasi masalah ini, Anda dapat:

- Untuk tugas di subnet publik, tentukan DIAKTIFKAN untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS.
- Untuk tugas di subnet privat, tentukan NONAKTIF untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas, dan mengonfigurasi gateway NAT di VPC Anda untuk merutekan permintaan menuju internet. Untuk informasi lebih lanjut, lihat NAT Gateway di Panduan Pengguna Amazon VPC.

Kesalahan API

Kesalahan ini menunjukkan bahwa ada masalah koneksi dengan titik akhir Amazon ECR.

Untuk informasi tentang cara mengatasi masalah ini, lihat Bagaimana cara mengatasi kesalahan Amazon ECR "CannotPullContainerError: Kesalahan API" di Amazon ECS di situs web. Dukungan

tulis/var/lib/docker/tmp/***GetImageBlob111111111***: tidak ada ruang tersisa di perangkat

Kesalahan ini menunjukkan bahwa ruang disk tidak mencukupi.

Untuk mengatasi masalah ini, kosongkan ruang disk.

Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat menggunakan perintah berikut untuk mengambil 20 file terbesar di sistem file Anda:

```
du -Sh / | sort -rh | head -20
```

Contoh output:

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
...
```

Dalam beberapa kasus, volume root mungkin diisi oleh wadah yang sedang berjalan. Jika kontainer menggunakan driver log `json-file` default tanpa batas `max-size`, terdapat kemungkinan bahwa berkas log bertanggung jawab pada sebagian besar ruang yang digunakan. Anda dapat

menggunakan perintah `docker ps` untuk memverifikasi kontainer yang menggunakan ruang dengan pemetaan nama direktori dari output di atas untuk ID kontainer. Sebagai contoh:

```
CONTAINER ID   IMAGE                                COMMAND             CREATED
 STATUS              PORTS                               NAMES
50501b5f4cbf   amazon/amazon-ecs-agent:latest   "/agent"            4 days ago
 Up 4 days                                              ecs-agent
```

Secara default, saat menggunakan driver log `json-file`, Docker menangkap output standar (dan kesalahan standar) dari semua kontainer Anda dan menulis output standar dan kesalahan dalam file menggunakan format JSON. Anda dapat mengatur `max-size` sebagai pilihan pengandar catatan, yang dapat mencegah berkas log mengambil terlalu banyak ruang. Untuk informasi selengkapnya, lihat Driver logging file JSON di dokumentasi Docker.

Berikut ini adalah cuplikan ketentuan kontainer yang menunjukkan bagaimana menggunakan pilihan ini:

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "256m"
    }
}
```

Alternatifnya, jika log kontainer Anda menghabiskan terlalu banyak ruang disk, adalah dengan menggunakan driver `awslogs` log. Driver `awslogs` log mengirimkan log ke CloudWatch, yang membebaskan ruang disk yang seharusnya digunakan untuk log kontainer Anda pada instance kontainer. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke CloudWatch .

Anda mungkin perlu memperbarui ukuran disk yang dapat diakses Docker.

Untuk informasi selengkapnya, lihat CannotPullContainerError: tidak ada ruang tersisa di perangkat.

ERROR: toomanyrequests: Terlalu Banyak Permintaan atau Anda telah mencapai batas tingkat tarik Anda.

Kesalahan ini menunjukkan bahwa ada pembatasan laju Docker Hub.

Jika Anda menerima salah satu kesalahan berikut, kemungkinan Anda mencapai tingkat batasan Docker Hub:

Untuk informasi selengkapnya tentang tingkat batasan Docker Hub, lihat Memahami pembatasan pada tingkat Docker Hub.

Jika Anda telah meningkatkan batas laju Docker Hub dan Anda perlu mengautentikasi tarikan Docker untuk instance container Anda, lihat Autentikasi registri pribadi untuk instance kontainer.

Tanggapan kesalahan dari daemon: Dapatkan*url*: net/http: permintaan dibatalkan sambil menunggu koneksi

Kesalahan ini menunjukkan bahwa koneksi habis waktu, karena rute ke internet tidak ada.

Untuk mengatasi masalah ini, Anda dapat:

- Untuk tugas di subnet publik, tentukan DIAKTIFKAN untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS.
- Untuk tugas di subnet privat, tentukan NONAKTIF untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas, dan mengonfigurasi gateway NAT di VPC Anda untuk merutekan permintaan menuju internet. Untuk informasi lebih lanjut, lihat NAT Gateway di Panduan Pengguna Amazon VPC.

ref pull telah dicoba ulang 1 kali: gagal menyalin:: gagal buka httpReaderSeeker: kode status tak terduga

Kesalahan ini menunjukkan bahwa ada kegagalan saat menyalin gambar.

Untuk mengatasi masalah ini, tinjau salah satu artikel berikut:

- Untuk tugas Fargate, lihat Bagaimana cara mengatasi kesalahan “cannotpullcontainererror” untuk tugas Amazon ECS saya di Fargate.
- Untuk tugas lain, lihat Bagaimana cara mengatasi kesalahan “cannotpullcontainererror” untuk tugas Amazon ECS saya.

akses tarik ditolak

Kesalahan ini menunjukkan bahwa tidak ada akses ke gambar.

Untuk mengatasi masalah ini, Anda mungkin perlu mengautentikasi klien Docker Anda dengan Amazon ECR Untuk informasi selengkapnya, lihat Autentikasi registri pribadi di Panduan Pengguna Amazon ECR.

perintah tarik gagal: panik: kesalahan runtime: alamat memori tidak valid atau dereferensi pointer nil

Kesalahan ini menunjukkan bahwa tidak ada akses ke gambar karena alamat memori yang tidak valid atau dereferensi pointer nil.

Untuk menyelesaikan masalah ini:

- Periksa apakah Anda memiliki aturan grup keamanan untuk mencapai Amazon S3.
- Saat Anda menggunakan titik akhir gateway, Anda harus menambahkan rute di tabel rute untuk mengakses titik akhir.

kesalahan menarik gambar conf/error menarik konfigurasi gambar

Kesalahan ini menunjukkan batas tarif telah tercapai atau ada kesalahan jaringan:

Untuk mengatasi masalah ini, lihat Bagaimana cara mengatasi kesalahan "CannotPullContainerError" di Tugas Jenis EC2 Peluncuran Amazon ECS saya.

Konteks dibatalkan

Kesalahan ini menunjukkan bahwa konteksnya dibatalkan.

Penyebab umum kesalahan ini adalah karena VPC yang digunakan tugas Anda tidak memiliki rute untuk menarik gambar penampung dari Amazon ECR.

Memverifikasi Amazon ECS menghentikan konektivitas tugas

Ada kalanya tugas berhenti karena masalah konektivitas jaringan. Ini mungkin masalah intermiten, tetapi kemungkinan besar disebabkan karena tugas tidak dapat terhubung ke titik akhir.

Menguji konektivitas tugas

Anda dapat menggunakan `AWSSupport-TroubleshootECSTaskFailedToStart` runbook untuk menguji konektivitas tugas. Saat Anda menggunakan runbook, Anda memerlukan informasi sumber daya berikut:

- ID tugas

  Gunakan ID tugas gagal terbaru.
- Cluster tempat tugas itu berada

Untuk informasi tentang cara menggunakan runbook, lihat `AWSSupport-TroubleshootECSTaskFailedToStart`di referensi Runbook AWS Systems Manager otomatisasi.

Runbook menganalisis tugas. Anda dapat melihat hasilnya di bagian Output untuk masalah berikut yang dapat mencegah tugas dimulai:

- Konektivitas jaringan ke registri kontainer yang dikonfigurasi
- Konektivitas titik akhir VPC
- Konfigurasi aturan grup keamanan

Memperbaiki masalah titik akhir VPC

Saat hasil `AWSSupport-TroubleshootECSTaskFailedToStart` runbook menunjukkan masalah titik akhir VPC, periksa konfigurasi berikut:

- VPC tempat Anda membuat titik akhir harus menggunakan DNS Pribadi.
- Pastikan Anda memiliki AWS PrivateLink titik akhir untuk layanan yang tugas tidak dapat terhubung ke dalam VPC yang sama dengan tugas. Untuk informasi lebih lanjut, lihat salah satu dari berikut ini:

Layanan	Informasi titik akhir VPC untuk layanan
Amazon ECR	Titik akhir VPC antarmuka Amazon ECR ()AWS PrivateLink
Systems Manager	Meningkatkan keamanan EC2 instans dengan menggunakan titik akhir VPC untuk Systems Manager
Secrets Manager	Menggunakan titik akhir AWS Secrets Manager VPC
CloudWatch	CloudWatch Titik akhir VPC

Layanan	Informasi titik akhir VPC untuk layanan	
Amazon S3	AWS PrivateLink untuk Amazon S3	

- Konfigurasikan aturan keluar untuk subnet tugas yang memungkinkan HTTPS pada port 443 DNS (TCP) lalu lintas. Untuk informasi selengkapnya, lihat Mengonfigurasi aturan grup keamanan di Panduan Pengguna Amazon Elastic Compute Cloud.
- Jika Anda menggunakan server domain nama kustom, maka konfirmasikan pengaturan kueri DNS. Kueri harus memiliki akses keluar pada port 53, dan menggunakan protokol UDP dan TCP. Selain itu, ia harus memiliki akses HTTPS pada port 443. Untuk informasi selengkapnya, lihat Aturan grup keamanan Coonfigure di Panduan Pengguna Amazon Elastic Compute Cloud.
- Jika subnet memiliki ACL jaringan, aturan ACL berikut diperlukan:
  - Aturan keluar yang memungkinkan lalu lintas yang memungkinkan lalu lintas di port 1024-65535.
  - Aturan masuk yang memungkinkan lalu lintas TCP pada port 443.

  Untuk informasi tentang cara mengonfigurasi aturan, lihat Mengontrol lalu lintas ke subnet menggunakan jaringan ACLs di Panduan Pengguna Amazon Virtual Private Cloud.

Memperbaiki masalah jaringan

Jika hasil `AWSSupport-TroubleshootECSTaskFailedToStart` runbook menunjukkan masalah jaringan, periksa konfigurasi berikut:

Tugas yang menggunakan mode jaringan awsvpc di subnet publik

Lakukan konfigurasi berikut berdasarkan runbook:

- Untuk tugas di subnet publik, tentukan DIAKTIFKAN untuk Tetapkan secara otomatis IP publik saat meluncurkan tugas. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS.
- Anda memerlukan gateway untuk menangani lalu lintas internet. Tabel rute untuk subnet tugas harus memiliki rute untuk lalu lintas ke gateway.

  Untuk informasi selengkapnya, lihat Menambahkan dan menghapus rute dari tabel rute di Panduan Pengguna Amazon Virtual Private Cloud.

Jenis gerbang	Tujuan tabel rute	Target tabel Rout
NAT	0.0.0.0/0	ID gerbang NAT
gateway internet	0.0.0.0/0	ID gerbang internet

- Jika subnet tugas memiliki ACL jaringan, aturan ACL berikut diperlukan:
  - Aturan keluar yang memungkinkan lalu lintas di port 1024-65535.
  - Aturan masuk yang memungkinkan lalu lintas TCP pada port 443.

  Untuk informasi tentang cara mengonfigurasi aturan, lihat Mengontrol lalu lintas ke subnet menggunakan jaringan ACLs di Panduan Pengguna Amazon Virtual Private Cloud.

Tugas yang menggunakan mode jaringan awsvpc di subnet pribadi

Lakukan konfigurasi berikut berdasarkan runbook:

- Pilih DISABLED untuk Auto-assign IP publik saat meluncurkan tugas.
- Konfigurasikan gateway NAT di VPC Anda untuk merutekan permintaan ke internet. Untuk informasi selengkapnya, lihat Gateway NAT di Panduan Pengguna Amazon Virtual Private Cloud.
- Tabel rute untuk subnet tugas harus memiliki rute untuk lalu lintas ke gateway NAT.

  Untuk informasi selengkapnya, lihat Menambahkan dan menghapus rute dari tabel rute di Panduan Pengguna Amazon Virtual Private Cloud.

Jenis gerbang	Tujuan tabel rute	Target tabel Rout
NAT	0.0.0.0/0	ID gerbang NAT

- Jika subnet tugas memiliki ACL jaringan, aturan ACL berikut diperlukan:
  - Aturan keluar yang memungkinkan lalu lintas di port 1024-65535.
  - Aturan masuk yang memungkinkan lalu lintas TCP pada port 443.

  Untuk informasi tentang cara mengonfigurasi aturan, lihat Mengontrol lalu lintas ke subnet menggunakan jaringan ACLs di Panduan Pengguna Amazon Virtual Private Cloud.

Tugas yang tidak menggunakan mode jaringan awsvpc di subnet publik

Lakukan konfigurasi berikut berdasarkan runbook:

- Pilih Aktifkan untuk Tetapkan IP Otomatis di bawah EC2Instans Jaringan untuk Amazon saat Anda membuat klaster.

  Opsi ini memberikan alamat IP publik ke antarmuka jaringan utama instance.
- Anda memerlukan gateway untuk menangani lalu lintas internet. Tabel rute untuk subnet instance harus memiliki rute untuk lalu lintas ke gateway.

  Untuk informasi selengkapnya, lihat Menambahkan dan menghapus rute dari tabel rute di Panduan Pengguna Amazon Virtual Private Cloud.

Jenis gerbang	Tujuan tabel rute	Target tabel Rout
NAT	0.0.0.0/0	ID gerbang NAT
gateway internet	0.0.0.0/0	ID gerbang internet

- Jika subnet instance memiliki ACL jaringan, aturan ACL berikut diperlukan:
  - Aturan keluar yang memungkinkan lalu lintas di port 1024-65535.
  - Aturan masuk yang memungkinkan lalu lintas TCP pada port 443.

  Untuk informasi tentang cara mengonfigurasi aturan, lihat Mengontrol lalu lintas ke subnet menggunakan jaringan ACLs di Panduan Pengguna Amazon Virtual Private Cloud.

Tugas yang tidak menggunakan mode jaringan awsvpc di subnet pribadi

Lakukan konfigurasi berikut berdasarkan runbook:

- Pilih Matikan untuk Tetapkan IP Otomatis di bawah EC2Instans Jaringan untuk Amazon saat Anda membuat klaster.
- Konfigurasikan gateway NAT di VPC Anda untuk merutekan permintaan ke internet. Untuk informasi lebih lanjut, lihat NAT Gateway di Panduan Pengguna Amazon VPC.
- Tabel rute untuk subnet instance harus memiliki rute untuk lalu lintas ke gateway NAT.

  Untuk informasi selengkapnya, lihat Menambahkan dan menghapus rute dari tabel rute di Panduan Pengguna Amazon Virtual Private Cloud.

Jenis gerbang	Tujuan tabel rute	Target tabel Rout
NAT	0.0.0.0/0	ID gerbang NAT

- Jika subnet tugas memiliki ACL jaringan, aturan ACL berikut diperlukan:
  - Aturan keluar yang memungkinkan lalu lintas di port 1024-65535.
  - Aturan masuk yang memungkinkan lalu lintas TCP pada port 443.

  Untuk informasi tentang cara mengonfigurasi aturan, lihat Mengontrol lalu lintas ke subnet menggunakan jaringan ACLs di Panduan Pengguna Amazon Virtual Private Cloud.

Melihat permintaan peran IAM untuk tugas Amazon ECS

Bila Anda menggunakan penyedia untuk kredensi tugas Anda dalam peran IAM, penyedia meminta disimpan dalam log audit. Log audit menerapkan pengaturan rotasi log yang sama sebagai log agen kontainer. variabel konfigurasi agen kontainer `ECS_LOG_ROLLOVER_TYPE`, `ECS_LOG_MAX_FILE_SIZE_MB`, dan `ECS_LOG_MAX_ROLL_COUNT` dapat diatur untuk mempengaruhi perilaku log audit. Untuk informasi selengkapnya, lihat Parameter konfigurasi log agen kontainer Amazon ECS.

Untuk agen kontainer versi 1.36.0 dan yang lebih baru, log audit terletak di `/var/log/ecs/audit.log`. Ketika log diputar, stempel waktu dalam format *`YYYY-MM-DD-HH`* ditambahkan ke bagian akhir nama berkas log.

Untuk agen kontainer versi 1.35.0 dan sebelumnya, log audit terletak di `/var/log/ecs/audit.log.`*`YYYY-MM-DD-HH`*.

Berikut adalah format untuk entri log:

- Stempel Waktu
- Kode tanggapan HTTP
- Alamat IP dan nomor port asal permintaan
- URI relatif penyedia kredensial
- Agen pengguna yang membuat permintaan
- ARN tugas milik kontainer yang meminta
- Nama dan nomor versi API `GetCredentials`

- Nama cluster Amazon ECS tempat instance container terdaftar
- ARN instans kontainer

Anda dapat menggunakan perintah berikut untuk melihat file log.

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

Output:

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0
 CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials
 1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

Melihat pesan acara layanan Amazon ECS

Saat memecahkan masalah dengan layanan, tempat pertama yang harus Anda periksa untuk informasi diagnostik adalah log peristiwa layanan. Anda dapat melihat peristiwa layanan menggunakan `DescribeServices` API, file AWS CLI, atau dengan menggunakan AWS Management Console.

Saat melihat pesan peristiwa layanan menggunakan Amazon ECS API, hanya peristiwa dari penjadwal layanan yang dikembalikan. Hal Ini termasuk penempatan tugas terbaru dan peristiwa kondisi instans. Namun, konsol Amazon ECS menampilkan peristiwa layanan dari sumber berikut.

- Penempatan tugas dan kejadian kesehatan instans dari penjadwal layanan Amazon ECS. Peristiwa ini memiliki awalan layanan ***(service-name)***. Untuk memastikan bahwa tampilan acara ini bermanfaat, kami hanya menampilkan acara `100` terbaru. Pesan peristiwa duplikat dihilangkan sampai penyebabnya teratasi, atau enam jam berlalu. Jika penyebabnya tidak terselesaikan dalam waktu enam jam, Anda menerima pesan acara layanan lain untuk tujuan itu.
- Peristiwa Auto Scaling layanan. Peristiwa ini memiliki awalan Pesan dan hanya terjadi ketika layanan dikonfigurasi dengan kebijakan penskalaan Application Auto Scaling.

Gunakan langkah-langkah berikut untuk melihat pesan peristiwa layanan Anda saat ini.

Console

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.

2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Clusters, pilih cluster.
4. Pilih layanan yang akan diperiksa.
5. Pilih Penerapan dan peristiwa, di bawah Acara, lihat pesan.

AWS CLI

Gunakan perintah jelaskan-layanan untuk melihat pesan peristiwa layanan untuk layanan tertentu.

AWS CLI Contoh berikut menjelaskan *service-name* layanan di *default* cluster, yang akan memberikan pesan acara layanan terbaru.

```
aws ecs describe-services \
    --cluster default \
    --services service-name \
    --region us-west-2
```

Pesan acara layanan Amazon ECS

Berikut ini adalah contoh pesan acara layanan yang mungkin Anda lihat di konsol Amazon ECS.

service (*service-name*) telah mencapai kondisi mapan.

Penjadwal layanan mengirimkan acara `service (service-name) has reached a steady state.` layanan ketika layanan sehat dan pada jumlah tugas yang diinginkan, sehingga mencapai kondisi mapan.

Penjadwal layanan melaporkan status secara berkala, sehingga Anda mungkin menerima pesan ini beberapa kali.

service (*service-name*) tidak dapat menempatkan tugas karena tidak ada instance kontainer yang memenuhi semua persyaratannya.

Penjadwal layanan mengirimkan pesan peristiwa ini ketika tidak dapat menemukan sumber daya yang tersedia untuk menambahkan tugas lain. Kemungkinan penyebab untuk ini adalah:

Gunakan penyedia kapasitas untuk menskalakan EC2 instans Anda secara otomatis. Untuk informasi selengkapnya, lihat Penyedia kapasitas Amazon ECS untuk jenis EC2 peluncuran .

Jika Anda bermaksud menggunakan penyedia kapasitas, pastikan Anda melewati strategi penyedia kapasitas atau memiliki strategi penyedia kapasitas default yang terkait dengan klaster Anda dan tidak meneruskan jenis peluncuran dan strategi penyedia kapasitas sebagai masukan

Tidak ada instans kontainer yang ditemukan di klaster Anda

Jika tidak ada instance kontainer yang terdaftar di cluster tempat Anda mencoba menjalankan tugas, Anda menerima kesalahan ini. Anda harus menambahkan instans kontainer untuk klaster Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Port tidak cukup

Jika tugas Anda menggunakan pemetaan host port tetap (misalnya, tugas Anda menggunakan port 80 pada host untuk server web), Anda harus memiliki setidaknya satu instans kontainer per tugas, karena hanya satu kontainer yang dapat menggunakan port host tunggal pada satu waktu. Anda harus menambahkan instans kontainer ke klaster milik Anda atau mengurangi jumlah tugas yang diinginkan.

Terlalu banyak port yang terdaftar

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak dapat melebihi batas port cadangan maksimum yang diizinkan sebesar 100 port host per instance kontainer. Menggunakan pemetaan port host dinamis dapat memperbaiki masalah.

Port sudah digunakan

Definisi tugas tugas ini menggunakan port yang sama dalam pemetaan portnya sebagai tugas yang sudah berjalan pada instance kontainer yang dipilih. Pesan acara layanan akan memiliki ID instance kontainer yang dipilih sebagai bagian dari pesan di bawah ini.

```
The closest matching container-instance is already using a port required by your
 task.
```

Memori tidak cukup

Jika ketentuan tugas Anda menentukan memori sejumlah 1000 MiB, dan masing-masing klaster Anda pada instans kontainer memiliki memori sejumlah 1024 MiB, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan

memori yang lebih sedikit pada ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

> **Note**
>
> Jika Anda mencoba memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori sebanyak mungkin untuk jenis instans tertentu, lihatMemesan memori instans kontainer Amazon ECS Linux.

CPU tidak cukup

Sebuah instans kontainer memiliki 1.024 unit CPU untuk setiap inti CPU. Jika ketentuan tugas Anda menentukan 1.000 unit CPU, dan setiap instans kontainer di klaster Anda memiliki 1.024 unit CPU, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan unit CPU yang lebih sedikit dalam ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

Poin lampiran ENI yang tersedia tidak cukup

Tugas yang menggunakan mode `awsvpc` jaringan masing-masing menerima elastic network interface (ENI) mereka sendiri, yang dilampirkan ke instance container yang menghostingnya. EC2 Instans Amazon memiliki batasan jumlah ENIs yang dapat dilampirkan padanya dan tidak ada instance kontainer di cluster yang memiliki kapasitas ENI yang tersedia.

Batas ENI untuk instans kontainer individu tergantung pada kondisi berikut:

- Jika Anda belum menyertakan untuk Pengaturan akun `awsvpcTrunking`, batas ENI untuk setiap instans kontainer tergantung pada tipe instans. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan EC2 Pengguna Amazon.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun tetapi Anda belum meluncurkan instance container baru menggunakan tipe instans yang didukung setelah memilih, batas ENI untuk setiap instance container masih pada nilai default. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan EC2 Pengguna Amazon.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun dan Anda telah meluncurkan instance container baru menggunakan jenis instans yang didukung setelah

memilih, tambahan ENIs akan tersedia. Untuk informasi selengkapnya, lihat Instans yang didukung untuk peningkatan antarmuka jaringan kontainer Amazon ECS.

Untuk informasi lebih lanjut tentang penyertaan pada pengaturan akun `awsvpcTrunking`, lihat Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux.

Anda dapat menambahkan instans kontainer untuk klaster Anda untuk menyediakan lebih banyak adapter jaringan yang tersedia.

Instans kontainer kehilangan atribut wajib

Beberapa parameter ketentuan tugas memerlukan versi API Docker jarak jauh tertentu untuk diinstal pada instans kontainer. Hal lainnya, seperti opsi pencatatan driver, mengharuskan instans kontainer untuk mendaftarkan pencatatan driver tersebut dengan variabel konfigurasi agen `ECS_AVAILABLE_LOGGING_DRIVERS`. Jika definisi tugas Anda berisi parameter yang memerlukan atribut instance kontainer tertentu, dan Anda tidak memiliki instance kontainer yang tersedia yang dapat memenuhi persyaratan ini, tugas tidak dapat ditempatkan.

Penyebab umum kesalahan ini adalah jika layanan Anda menggunakan tugas yang menggunakan mode `awsvpc` jaringan dan jenis EC2 peluncuran. Cluster yang Anda tentukan tidak memiliki instance kontainer yang terdaftar di subnet yang sama yang ditentukan pada `awsvpcConfiguration` saat layanan dibuat.

Anda dapat menggunakan AWSSupport-TroubleshootECSContainerInstance runbook untuk memecahkan masalah. Runbook meninjau apakah data pengguna untuk instance berisi informasi klaster yang benar, apakah profil instance berisi izin yang diperlukan, dan masalah konfigurasi jaringan. Untuk informasi selengkapnya, lihat AWSSupport-TroubleshootECSContainerInstancedi Panduan Pengguna referensi buku runbook AWS Systems Manager Otomasi.

Untuk informasi lebih lanjut tentang atribut yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate dan Konfigurasi agen kontainer Amazon ECS.

service (***`service-name`***) tidak dapat menempatkan tugas karena tidak ada instance kontainer yang memenuhi semua persyaratannya. Instance kontainer yang paling cocok tidak ***`container-instance-id`*** memiliki unit CPU yang tersedia.

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak berisi unit CPU yang cukup untuk memenuhi persyaratan dalam definisi tugas. Tinjau persyaratan CPU di kedua parameter ukuran tugas dan ketentuan kontainer dari ketentuan tugas.

service (*service-name*) tidak dapat menempatkan tugas karena tidak ada instance kontainer yang memenuhi semua persyaratannya. Instance kontainer yang paling cocok *container-instance-id* mengalami kesalahan “AGENT”.

Agen kontainer Amazon ECS pada instance kontainer pencocokan terdekat untuk penempatan tugas terputus. Jika Anda dapat terhubung ke instans kontainer dengan SSH, Anda dapat memeriksa pencatatan agen; untuk informasi lebih lanjut, lihat Parameter konfigurasi log agen kontainer Amazon ECS. Anda juga harus memverifikasi bahwa agen sedang berjalan pada instans. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat mencoba menghentikan dan memulai ulang agen dengan perintah berikut.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI dan Amazon ECS yang dioptimalkan Amazon Linux 2023 AMI

  ```
  sudo systemctl restart ecs
  ```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI

  ```
  sudo stop ecs && sudo start ecs
  ```

service (*service-name*) (task*task-id*) (instance*instance-id*) tidak sehat di (elb*elb-name*) karena (alasan Instance telah gagal setidaknya UnhealthyThreshold jumlah pemeriksaan kesehatan secara berurutan.)

Layanan ini terdaftar dengan penyeimbang beban dan pemeriksaan kondisi penyeimbang beban gagal. Pesan tersebut menyertakan ID tugas untuk membantu mengidentifikasi tugas spesifik mana yang gagal dalam pemeriksaan kesehatan. Untuk informasi selengkapnya, lihat Memecahkan masalah penyeimbang beban layanan di Amazon ECS.

service (*service-name*) tidak dapat secara konsisten memulai tugas dengan sukses.

Layanan ini berisi tugas yang gagal untuk dimulai setelah upaya dilakukan secara berturut-turut. Pada titik ini, penjadwal layanan mulai meningkatkan tambahan waktu antara pengulangan. Anda harus memecahkan masalah mengapa tugas Anda gagal untuk diluncurkan. Untuk informasi selengkapnya, lihat Logika throttle layanan Amazon ECS.

Setelah layanan diperbarui, misalnya dengan ketentuan tugas yang diperbarui, penjadwal layanan melanjutkan perilakunya secara normal.

operasi service (*service-name*) sedang dibatasi. Akan di coba lagi nanti.

Layanan ini tidak dapat meluncurkan lebih banyak tugas dikarenakan batas throttling API. Setelah penjadwal layanan dapat meluncurkan lebih banyak tugas, maka akan dilanjutkan.

Untuk meminta kenaikan tingkat batas kuota API, buka halaman AWS Dukungan Pusat, masuk jika diperlukan, dan pilih Buat kasus. Pilih Peningkatan kuota layanan. Lengkapi dan kirimkan formulir ini.

service (*service-name*) tidak dapat menghentikan atau memulai tugas selama penerapan karena konfigurasi penerapan layanan. Perbarui nilai minimumHealthyPercent atau MaximumPercent dan coba lagi.

Tugas pada layanan ini tidak dapat dihentikan atau dimulai saat layanan deployment dikarenakan konfigurasi pada deployment. Konfigurasi penyebaran terdiri dari `minimumHealthyPercent` dan `maximumPercent` nilai-nilai, yang didefinisikan ketika layanan dibuat. Nilai-nilai tersebut juga dapat diperbarui pada layanan yang ada.

`minimumHealthyPercent`Ini mewakili batas bawah pada jumlah tugas yang harus dijalankan untuk layanan selama penerapan atau ketika instance kontainer terkuras. Ini adalah persen dari jumlah tugas yang diinginkan untuk layanan ini. Nilai ini dibulatkan ke atas. Misalnya, jika persentase sehat minimum adalah `50` dan jumlah tugas yang diinginkan adalah empat, maka penjadwal dapat menghentikan dua tugas yang ada sebelum memulai dua tugas baru. Demikian juga, jika persentase minimum yang sehat adalah 75% dan jumlah tugas yang diinginkan adalah dua, maka penjadwal tidak dapat menghentikan tugas apa pun karena nilai yang dihasilkan juga dua.

`maximumPercent`Ini mewakili batas atas jumlah tugas yang harus dijalankan untuk layanan selama penerapan atau saat instance kontainer terkuras. Ini adalah persentase dari jumlah tugas yang diinginkan untuk suatu layanan. Nilai ini dibulatkan ke bawah. Misalnya, jika persentase maksimum adalah `200` dan jumlah tugas yang diinginkan adalah empat, maka penjadwal dapat memulai empat tugas baru sebelum menghentikan empat tugas yang ada. Demikian juga, jika persentase maksimum adalah `125` dan jumlah tugas yang diinginkan adalah tiga, maka penjadwal tidak dapat memulai tugas apa pun karena nilai yang dihasilkan juga tiga.

Saat pengaturan persentase minimum atau maksimum yang sehat, Anda harus memastikan bahwa penjadwal dapat menghentikan atau memulai setidaknya satu tugas saat deployment dipicu.

service (*service-name*) tidak dapat menempatkan tugas. Alasan: Anda telah mencapai batas jumlah tugas yang dapat Anda jalankan secara bersamaan

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

service (*service-name*) tidak dapat menempatkan tugas. Alasan: Kesalahan internal.

Berikut ini adalah kemungkinan alasan untuk kesalahan ini:

Layanan tidak dapat memulai tugas karena subnet berada di Availability Zone yang tidak didukung.

Untuk informasi tentang Wilayah Fargate dan Zona Ketersediaan yang didukung, lihat. the section called “AWS Wilayah Fargate”

Untuk informasi tentang cara melihat Zona Ketersediaan subnet, lihat Melihat subnet Anda di Panduan Pengguna Amazon VPC.

service (*service-name*) tidak dapat menempatkan tugas. Alasan: Konfigurasi CPU yang diminta berada di atas batas Anda.

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

service (*service-name*) tidak dapat menempatkan tugas. Alasan: Konfigurasi MEMORY yang diminta berada di atas batas Anda.

Anda dapat meminta peningkatan kuota untuk sumber daya yang menyebabkan kesalahan. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

service (*service-name*) tidak dapat menempatkan tugas. Alasan: Anda telah mencapai batas jumlah v yang dapat CPUs Anda jalankan secara bersamaan

AWS Fargate beralih dari kuota berbasis hitungan tugas ke kuota berbasis VCPU.

Anda dapat meminta kenaikan kuota untuk kuota berbasis VCPU Fargate. Untuk informasi selengkapnya, lihat Kuota layanan. Untuk meminta peningkatan kuota Fargate, lihat Meminta kenaikan kuota pada Panduan Pengguna Service Quotas.

service (*service-name*) tidak dapat mencapai kondisi tunak karena task set (*taskSet-ID*) tidak dapat diskalakan. Alasan: Jumlah tugas yang dilindungi lebih dari jumlah tugas yang diinginkan

Layanan ini memiliki tugas yang lebih terlindungi daripada yang diinginkanjumlah tugas. Anda dapat melakukan salah satu hal berikut:

- Tunggu hingga perlindungan pada tugas saat ini kedaluwarsa, memungkinkan mereka untuk dihentikan.
- Tentukan tugas mana yang dapat dihentikan dan gunakan `UpdateTaskProtection` API dengan `protectionEnabled` opsi yang disetel `false` ke unset protection untuk tugas-tugas ini.
- Tingkatkan jumlah tugas yang diinginkan dari layanan menjadi lebih dari jumlah tugas yang dilindungi.

service (*service-name*) tidak dapat mencapai kondisi mapan. Alasan: Tidak ada Instans Kontainer yang ditemukan di penyedia kapasitas Anda.

Penjadwal layanan mengirimkan pesan peristiwa ini ketika tidak dapat menemukan sumber daya yang tersedia untuk menambahkan tugas lain. Kemungkinan penyebab untuk ini adalah:

Tidak ada penyedia kapasitas yang terkait dengan cluster

Gunakan `describe-services` untuk memverifikasi bahwa Anda memiliki penyedia kapasitas yang terkait dengan klaster Anda dapat memperbarui strategi penyedia kapasitas untuk layanan tersebut.

Verifikasi bahwa ada kapasitas yang tersedia di penyedia kapasitas, Dalam kasus jenis EC2 peluncuran, pastikan bahwa instance kontainer memenuhi persyaratan definisi tugas.

Tidak ada instans kontainer yang ditemukan di klaster Anda

Jika tidak ada instance kontainer yang terdaftar di cluster tempat Anda mencoba menjalankan tugas, Anda menerima kesalahan ini. Anda harus menambahkan instans kontainer untuk klaster Anda. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.

Port tidak cukup

Jika tugas Anda menggunakan pemetaan port host tetap (misalnya, tugas Anda menggunakan port 80 pada host untuk server web), Anda harus memiliki setidaknya satu instance kontainer per tugas. Hanya satu kontainer yang dapat menggunakan port host tunggal pada satu waktu. Anda

harus menambahkan instans kontainer ke klaster milik Anda atau mengurangi jumlah tugas yang diinginkan.

Terlalu banyak port yang terdaftar

Instance pencocokan kontainer terdekat untuk penempatan tugas tidak dapat melebihi batas port cadangan maksimum yang diizinkan sebesar 100 port host per instance kontainer. Menggunakan pemetaan port host dinamis dapat memperbaiki masalah.

Port sudah digunakan

Definisi tugas tugas ini menggunakan port yang sama dalam pemetaan portnya sebagai tugas yang sudah berjalan pada instance kontainer yang dipilih. Pesan acara layanan akan memiliki ID instance kontainer yang dipilih sebagai bagian dari pesan di bawah ini.

```
The closest matching container-instance is already using a port required by your
 task.
```

Memori tidak cukup

Jika ketentuan tugas Anda menentukan memori sejumlah 1000 MiB, dan masing-masing klaster Anda pada instans kontainer memiliki memori sejumlah 1024 MiB, Anda hanya dapat menjalankan satu salinan tugas ini per instans kontainer. Anda dapat bereksperimen dengan memori yang lebih sedikit pada ketentuan tugas Anda sehingga Anda bisa meluncurkan lebih dari satu tugas per instans kontainer, atau meluncurkan lebih banyak instans kontainer menuju klaster Anda.

> **Note**
>
> Jika Anda mencoba untuk memaksimalkan pemanfaatan sumber daya Anda dengan menyediakan memori pada tugas sebanyak mungkin untuk tipe instans tertentu, lihat Memesan memori instans kontainer Amazon ECS Linux.

Poin lampiran ENI yang tersedia tidak cukup

Tugas yang menggunakan mode `awsvpc` jaringan masing-masing menerima elastic network interface (ENI) mereka sendiri, yang dilampirkan ke instance container yang menghostingnya. EC2 Instans Amazon memiliki batasan jumlah ENIs yang dapat dilampirkan padanya, dan tidak ada instance kontainer di cluster yang memiliki kapasitas ENI yang tersedia.

Batas ENI untuk instans kontainer individu tergantung pada kondisi berikut:

- Jika Anda belum menyertakan untuk Pengaturan akun `awsvpcTrunking`, batas ENI untuk setiap instans kontainer tergantung pada tipe instans. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan EC2 Pengguna Amazon.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun tetapi Anda belum meluncurkan instance container baru menggunakan tipe instans yang didukung setelah memilih, batas ENI untuk setiap instance container masih pada nilai default. Untuk informasi selengkapnya, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan EC2 Pengguna Amazon.
- Jika Anda telah memilih untuk masuk ke setelan `awsvpcTrunking` akun dan Anda telah meluncurkan instance container baru menggunakan jenis instans yang didukung setelah memilih, tambahan ENIs akan tersedia. Untuk informasi selengkapnya, lihat Instans yang didukung untuk peningkatan antarmuka jaringan kontainer Amazon ECS.

Untuk informasi lebih lanjut tentang penyertaan pada pengaturan akun `awsvpcTrunking`, lihat Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux.

Anda dapat menambahkan instans kontainer untuk klaster Anda untuk menyediakan lebih banyak adapter jaringan yang tersedia.

Instans kontainer kehilangan atribut wajib

Beberapa parameter ketentuan tugas memerlukan versi API Docker jarak jauh tertentu untuk diinstal pada instans kontainer. Hal lainnya, seperti opsi pencatatan driver, mengharuskan instans kontainer untuk mendaftarkan pencatatan driver tersebut dengan variabel konfigurasi agen `ECS_AVAILABLE_LOGGING_DRIVERS`. Jika definisi tugas Anda berisi parameter yang memerlukan atribut instance kontainer tertentu, dan Anda tidak memiliki instance kontainer yang tersedia yang dapat memenuhi persyaratan ini, tugas tidak dapat ditempatkan.

Penyebab umum kesalahan ini adalah jika layanan Anda menggunakan tugas yang menggunakan mode `awsvpc` jaringan dan jenis EC2 peluncuran dan kluster yang Anda tentukan tidak memiliki instance kontainer yang terdaftar di subnet yang sama yang ditentukan pada `awsvpcConfiguration` saat layanan dibuat.

Anda dapat menggunakan AWSSupport-TroubleshootECSContainerInstance runbook untuk memecahkan masalah. Runbook meninjau apakah data pengguna untuk instance berisi informasi klaster yang benar, apakah profil instance berisi izin yang diperlukan, dan masalah konfigurasi jaringan. Untuk informasi selengkapnya, lihat AWSSupport-TroubleshootECSContainerInstancedi Panduan Pengguna referensi buku runbook AWS Systems Manager Otomasi.

Untuk informasi lebih lanjut tentang atribut yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate dan Konfigurasi agen kontainer Amazon ECS.

service (***service-name***) tidak dapat menempatkan tugas. Alasan: Kapasitas tidak tersedia saat ini. Silakan coba lagi nanti atau di zona ketersediaan yang berbeda.

Saat ini tidak ada kapasitas yang tersedia untuk menjalankan layanan Anda.

Anda dapat melakukan salah satu hal berikut:

- Tunggu hingga kapasitas Fargate atau instance EC2 kontainer tersedia.
- Luncurkan kembali layanan dan tentukan subnet tambahan.

service (***service-name***) penerapan gagal: tugas gagal dimulai.

Tugas dalam layanan Anda gagal dimulai.

Untuk informasi tentang cara men-debug tugas yang dihentikan. lihat. Amazon ECS menghentikan pesan kesalahan tugas

service (***service-name***) Waktu habis menunggu Amazon ECS Agent dimulai. Silakan periksa log di/var/log/ecs/ecs-agent.log".

Agen kontainer Amazon ECS pada instance kontainer pencocokan terdekat untuk penempatan tugas terputus. Jika Anda dapat terhubung ke instance kontainer dengan SSH, Anda dapat memeriksa log agen. Untuk informasi selengkapnya, lihat Parameter konfigurasi log agen kontainer Amazon ECS. Anda juga harus memverifikasi bahwa agen sedang berjalan pada instans. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat mencoba menghentikan dan memulai ulang agen dengan perintah berikut.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

service (*service-name*) task set (*taskSet-ID*) (task*task-id*) tidak sehat di target-group (*targetGroup-ARN)*) karena. **TARGET GROUP IS NOT FOUND**

Tugas yang ditetapkan untuk layanan ini gagal dalam pemeriksaan kesehatan karena kelompok sasaran tidak ditemukan. Pesan tersebut menyertakan ID tugas untuk membantu mengidentifikasi tugas spesifik mana yang gagal dalam pemeriksaan kesehatan. Anda harus menghapus dan membuat ulang layanan. Jangan menghapus grup target Elastic Load Balancing kecuali layanan Amazon ECS yang sesuai telah dihapus.

service (*service-name*) task set (*taskSet-ID*) (task*task-id*) tidak sehat di target-group (*targetGroup-ARN)*) karena. **TARGET IS NOT FOUND**

Tugas yang ditetapkan untuk layanan ini gagal dalam pemeriksaan kesehatan karena target tidak ditemukan. Pesan tersebut menyertakan ID tugas untuk membantu mengidentifikasi tugas spesifik mana yang gagal dalam pemeriksaan kesehatan.

Layanan Amazon ECS pesan acara tidak sehat

Berikut ini adalah contoh layanan pesan acara yang tidak sehat.

EC2: (service*service-name*) (task*task-id*) (instance*instance-id*) (port*port-number*) tidak sehat di (target-group*target-group-name*) karena (alasan) *failure-reason*

Pesan ini menunjukkan bahwa tugas yang berjalan pada sebuah EC2 instance gagal dalam pemeriksaan kesehatan. Untuk informasi selengkapnya, lihat berikut ini:

- Bagaimana cara mendapatkan tugas Amazon ECS saya yang menggunakan tipe EC2 peluncuran Amazon untuk lulus pemeriksaan kesehatan Application Load Balancer?

EC2 dengan TaskSet: (service*service-name*, TaskSet*taskSet-id*) (task*task-id*) (instance) (port *instance-idport-number*) tidak sehat di (target-group*target-group-name*) karena (alasan) *failure-reason*

Pesan ini menunjukkan bahwa tugas dalam set tugas yang berjalan pada sebuah EC2 instance gagal dalam pemeriksaan kesehatan. Untuk informasi selengkapnya, lihat berikut ini:

- Bagaimana cara mendapatkan tugas Amazon ECS saya yang menggunakan tipe EC2 peluncuran Amazon untuk lulus pemeriksaan kesehatan Application Load Balancer?

Fargate: (layanan*service-name*) (tugas*task-id*) (port*port-number*) tidak sehat di (kelompok target*target-group-name*) karena (alasan) *failure-reason*

Pesan ini menunjukkan bahwa tugas Fargate gagal dalam pemeriksaan kesehatan.

Untuk informasi selengkapnya tentang pemecahan masalah kegagalan pemeriksaan kesehatan di tugas Fargate, lihat Bagaimana cara memecahkan masalah kegagalan pemeriksaan kesehatan untuk tugas Amazon ECS di Fargate? .

Fargate dengan TaskSet: (service*service-name*, TaskSet*taskSet-id*) (task) (port*task-id*) tidak sehat di (*target-group-name*target-group*port-number*) karena (alasan) *failure-reason*

Pesan ini menunjukkan bahwa tugas dalam set tugas yang berjalan di Fargate gagal dalam pemeriksaan kesehatan.

Untuk informasi selengkapnya tentang pemecahan masalah kegagalan pemeriksaan kesehatan di tugas Fargate, lihat Bagaimana cara memecahkan masalah kegagalan pemeriksaan kesehatan untuk tugas Amazon ECS di Fargate? .

Amazon ECS Availability Zone layanan penyeimbangan kembali pesan acara layanan

Berikut ini adalah contoh pesan acara layanan yang mungkin Anda lihat.

service (*service-name*) tidak AZ seimbang dengan *number-tasks* tugas di dalam*Availability Zone 1*, *number-tasks* di*Availability Zone 2*, dan *number-tasks* di*Availability Zone 3*. AZ Rebalancing sedang berlangsung.

Penjadwal layanan mengirimkan acara `service (service-name) is not AZ balanced` layanan ketika jumlah tugas tidak tersebar merata di seluruh Availability Zone. Tidak ada tindakan yang harus diambil. Ini adalah acara informasi.

service (*service-name*) adalah AZ seimbang dengan *number-tasks* tugas di*Availability Zone 1*, *number-tasks* tugas di*Availability Zone 2*, dan *number-tasks* tugas di*Availability Zone 3*.

Penjadwal layanan mengirimkan peristiwa `service (service-name) is AZ balanced` layanan saat penyeimbangan kembali layanan Availability Zone selesai. Tidak ada tindakan yang harus diambil. Ini adalah acara informasi.

*service-name*telah memulai *number-tasks* tugas *Availability Zone* ke AZ Rebalance:*task-ids*.

Penjadwal layanan mengirimkan *service-name task-set-name* /telah memulai *number* tugas dalam acara *Availability Zone* layanan ketika memulai tugas di Availability Zone karena penyeimbangan kembali layanan. Tidak ada tindakan yang harus diambil. Ini adalah acara informasi.

*service-name*telah berhenti *number-tasks* menjalankan tugas *Availability Zone* karena penyeimbangan kembali AZ:. *task-id*

Penjadwal layanan mengirimkan *service-name task-set-name* /telah menghentikan *number* tugas dalam acara *Availability Zone* layanan ketika menghentikan tugas di Availability Zone karena penyeimbangan kembali layanan. Tidak ada tindakan yang harus diambil. Ini adalah acara informasi.

service (*service-name*) tidak dapat menempatkan tugas *Availability Zone* karena tidak ada instance kontainer yang memenuhi semua persyaratannya.

Penjadwal layanan mengirimkan tidak dapat menempatkan tugas dalam acara *Availability Zone* layanan karena tidak ada instance kontainer yang memenuhi semua persyaratannya. *service-name* Untuk mengatasi masalah ini, luncurkan instance di Availability Zone.

service (*service-name*) tidak dapat menempatkan tugas di*Availability Zone*.

Penjadwal layanan mengirimkan *service-name* tidak dapat menempatkan tugas dalam acara *Availability Zone* layanan saat Anda menggunakan jenis peluncuran Fargate dan tidak ada kapasitas yang tersedia.

Anda dapat menambahkan subnet tambahan di Availability Zone dalam pesan kesalahan, atau kontak Dukungan untuk mendapatkan kapasitas tambahan.

service (*service-name*) tidak dapat melakukan AZ Rebalance karena *task-set-name* tidak dapat menskalakan karena*reason*.

Penjadwal layanan mengirimkan tidak dapat melakukan AZ Rebalance karena *task-set-name* tidak dapat menskalakan karena peristiwa *reason* layanan saat Anda menggunakan perlindungan skala tugas. *service-name*

Anda dapat melakukan salah satu hal berikut:

- Tunggu hingga perlindungan pada tugas saat ini kedaluwarsa, memungkinkan mereka untuk dihentikan.
- Tentukan tugas mana yang dapat dihentikan dan gunakan `UpdateTaskProtection` API dengan `protectionEnabled` opsi yang disetel `false` ke unset protection untuk tugas-tugas ini.
- Tingkatkan jumlah tugas yang diinginkan dari layanan menjadi lebih dari jumlah tugas yang dilindungi.

service (*service-name*) menghentikan AZ Rebalancing.

Penjadwal layanan mengirimkan peristiwa layanan Rebalancing AZ yang *service-name* dihentikan saat operasi penyeimbangan kembali Availability Zone berhenti. Ini adalah acara informasi. Amazon ECS mengirimkan acara tambahan yang memberikan informasi lebih lanjut.

Memecahkan masalah penyeimbang beban layanan di Amazon ECS

Layanan Amazon ECS dapat mendaftarkan tugas dengan penyeimbang beban Elastic Load Balancing. Kesalahan konfigurasi terhadap penyeimbang beban adalah penyebab umum kenapa tugas berhenti. Jika tugas Anda yang berhenti dimulai oleh layanan yang menggunakan penyeimbang beban, pertimbangkan kemungkinan penyebab berikut ini.

Peran terkait layanan Amazon ECS tidak ada

> Peran terkait layanan Amazon ECS memungkinkan layanan Amazon ECS mendaftarkan instans kontainer dengan penyeimbang beban Elastic Load Balancing. Peran yang terkait dengan layanan harus dibuat di akun Anda. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

Grup keamanan contoh kontainer

Jika kontainer Anda dipetakan kepada port 80 di instans kontainer Anda, grup keamanan instans kontainer harus mengizinkan lalu lintas masuk pada port 80 untuk pemeriksaan kondisi penyeimbang beban agar dapat melalui port tersebut.

Penyeimbang beban Elastic Load Balancing tidak dikonfigurasi untuk semua Availability Zone

Penyeimbang beban Anda harus dikonfigurasikan untuk dapat menggunakan semua Availability Zone pada sebuah Wilayah, atau setidaknya semua Availability Zone tempat instans kontainer berada. Jika layanan menggunakan penyeimbang beban dan memulai tugas pada instance container yang berada di Availability Zone yang tidak dikonfigurasi untuk digunakan oleh penyeimbang beban, tugas tersebut tidak akan pernah lolos pemeriksaan kesehatan. Ini mengakibatkan tugas terbunuh.

Pemeriksaan kesehatan penyeimbang beban Elastic Load Balancing salah dikonfigurasi

Parameter pemeriksaan kondisi penyeimbang beban dapat terlalu ketat atau mengarah pada sumber daya yang sudah tidak ada. Jika instance kontainer ditentukan tidak sehat, itu dihapus dari penyeimbang beban. Pastikan untuk memverifikasi bahwa parameter berikut ini dikonfigurasi dengan benar untuk layanan penyeimbang beban Anda.

Port Ping

Parameter nilai Port Ping untuk pemeriksaan kondisi penyeimbang beban adalah port pada instans kontainer bahwa pemeriksaan pada penyeimbang beban bertujuan menentukan apakah itu dalam kondisi yang optimal. Jika port ini salah dikonfigurasikan, maka penyeimbang beban memilik kemungkinan untuk mencabut kembali register instans kontainer Anda dari itu sendiri. Port ini perlu dikonfigurasikan untuk dapat menggunakan nilai `hostPort` untuk kontainer dalam definisi layanan tugas yang Anda gunakan dengan pemeriksaan kondisi.

Jalur Ping

Ini adalah bagian dari pemeriksaan kesehatan penyeimbang beban. Ini adalah titik akhir pada aplikasi Anda yang dapat memulai kembali kode status yang berhasil (misalnya, 200) ketika aplikasi sehat. Nilai ini sering diatur ke `index.html`, tetapi jika layanan Anda tidak menanggapi permintaan tersebut, maka pemeriksaan kondisi gagal. Jika kontainer Anda tidak memiliki sebuah file `index.html`, Anda dapat mengaturnya menuju `/` untuk menargetkan dasar URL untuk instans kontainer.

Waktu Respons Habis

Ini adalah jumlah waktu yang kontainer Anda harus dapat kembalikan sebuah respons terhadap ping pemeriksaan kondisi. Jika nilai ini lebih rendah dari jumlah waktu yang diperlukan untuk respon, pemeriksaan kondisi gagal.

Interval Pemeriksaan Kondisi

Ini adalah jumlah waktu antara ping pemeriksaan kondisi. Semakin pendek interval pemeriksaan kondisi Anda, semakin cepat instans kontainer Anda dapat mencapai Ambang Tidak Baik.

Batas Kondisi Tidak Baik

Ini merupakan jumlah berapa kali pemeriksaan kondisi Anda bisa gagal sebelum instans kontainer Anda dianggap tidak optimal. Jika Anda memiliki ambang batas 2 yang tidak sehat, dan interval pemeriksaan kesehatan 30 detik, maka tugas Anda memiliki 60 detik untuk menanggapi ping pemeriksaan kesehatan sebelum dianggap tidak sehat. Anda dapat menaikkan ambang batas yang tidak optimal atau interval pemeriksaan kondisi untuk memberikan tugas Anda lebih banyak waktu untuk merespons.

Tidak dapat memperbarui layanan ***`servicename`***: Nama wadah penyeimbang beban atau port diubah dalam definisi tugas

Jika layanan Anda menggunakan penyeimbang beban, Anda dapat menggunakan AWS CLI atau SDK untuk memodifikasi konfigurasi penyeimbang beban. Untuk informasi tentang cara mengubah konfigurasi, lihat UpdateServicedi Referensi API Amazon Elastic Container Service. Jika Anda memperbarui definisi tugas untuk layanan, nama kontainer dan port kontainer yang ditentukan dalam konfigurasi penyeimbang beban harus tetap dalam definisi tugas.

Anda telah mencapai batas jumlah tugas yang dapat Anda jalankan secara bersamaan.

Untuk akun baru, kuota Anda mungkin lebih rendah dari kuota layanan. Kuota layanan untuk akun Anda dapat dilihat di konsol Service Quotas. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

Pemecahan masalah penskalaan otomatis layanan di Amazon ECS

Application Auto Scaling mematikan proses scale-in saat penerapan Amazon ECS sedang berlangsung, dan proses tersebut dilanjutkan setelah penerapan selesai. Namun, proses penskalaan keluar terus terjadi, kecuali ditangguhkan, selama deployment. Untuk informasi lebih lanjut, lihat Menangguhkan dan melanjutkan penskalaan untuk Application Auto Scaling.

Memecahkan masalah kesalahan kesalahan CPU atau memori Amazon ECS definisi tugas yang tidak valid

Saat mendaftarkan definisi tugas menggunakan Amazon ECS API atau AWS CLI, jika Anda menentukan `memory` nilai `cpu` atau tidak valid, kesalahan berikut akan ditampilkan.

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task.
```

> **Note**
>
> Saat menggunakan Terraform, kesalahan berikut mungkin dikembalikan.
>
> ```
> Error: ClientException: No Fargate configuration exists for given values.
> ```

Untuk mengatasi masalah ini, Anda harus menentukan nilai yang didukung untuk tugas CPU serta memori dalam ketentuan tugas Anda. `cpu`Nilai dapat dinyatakan dalam unit CPU atau v CPUs dalam definisi tugas. Ini dikonversi ke bilangan bulat yang menunjukkan unit CPU ketika definisi tugas terdaftar. `memory`Nilai dapat dinyatakan dalam MiB atau GB dalam definisi tugas. Ini dikonversi ke bilangan bulat yang menunjukkan MiB ketika definisi tugas terdaftar.

Untuk definisi tugas yang menentukan `FARGATE` `requiresCompatibilities` parameter (bahkan jika juga `EC2` ditentukan), Anda harus menggunakan salah satu nilai dalam tabel berikut. Nilai-nilai ini menentukan rentang nilai yang didukung untuk CPU dan parameter memori.

Untuk tugas yang dihosting di Fargate, tabel berikut menunjukkan kombinasi CPU dan memori yang valid. Nilai memori dalam file JSON ditentukan dalam MiB. Anda dapat mengonversi nilai GB ke MiB dengan mengalikan nilainya dengan 1024. Misalnya 1 GB = 1024 MiB.

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux

Nilai CPU	Nilai memori	Sistem operasi yang didukung untuk AWS Fargate
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Antara 4 GB dan 16 GB dalam peningkatan 1 GB	Linux, Windows
4096 (4 vCPU)	Antara 8 GB dan 30 GB dalam peningkatan 1 GB	Linux, Windows
8192 (8 vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 16 GB dan 60 GB dalam peningkatan 4 GB	Linux
16384 (16vCPU) **Note** Opsi ini membutuhkan platform Linux `1.4.0` atau yang lebih baru.	Antara 32 GB dan 120 GB dalam peningkatan 8 GB	Linux

Untuk tugas yang dihosting di Amazon EC2, nilai CPU tugas yang didukung adalah antara 0,25 v CPUs dan 192 v. CPUs

Mekanisme kontrol CPU berbeda antara jenis peluncuran Amazon EC2 dan Fargate:

- Untuk tugas yang dihosting di Amazon EC2: Amazon ECS menggunakan periode CPU dan kuota CPU untuk mengontrol batas keras CPU ukuran tugas. Saat Anda menentukan vCPU dalam definisi tugas, Amazon ECS menerjemahkan nilai ke periode CPU dan setelan kuota CPU yang berlaku untuk. `cgroup`

- Untuk tugas yang dihosting di Fargate: Amazon ECS menggunakan pembagian CPU untuk mengontrol alokasi CPU. Kuota CPU dan nilai periode tidak digunakan untuk membatasi CPU dalam tugas Fargate.

Untuk EC2 tugas Amazon, kuota CPU mengontrol jumlah waktu CPU yang diberikan kepada a `cgroup` selama periode CPU tertentu. Kedua pengaturan dinyatakan dalam mikrodetik. Ketika kuota CPU sama dengan periode CPU berarti `cgroup` dapat mengeksekusi hingga 100% pada satu vCPU (atau fraksi lain yang totalnya 100% untuk beberapa v). CPUs Kuota CPU memiliki maksimum 1000000us dan periode CPU minimal 1ms. Anda dapat menggunakan nilai-nilai ini untuk menetapkan batas jumlah CPU Anda. Ketika Anda mengubah periode CPU tanpa mengubah kuota CPU, Anda memiliki batas efektif yang berbeda dari yang telah Anda tentukan dalam definisi tugas Anda.

Periode 100ms memungkinkan v CPUs mulai dari 0,125 hingga 10.

> **Note**
>
> Tingkat tugas CPU dan memori parameter diabaikan untuk Windows kontainer.

Melihat log agen kontainer Amazon ECS

Amazon ECS menyimpan log di `/var/log/ecs` folder instans kontainer Anda. Ada log yang tersedia dari agen penampung Amazon ECS dan dari `ecs-init` layanan yang mengontrol status agen (start/stop) pada instance container. Anda dapat melihat berkas log ini dengan menghubungkan ke instans kontainer menggunakan SSH.

> **Note**
>
> Jika Anda tidak yakin cara mengumpulkan semua log pada instans kontainer Anda, Anda dapat menggunakan kolektor log Amazon ECS. Untuk informasi selengkapnya, lihat Mengumpulkan log kontainer dengan kolektor log Amazon ECS .

Sistem operasi Linux

Proses `ecs-init` menyimpan log di `/var/log/ecs/ecs-init.log`.

`ecs-init.log`File tersebut berisi informasi tentang manajemen siklus hidup agen kontainer, konfigurasi, dan bootstrap.

Anda dapat menggunakan perintah berikut untuk melihat file log.

```
cat /var/log/ecs/ecs-init.log
```

Output:

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

Sistem operasi Windows

Anda dapat menggunakan kolektor log Amazon ECS untuk Windows. Untuk informasi selengkapnya, lihat Amazon ECS Logs Collector Untuk Windows di Github.

1. Terhubung ke instans Anda.
2. Buka PowerShell dan kemudian jalankan perintah berikut dengan hak administratif. Perintah mengunduh skrip dan mengumpulkan log.

   ```
   Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
   raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
   logs-collector.ps1
   .\ecs-logs-collector.ps1
   ```

Anda dapat mengaktifkan logging debug untuk agen Amazon ECS dan daemon Docker. Opsi ini memungkinkan skrip untuk mengumpulkan log sebelum mengaktifkan mode debug. Skrip memulai ulang daemon Docker dan agen Amazon ECS, dan kemudian menghentikan semua kontainer yang berjalan pada instance. Sebelum menjalankan perintah berikut, tiriskan instance kontainer dan pindahkan tugas penting apa pun ke instance kontainer lainnya.

Jalankan perintah berikut untuk mengaktifkan logging.

```
.\ecs-logs-collector.ps1 -RunMode debug
```

Mengumpulkan log kontainer dengan kolektor log Amazon ECS

Jika Anda tidak yakin cara mengumpulkan semua log yang berbeda pada instans kontainer Anda, Anda dapat menggunakan kolektor log Amazon ECS. Ini tersedia GitHub untuk Linux dan Windows. Skrip mengumpulkan log sistem operasi umum serta log agen kontainer Docker dan Amazon ECS, yang dapat membantu untuk memecahkan masalah kasus. AWS Dukungan Kemudian informasi yang dikumpulkan dikompres dan diarsipkan ke dalam satu file yang dapat dengan mudah dibagikan untuk tujuan diagnostik. Ini juga mendukung mengaktifkan mode debug untuk daemon Docker dan agen kontainer Amazon ECS pada varian Amazon Linux, seperti AMI Amazon ECS yang dioptimalkan. Saat ini, kolektor log Amazon ECS mendukung sistem operasi berikut:

- Amazon Linux
- Red Hat Enterprise Linux 7
- Debian 8
- Ubuntu 14.04
- Ubuntu 16.04
- Ubuntu 18.04
- Windows Server 2016

> **Note**
>
> Kode sumber untuk kolektor log Amazon ECS tersedia GitHub untuk Linux dan Windows. Kami menyarankan Anda agar mengirimkan permintaan pull untuk perubahan yang ingin Anda sertakan. Namun, Amazon Web Services saat ini tidak mendukung menjalankan salinan perangkat lunak yang diubah ini.

Untuk mengunduh dan menjalankan kolektor log Amazon ECS untuk Linux

1. Hubungkan menuju instans kontainer Anda.
2. Unduh skrip kolektor log Amazon ECS.

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. Jalankan penulisan untuk mengumpulkan catatan dan membuat arsip.

> **Note**
>
> Untuk mengaktifkan mode debug untuk daemon Docker dan agen penampung Amazon ECS, tambahkan `--mode=enable-debug` opsi ke perintah berikut. Ini mungkin memulai ulang daemon Docker, yang membunuh semua kontainer yang berjalan pada instance. Pertimbangkan pengurasan pada instans kontainer dan pemindahan tugas-tugas penting untuk instans kontainer lainnya sebelum mengaktifkan mode debug. Untuk informasi selengkapnya, lihat Menguras instans kontainer Amazon ECS.

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

> **Important**
>
> Kami menyarankan Anda mengedit log dan menghapus semua data sensitif dari file. Anda dapat mencari data yang dikenal, dan juga mencari variabel lingkungan sepertiAWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY, dan AWS_SESSION_TOKEN dalam file.

Setelah Anda menjalankan penulisannya, Anda dapat memeriksa catatan yang dikumpulkan di folder `collect` yang dibuat oleh penulisan. `collect.tgz`File ini adalah arsip terkompresi dari semua log, yang dapat Anda bagikan AWS Dukungan untuk bantuan diagnostik.

Untuk mengunduh dan menjalankan kolektor log Amazon ECS untuk Windows

1. Hubungkan menuju instans kontainer Anda. Untuk informasi selengkapnya, lihat Connect ke instans Windows menggunakan RDP di EC2 Panduan Pengguna Amazon.
2. Unduh skrip kolektor log Amazon ECS menggunakan PowerShell.

   ```
   Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
   raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
   logs-collector.ps1
   ```
3. Jalankan penulisan untuk mengumpulkan catatan dan membuat arsip.

> **Note**
>
> Untuk mengaktifkan mode debug untuk daemon Docker dan agen penampung Amazon ECS, tambahkan `-RunMode debug` opsi ke perintah berikut. Ini akan memulai kembali daemon Docker, yang menghancurkan semua kontainer yang berjalan pada instans. Pertimbangkan pengurasan instans kontainer dan pemindahan tugas-tugas penting untuk instans kontainer lainnya sebelum mengaktifkan mode debug. Untuk informasi selengkapnya, lihat Menguras instans kontainer Amazon ECS.

```
.\ecs-logs-collector.ps1
```

> **Important**
>
> Kami menyarankan Anda mengedit log dan menghapus semua data sensitif dari file. Anda dapat mencari data yang dikenal, dan juga mencari variabel lingkungan seperti`AWS_ACCESS_KEY_ID`,`AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` dalam file.

Setelah Anda menjalankan penulisannya, Anda dapat memeriksa catatan yang dikumpulkan di folder `collect` yang dibuat oleh penulisan. `collect.tgz`File ini adalah arsip terkompresi dari semua log, yang dapat Anda bagikan dengan AWS Support for diagnostic help.

Ambil detail diagnostik Amazon ECS dengan introspeksi agen

API introspeksi agen Amazon ECS memberikan informasi tentang keadaan keseluruhan agen Amazon ECS dan instans kontainer.

Anda dapat menggunakan API introspeksi agen untuk mendapatkan ID Docker untuk wadah dalam tugas Anda. Anda dapat menggunakan API introspeksi agen dengan menghubungkan ke instans kontainer menggunakan SSH.

Important

Instance container Anda harus memiliki peran IAM yang memungkinkan akses ke Amazon ECS untuk mencapai API introspeksi. Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.

Contoh berikut menunjukkan dua tugas, satu yang sedang berjalan dan satu yang dihentikan.

Note

Perintah berikut disalurkan melalui python -mjson.tool untuk keterbacaan yang lebih besar.

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

Output:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1095  100  1095    0     0   117k      0 --:--:-- --:--:-- --:--:--  133k
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-
a8d14064fd24",
            "Containers": [
                {
                    "DockerId":
 "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
                    "DockerName": "ecs-console-sample-app-static-6-simple-
app-86caf9bcabe3e9c61600",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
                    "DockerName": "ecs-console-sample-app-static-6-busybox-
ce83ce978a87a890ab01",
                    "Name": "busybox"
```

```
                }
            ],
            "Family": "console-sample-app-static",
            "KnownStatus": "STOPPED",
            "Version": "6"
        },
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
            "Containers": [
                {
                    "DockerId":
 "dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
                    "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
                    "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
                    "Name": "busybox"
                }
            ],
            "DesiredStatus": "RUNNING",
            "Family": "console-sample-app-static",
            "KnownStatus": "RUNNING",
            "Version": "6"
        }
    ]
}
```

Pada contoh sebelumnya, tugas yang dihentikan (*090eff9b-1ce3-4db6-848a-a8d14064fd24*) memiliki dua kontainer. Anda dapat menggunakan docker inspect ***container-ID*** untuk melihat informasi detail pada setiap kontainer. Untuk informasi selengkapnya, lihat Introspeksi wadah Amazon ECS.

Diagnostik Docker di Amazon ECS

Docker menyediakan beberapa alat diagnostik yang membantu Anda memecahkan masalah dengan kontainer dan tugas Anda. Untuk informasi selengkapnya tentang semua utilitas baris perintah

Docker yang tersedia, lihat referensi CLI Docker dalam dokumentasi Docker. Anda dapat mengakses utilitas baris perintah Docker dengan menghubungkan ke instans kontainer menggunakan SSH.

Kode keluar yang Docker kontainer laporkan juga dapat memberikan beberapa informasi diagnostik (misalnya, kode keluar 137 berarti bahwa kontainer menerima sinyal `SIGKILL`). Untuk informasi selengkapnya, lihat Status Keluar dalam dokumentasi Docker.

Daftar kontainer Docker di Amazon ECS

Anda dapat menggunakan perintah docker ps pada instans kontainer Anda untuk membuat daftar pada kontainer yang berjalan. Dalam contoh berikut, hanya agen penampung Amazon ECS yang berjalan. Untuk informasi selengkapnya, lihat docker ps di dokumentasi Docker.

```
docker ps
```

Output:

```
CONTAINER ID        IMAGE                            COMMAND             CREATED
      STATUS              PORTS                          NAMES
cee0d6986de0        amazon/amazon-ecs-agent:latest   "/agent"            22 hours ago
      Up 22 hours         127.0.0.1:51678->51678/tcp   ecs-agent
```

Anda dapat menggunakan perintah docker ps -a untuk melihat semua kontainer (bahkan berhenti atau menghancurkan kontainer). Hal ini sangat membantu untuk membuat daftar kontainer yang tiba-tiba berhenti. Pada contoh berikut, kontainer `f7f1f8a7a245` keluar 9 detik yang lalu, sehingga tidak muncul dalam output docker ps tanpa bendera `-a`.

```
docker ps -a
```

Output:

```
CONTAINER ID        IMAGE                                              COMMAND
 CREATED             STATUS                          PORTS                          NAMES
db4d48e411b1        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  19 seconds ago                                                                     ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700
f7f1f8a7a245        busybox:buildroot-2014.02                   "\"sh -c '/bin/sh -c
  22 hours ago        Exited (137) 9 seconds ago                                     ecs-
console-sample-app-static-6-busybox-ce83ce978a87a890ab01
```

```
189a8ff4b5f0        httpd:2                                          "httpd-foreground"
  22 hours ago        Exited (137) 40 seconds ago                                       ecs-
console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600
0c7dca9321e3        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  22 hours ago                                                                          ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0        amazon/amazon-ecs-agent:latest              "/agent"
  22 hours ago        Up 22 hours                    127.0.0.1:51678->51678/tcp   ecs-
agent
```

Lihat Log Docker di Amazon ECS

Anda dapat melihat STDOUT dan mengalirkan STDERR untuk kontainer dengan perintah docker logs. Dalam contoh ini, log ditampilkan untuk *dc7240fe892a* wadah dan disalurkan melalui head perintah untuk singkatnya. Untuk informasi selengkapnya, kunjungi log docker dalam dokumentasi Docker.

> **Note**
>
> Catatan Docker hanya tersedia pada instans kontainer jika Anda menggunakan log driver `json` default. Jika Anda telah mengonfigurasi tugas Anda untuk menggunakan driver `awslogs` log, maka log kontainer Anda tersedia di CloudWatch Log. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke CloudWatch .

```
docker logs dc7240fe892a | head
```

Output:

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472]
 AH00489: Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
 Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
```

```
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

Periksa Kontainer Docker di Amazon ECS

Jika Anda memiliki ID Docker dari sebuah kontainer, Anda dapat memeriksanya dengan perintah docker inspect. Memeriksa kontainer memberikan pandangan paling detail tentang lingkungan tempat kontainer telah diluncurkan. Untuk informasi selengkapnya, lihat pemeriksaan docker dalam dokumentasi Docker.

```
docker inspect dc7240fe892a
```

Output:

```
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "httpd-foreground"
        ],
        "CpuShares": 10,
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
local/apache2/bin",
            "HTTPD_PREFIX=/usr/local/apache2",
            "HTTPD_VERSION=2.4.12",
            "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
        ],
        "ExposedPorts": {
            "80/tcp": {}
```

```
        },
        "Hostname": "dc7240fe892a",
...
```

Mengonfigurasi keluaran verbose dari daemon Docker di Amazon ECS

Jika Anda mengalami masalah dengan wadah atau gambar Docker, Anda dapat mengaktifkan mode debug pada daemon Docker Anda. Menggunakan debugging memberikan lebih banyak output verbose dari daemon, Anda dapat menggunakan ini untuk mengambil pesan kesalahan yang dikirim dari pendaftar kontainer, seperti Amazon ECR.

Important

Prosedur ini ditulis untuk Amazon ECS yang dioptimalkan Amazon Linux AMI. Untuk sistem operasi lain, lihat Aktifkan debugging dan Kontrol dan konfigurasikan Docker dengan systemd dokumentasi Docker.

Untuk menggunakan mode debug daemon Docker di Amazon ECS Amazon Linux AMI yang dioptimalkan Amazon

1. Hubungkan menuju instans kontainer Anda.
2. Buka file pilihan Docker dengan editor teks, seperti vi. Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI, file opsi Docker ada di. `/etc/sysconfig/docker`
3. Cari pernyataan opsi Docker dan menambahkan pilihan `-D` untuk string, di dalam tanda kutip.

   Note

   Jika pernyataan pilihan Docker dimulai dengan #, hapus karakter tersebut untuk menghapus pernyataan dan aktifkan pilihan.

   Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI, pernyataan opsi Docker disebut. `OPTIONS` Misalnya:

   ```
   # Additional startup options for the Docker daemon, for example:
   ```

```
# OPTIONS="--ip-forward=true --iptables=true"
# By default we limit the number of open files per container
OPTIONS="-D --default-ulimit nofile=1024:4096"
```

4. Simpan file, dan tutup editor teks Anda.
5. Mulai ulang daemon Docker.

   ```
   sudo service docker restart
   ```

   Outputnya adalah sebagai berikut:

   ```
   Stopping docker:                                       [  OK  ]
   Starting docker: .                                  [  OK  ]
   ```

6. Mulai ulang agen Amazon ECS.

   ```
   sudo service ecs restart
   ```

Catatan Docker Anda sekarang harus menampilkan lebih banyak output verbose.

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
 server: \"{\\\"errors\\\":[{\\\"code\\\":\\\"DENIED\\\",\\\"message\\\":\\\"User:
 arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
 ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\"Connection\":[]string{\"keep-alive\"}, \"Content-Type\":
[]string{\"application/json; charset=utf-8\"}, \"Date\":[]string{\"Wed, 30 Dec 2015
 21:48:21 GMT\"}, \"Docker-Distribution-Api-Version\":[]string{\"registry/2.0\"},
 \"Content-Length\":[]string{\"235\"}}"
```

Memecahkan masalah Docker di Amazon **`API error (500): devmapper`** ECS

Kesalahan pada Docker berikut ini menunjukkan bahwa penyimpanan kolam yang tipis pada instans kontainer Anda sudah penuh, dan daemon Docker tidak dapat membuat kontainer baru:

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data
 blocks which is less than minimum required 4454 free data blocks. Create more free
 space in thin pool or use dm.min_free_space option to change behavior
```

Secara default, Amazon ECS mengoptimalkan Amazon Linux AMIs dari versi `2015.09.d` dan kemudian diluncurkan dengan volume 8-GiB untuk sistem operasi yang terpasang `/dev/xvda` dan dipasang sebagai root sistem file. Terdapat tambahan volume 22-GiB yang terlampirkan di `/dev/xvdcz` yang Docker gunakan untuk penyimpanan citra dan metadata. Jika ruang penyimpanan ini terisi, daemon Docker tidak dapat membuat kontainer baru.

Cara termudah untuk menambahkan penyimpanan ke instans kontainer Anda adalah dengan mengakhiri instans yang sudah ada dan meluncurkan instans yang baru dengan volume penyimpanan data yang lebih besar. Namun, jika Anda tidak dapat melakukannya, Anda dapat menambahkan penyimpanan ke grup volume yang Docker gunakan dan memperpanjang logika volume dengan mengikuti prosedur di Linux Amazon ECS yang dioptimalkan AMIs.

Jika penyimpanan instans kontainer Anda terlalu cepat terisi, ada beberapa tindakan yang dapat Anda ambil untuk mengurangi efek ini:

- Untuk melihat informasi jajak pendapat tipis, jalankan perintah berikut pada instance container Anda:

  ```
  docker info
  ```

- (Agen kontainer Amazon ECS 1.8.0 dan yang lebih baru) Anda dapat mengurangi jumlah waktu penampung yang berhenti atau keluar tetap pada instance penampung Anda. Parameter variabel konfigurasi agen `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` menetapkan durasi waktu untuk menunggu dari saat tugas dihentikan hingga kontainer Docker dihapus (secara default, nilai ini adalah 3 jam). Hal ini akan menghapus data kontainer Docker. Jika nilai ini disetel terlalu rendah, Anda mungkin tidak dapat memeriksa kontainer yang dihentikan atau melihat log sebelum dihapus. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.
- Anda dapat menghapus kontainer yang tidak berjalan dan gambar yang tidak digunakan dari instance penampung Anda. Anda dapat menggunakan contoh perintah berikut ini untuk menghapus kontainer yang berhenti secara manual dan citra yang tidak terpakai. Kontainer yang dihapus nantinya tidak dapat diperiksa, dan citra yang dihapus harus ditarik lagi sebelum memulai kontainer yang baru dari citra tersebut.

  Untuk menghapus kontainer yang tidak berjalan, jalankan perintah berikut pada instance container Anda:

  ```
  docker rm $(docker ps -aq)
  ```

Untuk menghapus gambar yang tidak digunakan, jalankan perintah berikut pada instance container Anda:

```
docker rmi $(docker images -q)
```

- Anda dapat menghapus blok data yang tidak digunakan dalam wadah. Anda dapat menggunakan perintah berikut ini untuk menjalankan fstrim pada setiap kontainer yang berjalan dan membuang setiap blok data yang tidak digunakan oleh sistem file kontainer.

```
sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -IZ fstrim /proc/Z/root/"
```

Memecahkan masalah Amazon ECS Exec

Berikut ini adalah catatan pemecahan masalah untuk membantu mendiagnosis mengapa Anda mungkin mendapatkan kesalahan saat menggunakan ECS Exec.

Verifikasi menggunakan Pemeriksa Exec

Skrip ECS Exec Checker menyediakan cara untuk memverifikasi dan memvalidasi bahwa cluster dan tugas Amazon ECS Anda telah memenuhi prasyarat untuk menggunakan fitur ECS Exec. Skrip ECS Exec Checker memverifikasi AWS CLI lingkungan dan cluster Anda dan tugas siap untuk ECS Exec, dengan memanggil berbagai atas nama Anda. APIs Alat ini membutuhkan versi terbaru dari AWS CLI dan yang `jq` tersedia. Untuk informasi lebih lanjut, lihat ECS Exec Checker di. GitHub

Terjadi kesalahan saat memanggil **`execute-command`**

Jika terjadi kesalahan `The execute command failed`, berikut ini kemungkinan penyebabnya.

- Tugas tidak memiliki izin yang diperlukan. Verifikasi bahwa definisi tugas yang digunakan untuk meluncurkan tugas Anda memiliki peran IAM tugas yang ditentukan dan peran tersebut memiliki izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin ECS Exec.
- Agen SSM tidak diinstal atau tidak berjalan.
- Ada antarmuka titik akhir Amazon VPC untuk Amazon ECS, tetapi tidak ada satu untuk Systems Manager Session Manager.

Memecahkan masalah Amazon ECS Anywhere

Amazon ECS Anywhere menyediakan dukungan untuk mendaftarkan instans eksternal seperti server lokal atau mesin virtual (VM) ke kluster Amazon ECS Anda. Berikut ini adalah permasalahan umum yang mungkin Anda alami serta rekomendasi pemecahan terhadap masalah umum yang mungkin Anda alami.

Topik

- Permasalahan registrasi instans eksternal
- Masalah jaringan instans eksternal
- Masalah menjalankan tugas pada instans eksternal

Permasalahan registrasi instans eksternal

Saat mendaftarkan instans eksternal dengan cluster Amazon ECS Anda, persyaratan berikut harus dipenuhi:

- AWS Systems Manager Aktivasi, yang terdiri dari ID aktivasi dan kode aktivasi, harus diambil. Anda menggunakannya untuk mendaftarkan instance eksternal sebagai instans terkelola Systems Manager. Ketika aktivasi Systems Manager diminta, tentukan batas pendaftaran dan tanggal kedaluwarsa. Batas pendaftaran tersebut menentukan jumlah maksimum instans yang dapat didaftarkan dengan menggunakan aktivasi. Nilai default untuk batas pendaftaran adalah 1 instance. Tanggal kedaluwarsa menentukan kapan aktivasi akan berakhir. Nilai default adalah 24 jam. Jika aktivasi Systems Manager yang Anda gunakan untuk mendaftarkan instance eksternal Anda tidak valid, mintalah yang baru. Untuk informasi selengkapnya, lihat Mendaftarkan instans eksternal ke kluster Amazon ECS.
- Kebijakan IAM digunakan untuk memberikan instans eksternal Anda izin yang diperlukan untuk berkomunikasi dengan operasi AWS API. Jika kebijakan terkelola ini tidak dibuat dengan benar dan tidak memiliki izin yang diperlukan, pendaftaran instans eksternal akan gagal. Untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere.
- Amazon ECS menyediakan skrip instalasi yang menginstal Docker, agen penampung Amazon ECS, dan Agen Systems Manager pada instans eksternal Anda. Jika penulisan pada penginstalan gagal, penulisan kemungkinan tidak dapat dijalankan lagi pada instans yang sama tanpa terjadinya sebuah kesalahan. Jika ini terjadi, ikuti proses pembersihan untuk menghapus AWS sumber daya dari instance sehingga Anda dapat menjalankan skrip instalasi lagi. Untuk informasi selengkapnya, lihat Membatalkan pendaftaran instans eksternal Amazon ECS.

> **Note**
>
> Ketahuilah bahwa, jika skrip instalasi berhasil meminta dan menggunakan aktivasi Systems Manager, menjalankan skrip instalasi untuk kedua kalinya menggunakan aktivasi Systems Manager lagi. Hal ini dapat menyebabkan Anda mencapai batas pendaftaran untuk aktivasi tersebut. Jika batas ini tercapai, Anda harus membuat aktivasi baru.

- Saat menjalankan skrip instalasi pada instance eksternal untuk beban kerja GPU, jika driver NVIDIA tidak terdeteksi atau dikonfigurasi dengan benar, kesalahan akan terjadi. Skrip instalasi menggunakan `nvidia-smi` perintah untuk mengkonfirmasi keberadaan driver NVIDIA.

Masalah jaringan instans eksternal

Untuk dapat mengomunikasikan perubahan apa pun, instans eksternal Anda memerlukan koneksi jaringan ke AWS. Jika instans eksternal Anda kehilangan koneksi jaringannya AWS, tugas yang berjalan pada instance Anda tetap berjalan kecuali dihentikan secara manual. Setelah koneksi ke AWS dipulihkan, AWS kredensyal yang digunakan oleh agen penampung Amazon ECS dan Agen Systems Manager pada instans eksternal diperpanjang secara otomatis. Untuk informasi selengkapnya tentang AWS domain yang digunakan untuk komunikasi antara instans eksternal Anda dan AWS, lihatJaringan .

Masalah menjalankan tugas pada instans eksternal

Jika tugas atau kontainer Anda gagal dijalanan pada instans eksternal Anda, penyebab paling umum adalah terkait dengan jaringan atau izin. Jika kontainer Anda menarik gambarnya dari Amazon ECR atau dikonfigurasi untuk mengirim log kontainer ke CloudWatch Log, definisi tugas Anda harus menentukan peran IAM eksekusi tugas yang valid. Tanpa peran IAM eksekusi tugas yang valid, kontainer Anda akan gagal untuk memulai. Untuk informasi selengkapnya tentang permasalahan yang berkaitan dengan jaringan, lihat Masalah jaringan instans eksternal.

> **Important**
>
> Amazon ECS menyediakan alat pengumpulan log Amazon ECS. Anda dapat menggunakannya untuk mengumpulkan catatan dari instans eksternal Anda dengan tujuan memecahkan permasalahan. Untuk informasi selengkapnya, lihat Mengumpulkan log kontainer dengan kolektor log Amazon ECS .

AWS Fargate kuota pelambatan

AWS Fargate membatasi tugas Amazon ECS dan tingkat peluncuran pod Amazon EKS ke kuota (sebelumnya disebut sebagai batas) menggunakan algoritma token bucket untuk setiap AWS akun berdasarkan per wilayah. Dengan algoritme ini, akun Anda memiliki bucket yang memegang sejumlah tertentu token. Jumlah token dalam bucket mewakili kuota tarif Anda pada detik tertentu. Setiap akun pelanggan memiliki bucket token task dan pods yang habis berdasarkan jumlah tugas dan pod yang diluncurkan oleh akun pelanggan. Bucket token ini memiliki bucket maksimum yang memungkinkan Anda membuat jumlah permintaan yang lebih tinggi secara berkala, dan tingkat isi ulang yang memungkinkan Anda mempertahankan tingkat permintaan yang stabil selama diperlukan.

Misalnya, ukuran bucket token task dan pods untuk akun pelanggan Fargate adalah 100 token, dan tingkat isi ulang adalah 20 token per detik. Oleh karena itu, Anda dapat segera meluncurkan hingga 100 tugas Amazon ECS dan pod Amazon EKS per akun pelanggan, dengan tingkat peluncuran berkelanjutan 20 tugas Amazon ECS dan pod Amazon EKS per detik.

Tindakan	Kapasitas maksimum bucket (atau Burst rate)	Tingkat isi ulang bucket (atau Tarif berkelanjutan)
Kuota tarif Fargate Resource untuk tugas Amazon ECS On Demand dan pod Amazon EKS [1]	100	20
Kuota tarif Fargate Resource untuk tugas Spot Amazon ECS	100	20

[1] Akun yang hanya meluncurkan pod Amazon EKS memiliki tingkat burst 20 dengan tingkat peluncuran pod berkelanjutan 20 peluncuran pod per detik saat menggunakan versi platform yang disebut dalam versi platform Amazon EKS.

Melambatkan **`RunTask`** API di Fargate

Selain itu, Fargate membatasi tingkat permintaan saat meluncurkan tugas menggunakan Amazon ECS `RunTask` API menggunakan kuota terpisah. Fargate membatasi permintaan Amazon ECS `RunTask` API untuk setiap AWS akun berdasarkan per wilayah. Setiap permintaan yang Anda buat menghapus satu token dari ember. Kami melakukan ini untuk membantu kinerja layanan, dan untuk

memastikan penggunaan yang adil untuk semua pelanggan Fargate. Panggilan API tunduk pada kuota permintaan apakah berasal dari konsol Amazon Elastic Container Service, alat baris perintah, atau aplikasi pihak ketiga. Kuota tarif untuk panggilan ke Amazon ECS `RunTask` API adalah 20 panggilan per detik (burst dan berkelanjutan). Namun, setiap panggilan ke API ini dapat meluncurkan hingga 10 tugas. Ini berarti Anda dapat meluncurkan 100 tugas dalam satu detik dengan melakukan 10 panggilan ke API ini, meminta 10 tugas untuk diluncurkan di setiap panggilan. Demikian pula, Anda juga dapat membuat 20 panggilan ke API ini, meminta 5 tugas untuk diluncurkan di setiap panggilan. Untuk informasi selengkapnya tentang pembatasan API untuk Amazon ECS `RunTask` API, lihat pembatasan permintaan API di Referensi API Amazon ECS.

Dalam praktiknya, tingkat peluncuran tugas dan pod juga tergantung pada pertimbangan lain seperti gambar kontainer yang akan diunduh dan dibongkar, pemeriksaan kesehatan dan integrasi lainnya diaktifkan, seperti mendaftarkan tugas atau pod ke penyeimbang beban. Pelanggan melihat variasi dalam tingkat peluncuran tugas dan pod dibandingkan dengan kuota yang diwakili sebelumnya berdasarkan fitur yang diaktifkan pelanggan.

Menyesuaikan kuota tingkat di Fargate

Anda dapat meminta kenaikan kuota pembatasan tarif Fargate untuk akun Anda. AWS Untuk informasi selengkapnya, lihat Meminta peningkatan kuota di Service Quotas Panduan Pengguna.

Alasan kegagalan Amazon ECS API

Jika tindakan API yang dipicu melalui Amazon ECS API, konsol, atau AWS CLI keluar dengan pesan `failures` galat, berikut ini mungkin membantu memecahkan masalah penyebabnya. Kegagalan mengembalikan alasan dan Nama Sumber Daya Amazon (ARN) dari sumber daya yang terkait dengan kegagalan.

Banyak sumber daya merupakan wilayah yang khusus, jadi ketika menggunakan konsol tersebut pastikan bahwa Anda mengatur Wilayah yang benar untuk sumber daya Anda. Saat menggunakan AWS CLI, pastikan AWS CLI perintah Anda dikirim ke Wilayah yang benar dengan `--region` *`region`* parameter.

Untuk informasi selengkapnya tentang struktur tipe `Failure` data, lihat Kegagalan dalam Referensi API Amazon Elastic Container Service.

Berikut ini adalah contoh pesan kegagalan yang mungkin Anda terima saat menjalankan perintah API.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`DescribeClusters`	`MISSING`	Klaster yang ditentukan tidak ditemukan. Verifikasi ejaan nama klaster.
`DescribeInstances`	`MISSING`	Instans kontainer yang ditentukan tidak ditemukan . Verifikasi bahwa Anda menetapkan klaster instans kontainer yang terdaftar dan bahwa kedua instans kontainer ARN atau ID adalah benar.
`DescribeServices`	`MISSING`	Layanan yang ditentukan tidak ditemukan. Verifikas i bahwa klaster yang benar atau wilayah yang ditentukan dan bahwa layanan ARN atau nama adalah valid.
`DescribeTasks`	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi klaster yang benar atau wilayah yang ditentukan dan bahwa kedua ARN tugas atau ID valid.
`DescribeTasks`	`TaskFailedToStart: RESOURCE:*`	Untuk `RESOURCE:CPU` kesalahan, jumlah yang CPUs diminta oleh tugas tidak tersedia pada instance penampung Anda. Ini biasanya terjadi ketika persyaratan unit CPU dalam definisi tugas Anda lebih

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
		besar dari ukuran CPU EC2 instans Amazon yang ditentukan dalam grup Auto Scaling yang dipetakan ke penyedia kapasitas. Anda perlu memeriksa konfigurasi penyedia kapasitas Anda. Untuk `RESOURCE:MEMORY` kesalahan, jumlah memori yang diminta oleh tugas tidak tersedia pada instance penampung Anda. Ini biasanya terjadi ketika persyaratan jumlah memori dalam definisi tugas Anda lebih besar daripada memori yang didukung pada EC2 instans Amazon yang ditentukan dalam grup Auto Scaling yang dipetakan ke penyedia kapasitas. Anda perlu memeriksa konfigurasi penyedia kapasitas Anda.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`TaskFailedToStart: AGENT`	Instans kontainer tempat Anda mencoba untuk meluncurkan tugas memiliki agen yang saat ini terputus. Untuk mencegah waktu tunggu diperpanjang untuk penempatan tugas, permintaan ditolak. Untuk informasi tentang cara memecahkan masalah agen yang terputus, lihat Bagaimana cara memecahkan masalah agen Amazon ECS yang terputus.
	`TaskFailedToStart: MemberOf placement constraint unsatisfied`	Tidak ada instance container yang memenuhi batasan penempatan yang ditentukan dalam definisi tugas Anda.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`TaskFailedToStart: ATTRIBUTE`	Ketentuan tugas Anda berisi parameter yang memerlukan atribut instans kontainer yang spesifik yang tidak tersedia pada instans kontainer Anda. Contohnya, jika tugas Anda menggunakan mode jaringan `awsvpc`, tetapi tidak ada instans dalam subnet yang Anda tentukan dengan atribut `ecs.capability.task-eni` . Untuk informasi lebih lanjut tentang atribut mana yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate dan Konfigurasi agen kontainer Amazon ECS.
	`TaskFailedToStart: NO ACTIVE INSTANCES`	Tidak ada contoh aktif di penyedia kapasitas Anda. Untuk informasi tentang cara mengelola grup Auto Scaling, lihat grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`TaskFailedToStart: EMPTY CAPACITY PROVIDER`	Tidak ada instance di cluster Anda. Ini kemungkinan besar karena penyedia kapasitas kosong, atau karena instance di penyedia kapasitas tidak terdaftar ke cluster. Untuk informasi tentang cara mengelola grup Auto Scaling, lihat grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling.
`GetTaskProtection`	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi bahwa nama cluster atau ARN dan tugas ARN atau ID valid.
	`TASK_NOT_VALID`	Tugas yang ditentukan bukan bagian dari layanan Amazon ECS. Hanya tugas yang dikelola layanan Amazon ECS yang dapat dilindungi. Verifikasi tugas ARN atau ID dan coba lagi.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
`RunTask` atau `StartTask`	`RESOURCE:*`	Sumber daya tersebut atau sumber daya yang diminta oleh tugas tidak tersedia pada instans kontainer dalam klaster. Jika sumber daya merupakan CPU, memori, port, atau antarmuka jaringan elastis, Anda mungkin perlu menambahkan instans kontainer tambahan untuk klaster Anda. Untuk `RESOURCE:ENI` kesalahan, klaster Anda tidak memiliki titik lampiran elastic network interface yang tersedia, yang diperlukan untuk tugas yang menggunakan mode `awsvpc` jaringan. EC2 Instans Amazon memiliki batasan jumlah antarmuka jaringan yang dapat dilampirkan padanya, dan antarmuka jaringan utama dihitung sebagai satu. Untuk informasi selengkapnya tentang berapa banyak antarmuka jaringan yang didukung untuk setiap jenis instans, lihat Alamat IP Per Antarmuka Jaringan Per Jenis Instance di Panduan EC2 Pengguna Amazon.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
		Untuk `RESOURCE:GPU` kesalahan, jumlah yang GPUs diminta oleh tugas tidak tersedia dan Anda mungkin perlu menambahkan instance kontainer berkemampuan GPU ke klaster Anda. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.
	`AGENT`	Instans kontainer tempat Anda mencoba untuk meluncurkan tugas memiliki agen yang saat ini terputus. Untuk mencegah waktu tunggu diperpanjang untuk penempatan tugas, permintaan ditolak. Untuk informasi tentang cara memecahkan masalah agen yang terputus, lihat Bagaimana cara memecahkan masalah agen Amazon ECS yang terputus.
	`LOCATION`	Instans kontainer tempat Anda coba untuk meluncurkan tugas ada di Availability Zone yang berbeda dari subnet yang Anda tentukan di `awsVpcConfiguration` .

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`ATTRIBUTE`	Ketentuan tugas Anda berisi parameter yang memerlukan atribut instans kontainer yang spesifik yang tidak tersedia pada instans kontainer Anda. Contohnya, jika tugas Anda menggunakan mode jaringan `awsvpc`, tetapi tidak ada instans dalam subnet yang Anda tentukan dengan atribut `ecs.capability.task-eni` . Untuk informasi lebih lanjut tentang atribut mana yang diperlukan untuk parameter ketentuan tugas tertentu dan variabel konfigurasi agen, lihat Parameter definisi tugas Amazon ECS untuk jenis peluncuran Fargate dan Konfigurasi agen kontainer Amazon ECS.
`StartTask`	`MISSING`	Instance container yang Anda coba untuk meluncurkan tugas ke tidak dapat ditemukan. Periksa apakah cluster atau Region yang salah ditentukan, atau contoh kontainer ARN atau ID salah eja.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`INACTIVE`	Instance container yang Anda coba untuk meluncurkan tugas sebelumnya telah dideregis trasi dengan Amazon ECS dan tidak dapat digunakan.
`StopServiceDeployment`	`ECS deployment failed`	Akun penipuan menjalankan `StopServiceDeployment` API.
`TagResource`	`InvalidParameterException`	ARN untuk layanan yang Anda beri tag memiliki format pendek. Anda harus bermigrasi ke format panjang. Untuk informasi tentang cara memigrasi ARN, lihat. Migrasikan ARN layanan singkat Amazon ECS ke ARN yang panjang
`UpdateTaskProtection`	`DEPLOYMENT_BLOCKED`	Tidak dapat mengatur perlindungan tugas karena satu atau lebih tugas yang dilindungi mencegah penyebaran layanan mencapai kondisi mapan. Hapus perlindungan tugas pada tugas yang ada atau tunggu hingga perlindungan tugas berakhir.

Tindakan API	Alasan kegagalan atau alasan berhenti	Penyebab
	`MISSING`	Tugas yang ditentukan tidak ditemukan. Verifikasi bahwa nama cluster atau ARN dan tugas ARN atau ID valid.
	`TASK_NOT_VALID`	Tugas yang ditentukan bukan bagian dari layanan Amazon ECS. Hanya tugas yang dikelola layanan Amazon ECS yang dapat dilindungi. Verifikasi tugas ARN atau ID dan coba lagi.

> **Note**
>
> Selain skenario kegagalan yang dijelaskan di sini, operasi API juga dapat gagal karena pengecualian, yang mengakibatkan respons kesalahan. Untuk daftar pengecualian tersebut, lihat Kesalahan Umum.

Keamanan di Amazon Elastic Container Service

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon Elastic Container Service, lihat AWS Layanan dalam Lingkup menurut Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon ECS. Topik berikut menunjukkan cara mengonfigurasi Amazon ECS untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon ECS Anda.

Topik

- Identity and Access Management untuk Amazon Elastic Container Service
- Logging dan Monitoring di Amazon Elastic Container Service
- Validasi kepatuhan untuk Amazon Elastic Container Service
- AWS Fargate Standar Pemrosesan Informasi Federal (FIPS-140)
- Keamanan Infrastruktur di Amazon Elastic Container Service
- AWS model tanggung jawab bersama untuk Amazon ECS
- Praktik terbaik keamanan jaringan untuk Amazon ECS
- Praktik terbaik tugas Amazon ECS dan keamanan kontainer

Identity and Access Management untuk Amazon Elastic Container Service

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon ECS. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon Elastic Container Service bekerja dengan IAM
- Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service
- AWS kebijakan terkelola untuk Amazon Elastic Container Service
- Menggunakan peran terkait layanan untuk Amazon ECS
- Peran IAM untuk Amazon ECS
- Izin diperlukan untuk konsol Amazon ECS
- Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green
- Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS
- Berikan izin untuk menandai sumber daya pada pembuatan
- Memecahkan masalah identitas dan akses Amazon Elastic Container Service
- Praktik terbaik IAM untuk Amazon ECS

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon ECS.

Pengguna layanan — Jika Anda menggunakan layanan Amazon ECS untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Amazon ECS untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta

izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon ECS, lihatMemecahkan masalah identitas dan akses Amazon Elastic Container Service.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon ECS di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon ECS. Tugas Anda adalah menentukan fitur dan sumber daya Amazon ECS mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Amazon ECS, lihat. Bagaimana Amazon Elastic Container Service bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon ECS. Untuk melihat contoh kebijakan berbasis identitas Amazon ECS yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Guna mengetahui informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Autentikasi multi-faktor AWS di IAM dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial

sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Untuk mengambil peran IAM sementara AWS Management Console, Anda dapat beralih dari pengguna ke peran IAM (konsol). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Buat peran untuk penyedia identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.

- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
  - Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS permintaan API. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensyal

sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan

terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat Pilih antara kebijakan yang dikelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat Ringkasan daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh

batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.

- Kebijakan kontrol layanan (SCPs) — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.
- Kebijakan kontrol sumber daya (RCPs) — RCPs adalah kebijakan JSON yang dapat Anda gunakan untuk menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda tanpa memperbarui kebijakan IAM yang dilampirkan ke setiap sumber daya yang Anda miliki. RCP membatasi izin untuk sumber daya di akun anggota dan dapat memengaruhi izin efektif untuk identitas, termasuk Pengguna root akun AWS, terlepas dari apakah itu milik organisasi Anda. Untuk informasi selengkapnya tentang Organizations dan RCPs, termasuk daftar dukungan Layanan AWS tersebut RCPs, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon Elastic Container Service bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon ECS, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon ECS.

Fitur IAM	Dukungan Amazon ECS
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Parsial
Kunci kondisi kebijakan	Ya
ACLs	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Ya
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon ECS dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Amazon ECS

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Amazon ECS

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Kebijakan berbasis sumber daya dalam Amazon ECS

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke principal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Tindakan kebijakan untuk Amazon ECS

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan Amazon ECS, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon ECS menggunakan awalan berikut sebelum tindakan:

```
ecs
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "ecs:action1",
      "ecs:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "ecs:Describe*"
```

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Sumber daya kebijakan untuk Amazon ECS

Mendukung sumber daya kebijakan: Sebagian

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon ECS dan jenisnya ARNs, lihat Sumber daya yang ditentukan oleh Amazon Elastic Container Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service.

Beberapa tindakan Amazon ECS API mendukung banyak sumber daya. Misalnya, beberapa klaster dapat direferensikan ketika memanggil tindakan API `DescribeClusters`. Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARNs dengan koma.

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Misalnya, sumber daya cluster Amazon ECS memiliki ARN berikut:

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

Untuk menentukan `my-cluster-1` dan `my-cluster-2` mengelompokkan dalam pernyataan Anda, gunakan yang berikut ini ARNs:

```
"Resource": [
```

```
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-1",
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-2"
```

Untuk menentukan semua klaster milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/*"
```

Untuk definisi tugas, Anda dapat menentukan revisi terbaru, atau revisi tertentu.

Untuk menentukan semua revisi definisi tugas, gunakan wildcard (*):

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:*"
```

Untuk menentukan revisi definisi tugas tertentu, gunakan $ {TaskDefinitionRevisionNumber}:

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:${TaskDefinitionRevisionNumber}"
```

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Kunci kondisi kebijakan untuk Amazon ECS

Mendukung kunci kondisi kebijakan khusus layanan: Yes

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tanda dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Amazon ECS mendukung kunci kondisi khusus layanan berikut yang dapat Anda gunakan untuk menyediakan pemfilteran halus untuk kebijakan IAM Anda:

Kunci Syarat	Deskripsi	Jenis Evaluasi
aws: RequestTag /$ {} TagKey	Kunci konteks diformat `"aws:RequestTag/ tag-key":"tag-value "` di mana *tag-key* dan *tag-value* merupakan kunci tag dan pasangan nilai. Memeriksa apakah pasangan kunci tag — nilai ada dalam permintaan. AWS Misalnya, Anda dapat memeriksa apakah permintaan tersebut beserta kunci tanda `"Dept"` dan memiliki nilai `"Accounting"` .	Rangkaian
aws: ResourceTag /$ {} TagKey	Kunci konteks diformat `"aws:ResourceTag/ tag-key":"tag-value "` di mana *tag-key* dan *tag-value* merupakan kunci tag dan pasangan nilai. Periksa apakah tanda yang dilampirkan ke sumber daya identitas (pengguna atau peran) cocok dengan nama dan nilai kunci yang ditentukan.	String
aws: TagKeys	Kunci konteks ini diformat `"aws:TagKeys":" tag-key"` di mana *tag-key* adalah daftar kunci tag tanpa nilai (misalnya,`["Dept","Cost-Center"]`). Memeriksa kunci tag yang ada dalam AWS permintaan.	String
ecs: ResourceTag /$ {} TagKey	Kunci konteks diformat `"ecs:ResourceTag/ tag-key":"tag-value "` di mana *tag-key* dan *tag-value* merupakan kunci tag dan pasangan nilai.	String

Kunci Syarat	Deskripsi	Jenis Evaluasi
	Periksa apakah tanda yang dilampirkan ke sumber daya identitas (pengguna atau peran) cocok dengan nama dan nilai kunci yang ditentukan.	
ecs:pengaturan akun	kunci konteksnya diformat "ecs:account-setting":" *account-setting* " di mana *account-setting* nama pengaturan akun.	String
ecs: auto-assign-public-ip	Kunci konteks diformat "ecs:auto-assign-public-ip":" *value*" di mana *value-* "" atau true "false".	String
ecs:kapasitas-penyedia	Kunci konteks diformat "ecs:capacity-provider":" *capacity-provider-arn* " di *capacity-provider-arn* mana ARN untuk penyedia kapasitas.	ARN, Null
ecs:klaster	Kunci konteks diformat "ecs:cluster":" *cluster-arn* " di mana *cluster-arn* ARN untuk cluster Amazon ECS.	ARN, Null
ecs:komputasi-kompatibilitas	Kunci konteks diformat "ecs:compute-compatability":" *compatability-type* " di mana *compatability-type* adalah "FARGATE“," EC2 "atau"EXTERNAL”.	String
ecs:instans kontainer	Kunci konteks diformat "ecs:container-instances":" *container-instance-arns* " di mana *container-instance-arns* satu atau lebih instance ARNs kontainer.	ARN, Null
ecs:nama-kontainer	Kunci konteks diformat "ecs:container-name":" *container-name* " di mana *container-instance-* adalah nama wadah Amazon ECS yang didefinisikan dalam definisi tugas.	String

Kunci Syarat	Deskripsi	Jenis Evaluasi
ecs: enable-execute-command	Kunci konteks diformat "ecs:enable-execute-command":" *value*" di mana *value-* "" atau true "false".	String
ecs: enable-service-connect	Kunci konteks diformat "ecs:enable-service-connect":" *value*" di *value* mana "true" atau"false".	String
ecs: enable-ebs-volumes	Kunci konteks diformat "ecs:enable-ebs-volumes":" *value*" di *value* mana "true" atau"false".	String
ecs: enable-managed-tags	Kunci konteks diformat "ecs:enable-managed-tags":" *value*" di *value* mana "true" atau"false".	String
ecs: enable-vpc-lattice	Kunci konteks diformat "ecs:vpc-lattice":" *value*" di *value* mana "true" atau"false".	String
ecs: -kunci fargate-ephemeral-storage-kms	Kunci konteks diformat "ecs:fargate-ephemeral-storage-kms-key":" *key*" di *key* mana ID AWS KMS kuncinya.	String
ecs:namespace	Kunci konteks diformat "ecs:namespace":" *namespace-arn* " di mana *namespace-arn* ARN untuk AWS Cloud Map namespace.	ARN, Null
ecs:menyebarkan tag	Kunci konteks diformat "ecs:propagate-tags":" *value*" di mana *value*"TASK_DEFINITION" ,"SERVICE" , atau"NONE".	String
ecs:layanan	Kunci konteks diformat "ecs:service":" *service-arn* " di mana *service-arn* ARN untuk layanan Amazon ECS.	ARN, Null

Kunci Syarat	Deskripsi	Jenis Evaluasi
ecs:ketentuan tugas-definis	Kunci konteks diformat "ecs:task-definition":" *task-definition-arn* " di mana *task-definition-arn* ARN untuk definisi tugas Amazon ECS.	ARN, Null
ecs:subnet	Kunci konteks diformat "ecs:subnet":" *subnets*" di *subnets* mana subnet dari tugas atau layanan yang menggunakan awsvpc mode.	String
ecs:tugas	Kunci konteks diformat "ecs:task":" *task-arn*" di mana *task-arn* ARN untuk layanan Amazon ECS.	ARN, Null
ecs:tugas-cpu	Kunci konteks diformat "ecs:task-cpu":" *task-cpu*" di mana *task-cpu* CPU tugas, sebagai bilangan bulat dengan 1024 = 1 vCPU.	Bilangan Bulat
ecs:tugas-memori	Kunci konteks diformat "ecs:task-memory":" *task-memory* " di mana *task-memory* memori tugas di MiB.	Bilangan Bulat

Untuk melihat daftar kunci kondisi Amazon ECS, lihat Kunci kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Elastic Container Service.

Untuk melihat contoh kebijakan berbasis identitas Amazon ECS, lihat. Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Daftar kontrol akses (ACLs) di Amazon ECS

Mendukung ACLs: Tidak

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan Amazon ECS

> **Important**
>
> Amazon ECS mendukung kontrol akses berbasis atribut untuk semua sumber daya Amazon ECS. Untuk menentukan apakah Anda dapat menggunakan atribut untuk cakupan tindakan, gunakan tabel Tindakan yang ditentukan oleh Amazon ECS di Referensi Otorisasi Layanan. Pertama verifikasi bahwa ada sumber daya di kolom Sumber Daya. Kemudian, gunakan kolom tombol Kondisi untuk melihat tombol untuk action/resource kombinasi.

Mendukung ABAC (tanda dalam kebijakan): Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Penandaan ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi ketika tanda milik prinsipal cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi saat manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang menandai sumber daya Amazon ECS, lihat. Menandai sumber daya Amazon ECS

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Menjelaskan layanan Amazon ECS berdasarkan tag.

Menggunakan kredensyal Sementara dengan Amazon ECS

Mendukung kredensial sementara: Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensil sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensyal sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensil sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat Beralih dari pengguna ke peran IAM (konsol) dalam Panduan Pengguna IAM.

Anda dapat membuat kredensil sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensyal sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensyal sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Teruskan sesi akses untuk Amazon ECS

Mendukung sesi akses maju (FAS): Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.

Peran layanan untuk Amazon ECS

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari

dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon ECS. Edit peran layanan hanya jika Amazon ECS memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Amazon ECS

Mendukung peran terkait layanan: Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan Amazon ECS, lihat. Menggunakan peran terkait layanan untuk Amazon ECS

Contoh kebijakan berbasis identitas untuk Amazon Elastic Container Service

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon ECS. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon ECS, termasuk format ARNs untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Elastic Container Service di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan Amazon ECS

- Izinkan pengguna Amazon ECS untuk melihat izin mereka sendiri
- Contoh cluster Amazon ECS
- Contoh contoh wadah Amazon ECS
- Contoh definisi tugas Amazon ECS
- Jalankan Contoh Tugas Amazon ECS
- Mulai contoh tugas Amazon ECS
- Buat daftar dan jelaskan contoh tugas Amazon ECS
- Buat contoh layanan Amazon ECS
- Perbarui contoh layanan Amazon ECS
- Menjelaskan layanan Amazon ECS berdasarkan tag
- Tolak Contoh Penggantian Namespace Connect Layanan Amazon ECS

Praktik terbaik kebijakan Amazon ECS

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon ECS di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber

daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.

- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Izinkan pengguna Amazon ECS untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
```

```
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh cluster Amazon ECS

Kebijakan IAM berikut memungkinkan izin untuk membuat dan membuat daftar cluster. Tindakan `CreateCluster` dan `ListClusters` tidak menerima sumber daya apa pun, sehingga definisi sumber daya diatur ke `*` untuk semua sumber daya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:ListClusters"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Kebijakan IAM berikut memungkinkan izin untuk mendeskripsikan dan menghapus klaster tertentu. `DeleteCluster`Tindakan `DescribeClusters` dan menerima cluster ARNs sebagai sumber daya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeClusters",
                "ecs:DeleteCluster"
            ],
            "Resource": ["arn:aws:ecs:us-east-1:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

Kebijakan IAM berikut dapat dilampirkan ke pengguna atau grup yang hanya mengizinkan pengguna atau grup tersebut untuk melakukan operasi pada klaster tertentu.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:Describe*",
                "ecs:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ecs:DeleteCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances",
                "ecs:RegisterContainerInstance",
                "ecs:SubmitContainerStateChange",
                "ecs:SubmitTaskStateChange"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
```

```
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTasks",
                "ecs:ListTasks",
                "ecs:UpdateContainerAgent",
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-
east-1:<aws_account_id>:cluster/default"}
            }
        }
    ]
}
```

Contoh contoh wadah Amazon ECS

Pendaftaran instans kontainer ditangani oleh agen Amazon ECS, tetapi mungkin ada saat-saat di mana Anda ingin mengizinkan pengguna membatalkan pendaftaran instance secara manual dari klaster. Mungkin instans kontainer secara tidak sengaja didaftarkan ke klaster yang salah, atau instans diakhiri dengan tugas yang masih berjalan di dalamnya.

Kebijakan IAM berikut memungkinkan pengguna untuk membuat daftar dan membatalkan pendaftaran instance kontainer dalam klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances"
            ],
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
```

```
    ]
}
```

Kebijakan IAM berikut memungkinkan pengguna untuk mendeskripsikan instance kontainer tertentu dalam cluster tertentu. Untuk membuka izin ini hingga semua instans kontainer dalam sebuah klaster, Anda dapat mengganti UUID instans kontainer dengan *.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeContainerInstances"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<cluster_name>/<container_instance_UUID>"]
        }
    ]
}
```

Contoh definisi tugas Amazon ECS

Kebijakan IAM definisi tugas tidak mendukung izin tingkat sumber daya, tetapi kebijakan IAM berikut memungkinkan pengguna untuk mendaftar, mendaftar, dan menjelaskan definisi tugas:

Jika Anda menggunakan konsol, Anda harus menambahkan `CloudFormation: CreateStack` sebagai`Action`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RegisterTaskDefinition",
                "ecs:ListTaskDefinitions",
                "ecs:DescribeTaskDefinition"
            ],
            "Resource": ["*"]
```

```
        }
    ]
}
```

Jalankan Contoh Tugas Amazon ECS

Sumber daya untuk `RunTask` adalah ketentuan tugas. Untuk membatasi klaster mana yang pengguna dapat menjalankan ketentuan tugas, Anda dapat menentukannya dalam blok `Condition`. Keuntungannya adalah Anda tidak perlu membuat daftar ketentuan tugas dan klaster di sumber daya Anda untuk mengizinkan akses yang sesuai. Anda bisa menerapkan satu, yang lain, atau keduanya.

Kebijakan IAM berikut memungkinkan izin untuk menjalankan revisi definisi tugas tertentu pada klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

Mulai contoh tugas Amazon ECS

Sumber daya untuk `StartTask` adalah ketentuan tugas. Untuk membatasi klaster dan instans kontainer mana pengguna dapat memulai ketentuan tugas, Anda dapat menentukannya dalam blok `Condition`. Keuntungannya adalah Anda tidak perlu membuat daftar ketentuan tugas dan klaster di sumber daya Anda untuk mengizinkan akses yang sesuai. Anda bisa menerapkan satu, yang lain, atau keduanya.

Kebijakan IAM berikut memungkinkan izin untuk memulai revisi definisi tugas tertentu pada cluster tertentu dan instance kontainer tertentu.

> **Note**
>
> Untuk contoh ini, saat Anda memanggil `StartTask` API dengan AWS CLI atau AWS SDK lain, Anda harus menentukan revisi definisi tugas agar `Resource` pemetaan cocok.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:StartTask"],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:container-instances":
 ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/<cluster_name>/
<container_instance_UUID>"]
                }
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

Buat daftar dan jelaskan contoh tugas Amazon ECS

Kebijakan IAM berikut memungkinkan pengguna untuk membuat daftar tugas untuk klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:ListTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["*"]
```

```
        }
    ]
}
```

Kebijakan IAM berikut memungkinkan pengguna untuk mendeskripsikan tugas tertentu dalam klaster tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task/<cluster_name>/
<task_UUID>"]
        }
    ]
}
```

Buat contoh layanan Amazon ECS

Kebijakan IAM berikut memungkinkan pengguna untuk membuat layanan Amazon ECS di: AWS Management Console

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:CreateService",
```

```
                "elasticloadbalancing:Describe*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Perbarui contoh layanan Amazon ECS

Kebijakan IAM berikut memungkinkan pengguna untuk memperbarui layanan Amazon ECS di: AWS Management Console

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:UpdateService",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
```

```
        }
    ]
}
```

Menjelaskan layanan Amazon ECS berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya Amazon ECS berdasarkan tag. Contoh berikut menunjukkan cara agar Anda dapat membuat kebijakan yang mengizinkan penjelasan layanan. Namun, izin diberikan hanya jika tanda layanan `Owner` memiliki nilai nama pengguna dari pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DescribeServices",
            "Effect": "Allow",
            "Action": "ecs;:DescribeServices",
            "Resource": "*"
        },
        {
            "Sid": "ViewServiceIfOwner",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "arn:aws:ecs:*:*:service/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna bernama `richard-roe` mencoba mendeskripsikan layanan Amazon ECS, layanan tersebut harus diberi tag `Owner=richard-roe` atau. `owner=richard-roe` Jika tidak, aksesnya akan ditolak. Kunci tanda syarat `Owner` cocok dengan `Owner` dan `owner` karena nama kunci syarat tidak terpengaruh

huruf besar/kecil. Untuk informasi lebih lanjut, lihat Elemen Kebijakan IAM JSON: Persyaratan dalam Panduan Pengguna IAM.

Tolak Contoh Penggantian Namespace Connect Layanan Amazon ECS

Kebijakan IAM berikut menyangkal pengguna untuk mengganti namespace Service Connect default dalam konfigurasi layanan. Namespace default diatur dalam cluster. Namun, Anda dapat menggantinya dalam konfigurasi layanan. Untuk konsistensi, pertimbangkan untuk mengatur semua layanan baru Anda untuk menggunakan namespace yang sama. Gunakan kunci konteks berikut untuk meminta layanan menggunakan namespace tertentu. Ganti`<region>`,`<aws_account_id>`, `<cluster_name>` dan `<namespace_id>` dengan milik Anda sendiri dalam contoh berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:namespace":
 "arn:aws:servicediscovery:<region>:<aws_account_id>:namespace/<namespace_id>"
                }
            },
            "Resource": "*"
        }
    ]
}
```

AWS kebijakan terkelola untuk Amazon Elastic Container Service

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan yang dikelola pelanggan IAM yang hanya memberi tim Anda izin yang mereka butuhkan.

Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan AWS AWS terkelola, lihat kebijakan terkelola di Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan AWS terkelola untuk mendukung fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan AWS terkelola saat fitur baru diluncurkan atau saat operasi baru tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan ReadOnlyAccess AWS terkelola menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS tambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat kebijakan yang dikelola AWS untuk fungsi tugas di Panduan Pengguna IAM.

Amazon ECS dan Amazon ECR menyediakan beberapa kebijakan terkelola dan hubungan kepercayaan yang dapat Anda lampirkan ke pengguna, grup, peran, EC2 instans Amazon, dan tugas Amazon ECS yang memungkinkan berbagai tingkat kontrol atas sumber daya dan operasi API. Anda dapat menerapkan kebijakan ini secara langsung atau menggunakannya sebagai titik awal untuk membuat kebijakan Anda sendiri. Untuk informasi selengkapnya tentang kebijakan terkelola Amazon ECR, lihat kebijakan yang dikelola Amazon ECR.

Amazonecs_ FullAccess

Anda dapat melampirkan kebijakan `AmazonECS_FullAccess` ke identitas IAM Anda. Kebijakan ini memberikan akses administratif ke sumber daya Amazon ECS dan memberikan identitas IAM (seperti pengguna, grup, atau peran) akses ke layanan AWS yang terintegrasi dengan Amazon ECS untuk menggunakan semua fitur Amazon ECS. Menggunakan kebijakan ini memungkinkan akses ke semua fitur Amazon ECS yang tersedia di. AWS Management Console

Untuk melihat izin kebijakan ini, lihat AmazonECS_ FullAccess di Referensi Kebijakan Terkelola.AWS

Amazon ECSInfrastructure RolePolicyForVolumes

Anda dapat melampirkan kebijakan `AmazonECSInfrastructureRolePolicyForVolumes` terkelola ke entitas IAM Anda.

Kebijakan ini memberikan izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API atas nama Anda. Anda dapat melampirkan kebijakan ini ke peran IAM yang Anda berikan dengan konfigurasi volume saat meluncurkan tugas dan layanan Amazon ECS. Peran ini memungkinkan Amazon ECS untuk mengelola volume yang melekat pada tugas Anda. Untuk informasi selengkapnya, lihat peran IAM infrastruktur Amazon ECS.

Untuk melihat izin kebijakan ini, lihat Amazon ECSInfrastructure RolePolicyForVolumes di Referensi Kebijakan AWS Terkelola.

EC2ContainerServiceforEC2Peran Amazon

Anda dapat melampirkan kebijakan `AmazonEC2ContainerServiceforEC2Role` ke identitas IAM Anda. Kebijakan ini memberikan izin administratif yang memungkinkan instans penampung Amazon ECS melakukan panggilan atas nama Anda. AWS Untuk informasi selengkapnya, lihat Peran IAM instance wadah Amazon ECS.

Amazon ECS melampirkan kebijakan ini ke peran layanan yang memungkinkan Amazon ECS melakukan tindakan atas nama Anda terhadap EC2 instans Amazon atau instans eksternal.

Untuk melihat izin kebijakan ini, lihat EC2ContainerServiceforEC2Peran Amazon di Referensi Kebijakan AWS Terkelola.

Pertimbangan

Anda harus mempertimbangkan rekomendasi dan pertimbangan berikut saat menggunakan kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola.

- Dengan mengikuti saran keamanan standar pemberian hak istimewa paling rendah, Anda dapat memodifikasi kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` agar sesuai dengan kebutuhan spesifik Anda. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan. Misalnya, `UpdateContainerInstancesState` izin diberikan untuk pengeringan Instans Spot. Jika izin tersebut tidak diperlukan untuk kasus penggunaan Anda, izin tersebut dapat dikecualikan dengan menggunakan kebijakan kustom.
- Kontainer yang berjalan pada instans kontainer Anda memiliki akses ke semua izin yang disediakan untuk peran instans kontainer melalui Metadata instans. Kami merekomendasikan Anda untuk membatasi izin dalam peran instans kontainer Anda agar hanya menyediakan izin minimal di kebijakan `AmazonEC2ContainerServiceforEC2Role` terkelola. Jika kontainer dalam tugas Anda memerlukan izin tambahan yang tidak terdaftar, sebaiknya berikan tugas tersebut dengan peran IAM mereka sendiri. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

Kontainer pada jembatan `docker0` dapat dicegah untuk tidak mengakses izin yang disediakan untuk peran instans kontainer.. Tindakan ini masih bisa dilakukan bersamaan dengan berlakunya izin yang diberikan Peran IAM tugas Amazon ECS dengan menjalankan perintah iptables pada instans kontainer Anda. Kontainer tidak dapat melakukan kueri terhadap metadata instans saat aturan ini berlaku. Perintah ini mengasumsikan konfigurasi jembatan Docker default dan tidak bekerja dengan kontainer yang menerapkan mode jaringan `host`. Untuk informasi selengkapnya, lihat Mode jaringan.

```
sudo yum install -y iptables-services; sudo iptables --insert DOCKER USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
```

Anda harus menyimpan aturan iptables ini pada instans kontainer Anda untuk itu untuk bertahan saat booting ulang. Untuk AMI Amazon ECS yang dioptimalkan, gunakan perintah berikut. Untuk sistem operasi lain, konsultasikan dokumentasi untuk OS tersebut.

- Untuk Amazon ECS yang dioptimalkan Amazon Linux 2 AMI:

  ```
  sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
  ```

- Untuk Amazon ECS yang dioptimalkan Amazon Linux AMI:

  ```
  sudo service iptables save
  ```

Amazon EC2 ContainerServiceEventsRole

Anda dapat melampirkan kebijakan `AmazonEC2ContainerServiceEventsRole` ke identitas IAM Anda. Kebijakan ini memberikan izin yang memungkinkan Amazon EventBridge (sebelumnya CloudWatch Acara) menjalankan tugas atas nama Anda. Kebijakan ini dapat dilampirkan ke peran IAM yang ditentukan saat Anda membuat tugas terjadwal. Untuk informasi selengkapnya, lihat Peran Amazon ECS EventBridge IAM.

Untuk melihat izin kebijakan ini, lihat Amazon EC2 ContainerServiceEventsRole di Referensi Kebijakan AWS Terkelola.

Amazon ECSTask ExecutionRolePolicy

Kebijakan IAM `AmazonECSTaskExecutionRolePolicy` terkelola memberikan izin yang diperlukan oleh agen kontainer Amazon ECS dan AWS Fargate agen kontainer untuk melakukan panggilan AWS API atas nama Anda. Kebijakan ini dapat ditambahkan ke peran IAM eksekusi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Untuk melihat izin kebijakan ini, lihat Amazon ECSTask ExecutionRolePolicy di Referensi Kebijakan AWS Terkelola.

Amazon ECSService RolePolicy

Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola memungkinkan Amazon Elastic Container Service mengelola klaster Anda. Kebijakan ini dapat ditambahkan ke peran terkait layanan AWSServiceRoleForECS Anda.

Untuk melihat izin kebijakan ini, lihat Amazon ECSService RolePolicy di Referensi Kebijakan AWS Terkelola.

`AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecur`

Anda dapat melampirkan `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` kebijakan ke entitas IAM Anda. Kebijakan ini memberikan akses administratif ke AWS Private Certificate Authority, Secrets Manager, dan AWS Layanan lain yang diperlukan untuk mengelola fitur Amazon ECS Service Connect TLS atas nama Anda.

Untuk melihat izin kebijakan ini, lihat Amazon ECSInfrastructure RolePolicyForServiceConnectTransportLayerSecurity di Referensi Kebijakan AWS Terkelola.

`AWSApplicationAutoscalingECSServicePolicy`

Anda tidak dapat melampirkan `AWSApplicationAutoscalingECSServicePolicy` ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan Application Auto Scaling untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya. lihat Peran tertaut layanan untuk Application Auto Scaling.

Untuk melihat izin kebijakan ini, lihat Kebijakan AWSApplicationPenskalaan Otomatis di Referensi ECSService Kebijakan Terkelola.AWS

AWSCodeDeployRoleForECS

Anda tidak dapat melampirkan `AWSCodeDeployRoleForECS` ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CodeDeploy untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Membuat peran layanan CodeDeploy di Panduan AWS CodeDeploy Pengguna.

Untuk melihat izin kebijakan ini, lihat AWSCodeDeployRoleForECS di Referensi Kebijakan AWS Terkelola.

AWSCodeDeployRoleForECSLimited

Anda tidak dapat melampirkan `AWSCodeDeployRoleForECSLimited` ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CodeDeploy untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Membuat peran layanan CodeDeploy di Panduan AWS CodeDeploy Pengguna.

Untuk melihat izin kebijakan ini, lihat AWSCodeDeployRoleForECSLimitedi Referensi Kebijakan AWS Terkelola.

AmazonECSInfrastructureRolePolicyForLoadBalancers

Anda dapat melampirkan `AmazonECSInfrastructureRolePolicyForLoadBalancers` kebijakan ke entitas IAM Anda. Kebijakan ini memberikan izin yang memungkinkan Amazon ECS mengelola sumber daya Elastic Load Balancing atas nama Anda. Kebijakan tersebut meliputi:

- Izin baca-saja untuk mendeskripsikan pendengar, aturan, grup target, dan kesehatan target
- Izin untuk mendaftarkan dan membatalkan pendaftaran target dengan grup target
- Izin untuk memodifikasi pendengar untuk Application Load Balancers dan Network Load Balancers
- Izin untuk mengubah aturan untuk Application Load Balancers

Izin ini memungkinkan Amazon ECS untuk secara otomatis mengelola konfigurasi penyeimbang beban saat layanan dibuat atau diperbarui, memastikan perutean lalu lintas yang tepat ke kontainer Anda.

Untuk melihat izin kebijakan ini, lihat Amazon ECSInfrastructure RolePolicyForLoadBalancers di Referensi Kebijakan AWS Terkelola.

AmazonECSInfrastructureRolePolicyForVpcLattice

Anda dapat melampirkan AmazonECSInfrastructureRolePolicyForVpcLattice kebijakan ke entitas IAM Anda. Kebijakan ini Menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola fitur Kisi VPC di beban kerja Amazon ECS atas nama Anda.

Untuk melihat izin kebijakan ini, lihat Amazon ECSInfrastructure RolePolicyForVpcLattice di Referensi Kebijakan AWS Terkelola.

Menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola fitur VPC Lattice di beban kerja Amazon ECS atas nama Anda.

Amazon ECS memperbarui kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon ECS sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen Amazon ECS.

Perubahan	Deskripsi	Tanggal
Tambahkan izin ke the section called “EC2ContainerServiceforEC2Peran Amazon”	Kebijakan IAM AmazonEC2ContainerServiceforEC2Role terkelola telah diperbarui untuk menyertakan ecs:ListTagsForResource izin. Izin ini memungkinkan agen Amazon ECS untuk mengambil tag instance tugas dan container melalui titik akhir metadata tugas (). ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags	Agustus 4, 2025
Tambahkan izin ke Amazon ECSInfrastructure RolePolicyForLoadBalancers.	Kebijakan IAM AmazonECSInfrastructureRolePolicyForLoadBalan	Juli 25, 2025

Perubahan	Deskripsi	Tanggal
	`cers` terkelola diperbaru i dengan izin baru untuk mendeskripsikan, membatalk an pendaftaran, dan mendaftarkan grup sasaran.	
Tambahkan the section called "`AmazonECSInfrastru ctureRolePolicyFor LoadBalancers`" kebijakan baru	Menambahkan ECSInfras tructure RolePolicyForLoadB alancers kebijakan Amazon baru yang menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola penyeimbang beban yang terkait dengan beban kerja Amazon ECS.	Juli 15, 2025
Tambahkan izin ke Amazon ECSService RolePolicy.	Kebijakan IAM `AmazonECS ServiceRolePolicy` terkelola diperbarui dengan AWS Cloud Map izin baru yang Amazon ECS dapat memperbarui atribut AWS Cloud Map layanan untuk layanan yang dikelola Amazon ECS.	Juli 15, 2025
Tambahkan izin ke Amazon ECSService RolePolicy	Kebijakan IAM `AmazonECS ServiceRolePolicy` terkelola diperbarui dengan AWS Cloud Map izin baru yang Amazon ECS dapat memperbarui atribut AWS Cloud Map layanan untuk layanan yang dikelola Amazon ECS.	Juni 24, 2025

Perubahan	Deskripsi	Tanggal
Tambahkan izin ke Amazon ECSInfrastructure RolePolicyForVolumes	`AmazonECSInfrastructureRolePolicyForVolumes` Kebijakan telah diperbarui untuk menambahkan `ec2:DescribeInstances` izin. Izin tersebut membantu mencegah tabrakan nama perangkat untuk volume Amazon EBS yang dilampirkan ke tugas Amazon ECS yang berjalan pada instance container yang sama.	Juni 2, 2025
Tambahkan Amazon baru ECSInfrastructure RolePolicyForVpcLattice	Menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola fitur VPC Lattice di beban kerja Amazon ECS atas nama Anda.	November 18, 2024
Tambahkan izin ke Amazon ECSInfrastructure RolePolicyForVolumes	`AmazonECSInfrastructureRolePolicyForVolumes` Kebijakan ini telah diperbarui untuk memungkinkan pelanggan membuat volume Amazon EBS dari snapshot.	Oktober 10, 2024

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke the section called “Amazonecs_FullAccess”	`AmazonECS_FullAccess` Kebijakan telah diperbarui untuk menambahkan `iam:PassRole` izin untuk peran IAM untuk peran bernama. `ecsInfrastructureRole` Ini adalah peran IAM default yang dibuat oleh AWS Management Console yang dimaksudkan untuk digunakan sebagai peran infrastruktur ECS yang memungkinkan Amazon ECS mengelola volume Amazon EBS yang melekat pada tugas ECS.	Agustus 13, 2024
Tambahkan ECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity kebijakan Amazon baru	Menambahkan ECSInfrastructure RolePolicyForServiceConnectTransportLayerSecurity kebijakan Amazon baru yang menyediakan akses administratif ke AWS KMS AWS Private Certificate Authority, Secrets Manager, dan memungkinkan fitur Amazon ECS Service Connect TLS berfungsi dengan baik.	Januari 22, 2024

Perubahan	Deskripsi	Tanggal
Tambahkan kebijakan baru Amazon ECSInfrastructure RolePolicyForVolumes	`AmazonECSInfrastructureRolePolicyForVolumes` Kebijakan itu ditambahkan. Kebijakan ini memberikan izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API guna mengelola volume Amazon EBS yang terkait dengan beban kerja Amazon ECS.	Januari 11, 2024
Tambahkan izin ke Amazon ECSService RolePolicy	Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola telah diperbarui dengan `events` izin baru `autoscaling` dan `autoscaling-plans` tambahan serta izin.	Desember 4, 2023
Tambahkan izin ke Amazon EC2 ContainerServiceEventsRole	Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola telah diperbarui untuk mengizinkan akses ke operasi AWS Cloud Map `DiscoverInstancesRevision` API.	4 Oktober 2023

Perubahan	Deskripsi	Tanggal
Tambahkan izin ke Peran Amazon EC2 Container Servicefor EC2	`AmazonEC2ContainerServicefo rEC2Role` Kebijakan telah dimodifikasi untuk menambahkan `ecs:TagResource` izin, yang mencakup kondisi yang membatasi izin hanya untuk klaster yang baru dibuat dan instance kontainer terdaftar.	6 Maret 2023
Tambahkan izin ke the section called “Amazonecs_ FullAccess”	`AmazonECS_FullAccess` Kebijakan diubah untuk menambahkan `elasticloadbalancing:AddTags` izin, yang mencakup kondisi yang membatasi izin hanya untuk penyeimbang beban, grup target, aturan, dan pendengar yang baru dibuat. Izin ini tidak mengizinkan tag ditambahkan ke sumber daya Elastic Load Balancing yang sudah dibuat.	4 Januari 2023
Amazon ECS mulai melacak perubahan	Amazon ECS mulai melacak perubahan untuk kebijakan yang AWS dikelola.	8 Juni 2021

Menghapus kebijakan IAM AWS terkelola untuk Amazon Elastic Container Service

Kebijakan IAM AWS terkelola berikut dihapus secara bertahap. Kebijakan ini sekarang digantikan oleh kebijakan yang sudah diperbarui. Kami menyarankan Anda untuk memperbarui pengguna atau peran untuk menggunakan kebijakan yang diperbarui tersebut.

Amazon EC2 ContainerServiceFullAccess

> **Important**
>
> Kebijakan IAM yang `AmazonEC2ContainerServiceFullAccess` dikelola dihapus pada 29 Januari 2021, sebagai tanggapan atas temuan keamanan dengan izin. `iam:passRole` Izin ini memberikan akses ke semua sumber daya termasuk kredensial untuk berperan di akun. Setelah kebijakan dihapus, Anda tidak dapat melampirkan kebijakan ke pengguna atau peran baru. Setiap pengguna atau peran yang sudah memiliki kebijakan yang dilampirkan dapat terus menggunakannya. Namun, kami menyarankan Anda memperbarui pengguna atau peran Anda untuk menggunakan kebijakan `AmazonECS_FullAccess` terkelola sebagai gantinya. Untuk informasi selengkapnya, lihat Migrasi ke kebijakan terkelola `AmazonECS_FullAccess`.

Amazon EC2 ContainerServiceRole

> **Important**
>
> Kebijakan IAM `AmazonEC2ContainerServiceRole` terkelola dihapus. Sekarang digantikan oleh peran terkait layanan Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon ECS.

Amazon EC2 ContainerServiceAutoscaleRole

> **Important**
>
> Kebijakan IAM `AmazonEC2ContainerServiceAutoscaleRole` terkelola dihapus. Sekarang digantikan oleh peran terkait layanan Application Auto Scaling untuk Amazon ECS. Untuk informasi selengkapnya, lihat Peran tertaut layanan untuk Application Auto Scaling dalam Panduan Pengguna Application Auto Scaling.

Migrasi ke kebijakan terkelola **`AmazonECS_FullAccess`**

Kebijakan IAM yang `AmazonEC2ContainerServiceFullAccess` dikelola dihapus pada 29 Januari 2021, sebagai tanggapan atas temuan keamanan dengan izin. `iam:passRole` Izin ini

memberikan akses ke semua sumber daya termasuk kredensial untuk berperan di akun. Setelah kebijakan dihapus, Anda tidak dapat melampirkan kebijakan tersebut ke grup, pengguna atau peran baru. Setiap grup, pengguna, atau peran yang sudah memiliki kebijakan yang dilampirkan dapat terus menggunakannya. Namun, kami menyarankan Anda memperbarui grup, pengguna, atau peran Anda untuk menggunakan kebijakan `AmazonECS_FullAccess` terkelola sebagai gantinya.

Izin yang diberikan oleh kebijakan `AmazonECS_FullAccess` termasuk daftar lengkap izin yang diperlukan untuk menggunakan ECS sebagai administrator. Jika saat ini Anda menggunakan izin yang diberikan oleh `AmazonEC2ContainerServiceFullAccess` kebijakan yang tidak ada dalam `AmazonECS_FullAccess` kebijakan, Anda dapat menambahkannya ke pernyataan kebijakan sebaris. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Service.

Gunakan langkah-langkah berikut untuk menentukan apakah Anda memiliki grup, pengguna, atau peran yang saat ini menggunakan kebijakan IAM `AmazonEC2ContainerServiceFullAccess` terkelola. Kemudian perbarui grup, pengguna, atau peran tersebut untuk melepaskan kebijakan sebelumnya dan melampirkan kebijakan `AmazonECS_FullAccess`.

Untuk memperbarui grup, pengguna, atau peran untuk menggunakan kebijakan FullAccess Amazonecs_ ()AWS Management Console

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan lalu cari dan pilih kebijakan `AmazonEC2ContainerServiceFullAccess`.
3. Pilih tab Penggunaan kebijakan yang menampilkan peran IAM apa pun yang saat ini menggunakan kebijakan ini.
4. Untuk setiap peran IAM yang saat ini menggunakan `AmazonEC2ContainerServiceFullAccess` kebijakan, pilih peran dan gunakan langkah-langkah berikut untuk melepaskan kebijakan yang dihapus secara bertahap dan melampirkan kebijakan. `AmazonECS_FullAccess`

    a. Pada tab Izin, pilih X di sebelah EC2 ContainerServiceFullAccess kebijakan Amazon.

    b. Pilih Tambahkan izin.

    c. Pilih Lampirkan kebijakan yang ada secara langsung, cari dan pilih FullAccess kebijakan AmazonECS_, lalu pilih Berikutnya: Tinjau.

    d. Tinjau perubahan lalu pilih Tambahkan izin.

e. Ulangi langkah-langkah ini untuk setiap grup, pengguna, atau peran yang menggunakan `AmazonEC2ContainerServiceFullAccess` kebijakan.

Untuk memperbarui grup, pengguna, atau peran untuk menggunakan **`AmazonECS_FullAccess`** kebijakan (AWS CLI)

1. Gunakan generate-service-last-accessed-detailsperintah untuk membuat laporan yang menyertakan detail tentang kapan kebijakan yang dihapus terakhir kali digunakan.

```
aws iam generate-service-last-accessed-details \
     --arn arn:aws:iam::aws:policy/AmazonEC2ContainerServiceFullAccess
```

Contoh output:

```
{
    "JobId": "32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE"
}
```

2. Gunakan ID pekerjaan dari output sebelumnya dengan get-service-last-accessed-detailsperintah untuk mengambil laporan layanan yang terakhir diakses. Laporan ini menampilkan Nama Sumber Daya Amazon (ARN) entitas IAM yang terakhir menggunakan kebijakan penghapusan bertahap.

```
aws iam get-service-last-accessed-details \
      --job-id 32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE
```

3. Gunakan salah satu perintah berikut untuk melepaskan `AmazonEC2ContainerServiceFullAccess` kebijakan dari grup, pengguna, atau peran.

   - detach-group-policy
   - detach-role-policy
   - detach-user-policy

4. Gunakan salah satu perintah berikut untuk melampirkan `AmazonECS_FullAccess` kebijakan ke grup, pengguna, atau peran.

   - attach-group-policy
   - attach-role-policy
   - attach-user-policy

Menggunakan peran terkait layanan untuk Amazon ECS

Amazon Elastic Container Service menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Amazon ECS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon ECS dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan lain AWS atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon ECS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon ECS mendefinisikan izin peran terkait layanannya, dan kecuali ditentukan lain, hanya Amazon ECS yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Amazon ECS

Amazon ECS menggunakan peran terkait layanan bernama ECS. AWSService RoleFor

AWSServiceRoleForPeran terkait layanan ECS mempercayai layanan berikut untuk mengambil peran:

- `ecs.amazonaws.com`

Kebijakan izin peran bernama Amazon ECSService RolePolicy memungkinkan Amazon ECS menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan: Saat menggunakan mode `awsvpc` jaringan untuk tugas Amazon ECS Anda, Amazon ECS mengelola siklus hidup antarmuka jaringan elastis yang terkait dengan tugas tersebut. Ini juga termasuk tag yang ditambahkan Amazon ECS ke antarmuka jaringan elastis Anda.
- Tindakan: Saat menggunakan penyeimbang beban dengan layanan Amazon ECS Anda, Amazon ECS mengelola pendaftaran dan deregistrasi sumber daya dengan penyeimbang beban.
- Tindakan: Saat menggunakan penemuan layanan Amazon ECS, Amazon ECS mengelola Route 53 dan AWS Cloud Map sumber daya yang diperlukan agar penemuan layanan berfungsi.

- Tindakan: Saat menggunakan penskalaan otomatis layanan Amazon ECS, Amazon ECS mengelola sumber daya Auto Scaling yang diperlukan.
- Tindakan: Amazon ECS membuat dan mengelola CloudWatch alarm dan aliran log yang membantu pemantauan sumber daya Amazon ECS Anda.
- Tindakan: Saat menggunakan Amazon ECS Exec, Amazon ECS mengelola izin yang diperlukan untuk memulai sesi Amazon ECS Exec ke tugas Anda.
- Tindakan: Saat menggunakan Amazon ECS Service Connect, Amazon ECS mengelola AWS Cloud Map sumber daya yang diperlukan untuk menggunakan fitur tersebut.
- Tindakan: Saat menggunakan penyedia kapasitas Amazon ECS, Amazon ECS mengelola izin yang diperlukan untuk memodifikasi grup Auto Scaling dan instans Amazon-nya. EC2
- Tindakan: Amazon ECS dapat memperbarui atribut AWS Cloud Map layanan untuk layanan yang dikelola Amazon ECS.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin peran tertaut layanan dalam Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Amazon ECS

Dalam kebanyakan kasus, Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat klaster atau membuat atau memperbarui layanan di, API AWS Management Console AWS CLI, atau AWS API, Amazon ECS akan membuat peran terkait layanan untuk Anda. Jika Anda tidak melihat peran AWSServiceRoleForECS setelah membuat klaster, lakukan hal berikut untuk memperbaiki masalah:

- Verifikasi dan konfigurasikan izin untuk mengizinkan Amazon ECS membuat, mengedit, atau menghapus peran terkait layanan atas nama Anda. Untuk informasi selengkapnya, lihat Izin peran terkait layanan dalam Panduan Pengguna IAM.
- Coba lagi operasi pembuatan klaster, atau buat peran terkait layanan secara manual.

  Anda dapat menggunakan konsol IAM untuk membuat peran terkait layanan AWSServiceRoleForECS. Di AWS CLI atau AWS API, buat peran terkait layanan dengan nama `ecs.amazonaws.com` layanan. Untuk informasi selengkapnya, silakan lihat Membuat peran terkait layanan dalam Panduan Pengguna IAM.

> **Important**
>
> Peran tertaut layanan ini dapat muncul di akun Anda jika Anda menyelesaikan tindakan di layanan lain yang menggunakan fitur yang disupport oleh peran ini.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat klaster atau membuat atau memperbarui layanan, Amazon ECS akan membuat peran terkait layanan untuk Anda lagi.

Jika Anda menghapus peran tertaut layanan ini, Anda dapat menggunakan proses IAM yang sama untuk membuat ulang peran tersebut.

Mengedit peran terkait layanan untuk Amazon ECS

Amazon ECS tidak mengizinkan Anda mengedit peran terkait layanan AWSService RoleFor ECS. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Memperbarui peran terkait layanan di Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon ECS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran yang terhubung dengan layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan Amazon ECS menggunakan peran saat Anda mencoba menghapus sumber daya, maka penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk memeriksa apakah peran yang terhubung dengan layanan memiliki sesi aktif

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Peran dan pilih nama AWSService RoleFor ECS (bukan kotak centang).
3. Di halaman Summary (Ringkasan), pilih Access Advisor (Penasihat Akses) dan tinjau aktivitas terbaru untuk peran yang terhubung dengan layanan.

> **Note**
>
> Jika Anda tidak yakin apakah Amazon ECS menggunakan peran AWSService RoleFor ECS, Anda dapat mencoba menghapus peran tersebut. Jika layanan menggunakan peran tersebut, maka penghapusan gagal dan Anda dapat melihat wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Untuk menghapus sumber daya Amazon ECS yang digunakan oleh peran terkait AWSService RoleFor layanan ECS

Anda harus menghapus semua cluster Amazon ECS di semua AWS Wilayah sebelum Anda dapat menghapus peran AWSService RoleFor ECS.

1. Skala semua layanan Amazon ECS ke hitungan 0 yang diinginkan di semua wilayah, lalu hapus layanan. Untuk informasi selengkapnya, lihat Memperbarui layanan Amazon ECS dan Menghapus layanan Amazon ECS menggunakan konsol.
2. Paksa hapus semua registrasi instans kontainer dari semua klaster di semua wilayah. Untuk informasi selengkapnya, lihat Membatalkan pendaftaran instans kontainer Amazon ECS.
3. Hapus semua cluster Amazon ECS di semua wilayah. Untuk informasi selengkapnya, lihat Menghapus kluster Amazon ECS.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran terkait layanan AWSService RoleFor ECS. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan Amazon ECS

Amazon ECS mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat Wilayah AWS dan titik akhir.

Peran IAM untuk Amazon ECS

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Di Amazon ECS, Anda dapat membuat peran untuk memberikan izin ke sumber daya Amazon ECS seperti kontainer atau layanan.

Peran yang diperlukan Amazon ECS bergantung pada jenis peluncuran definisi tugas dan fitur yang Anda gunakan. Gunakan tabel berikut untuk menentukan peran IAM mana yang Anda butuhkan untuk Amazon ECS.

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran pelaksanaan tugas	Peran ini memungkinkan Amazon ECS untuk menggunakan AWS layanan lain atas nama Anda.	Tugas Anda di-host pada AWS Fargateatau pada instans eksternal dan: • menarik gambar kontainer dari repositori pribadi Amazon ECR. • menarik gambar kontainer dari repositori pribadi Amazon ECR di akun yang berbeda dari akun yang menjalankan tugas. • mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log. Tugas Anda di-host di salah satu AWS	Peran IAM eksekusi tugas Amazon ECS

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
		Fargate atau EC2 instans Amazon dan: • menggunakan otentikasi registri pribadi. • menggunakan Runtime Monitoring. • definisi tugas mereferensikan data sensitif menggunakan rahasia Secrets Manager atau parameter AWS Systems Manager Parameter Store.	
role tugas	Peran ini memungkinkan kode aplikasi Anda (pada wadah) untuk menggunakan AWS layanan lain.	Aplikasi Anda mengakses AWS layanan lain, seperti Amazon S3.	Peran IAM tugas Amazon ECS
role instans kontainer	Peran ini memungkinkan EC2 instance atau instans eksternal Anda untuk mendaftar dengan cluster.	Tugas Anda di-host di EC2 instans Amazon atau instans eksternal .	Peran IAM instance wadah Amazon ECS

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran Amazon ECS Anywhere	Peran ini memungkinkan instans eksternal Anda untuk mengakses AWS APIs.	Tugas Anda di-host pada instance eksternal.	Peran IAM Amazon ECS Anywhere
Infrastruktur Amazon ECS untuk peran penyeimbang beban	Peran ini memungkinkan Amazon ECS mengelola sumber daya penyeimbang beban di klaster Anda atas nama Anda untuk penerapan. blue/green	Anda ingin menggunakan blue/green penerapan Amazon ECS.	Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban
Peran Amazon ECS CodeDeploy	Peran ini memungkinkan CodeDeploy untuk membuat pembaruan pada layanan Anda.	Anda menggunakan jenis penyebaran CodeDeploy biru/hijau untuk menyebarkan layanan.	Peran Amazon ECS CodeDeploy IAM
Peran Amazon ECS EventBridge	Peran ini memungkinkan EventBridge untuk membuat pembaruan pada layanan Anda.	Anda menggunakan EventBridge aturan dan target untuk menjadwalkan tugas Anda.	Peran Amazon ECS EventBridge IAM

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran infrastruktur Amazon ECS	Peran ini memungkinkan Amazon ECS mengelola sumber daya infrastruktur di cluster Anda.	• Anda ingin melampirkan volume Amazon EBS ke Fargate Anda EC2 atau meluncurkan tugas Amazon ECS jenis. Peran infrastruktur memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas Anda. • Anda ingin menggunakan Transport Layer Security (TLS) untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda. • Anda ingin membuat grup target VPC Lattice.	Peran IAM infrastruktur Amazon ECS

Praktik terbaik untuk peran IAM di Amazon ECS

Peran yang diperlukan Amazon ECS bergantung pada jenis peluncuran definisi tugas dan fitur yang Anda gunakan. Kami menyarankan Anda membuat peran terpisah dalam tabel alih-alih berbagi peran.

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran pelaksanaan tugas	Peran ini memungkinkan Amazon ECS untuk menggunakan AWS layanan lain atas nama Anda.	Tugas Anda di-host pada AWS Fargateatau pada instans eksternal dan: • menarik gambar kontainer dari repositori pribadi Amazon ECR. • menarik gambar kontainer dari repositori pribadi Amazon ECR di akun yang berbeda dari akun yang menjalankan tugas. • mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log. Tugas Anda di-host di salah satu AWS Fargate atau EC2 instans Amazon dan: • menggunakan otentikasi registri pribadi. • menggunakan Runtime Monitoring.	Peran IAM eksekusi tugas Amazon ECS

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
		• definisi tugas mereferensikan data sensitif menggunakan rahasia Secrets Manager atau parameter AWS Systems Manager Parameter Store.	
role tugas	Peran ini memungkinkan kode aplikasi Anda (pada wadah) untuk menggunakan AWS layanan lain.	Aplikasi Anda mengakses AWS layanan lain, seperti Amazon S3.	Peran IAM tugas Amazon ECS
role instans kontainer	Peran ini memungkinkan EC2 instance atau instans eksternal Anda untuk mendaftar dengan cluster.	Tugas Anda di-host di EC2 instans Amazon atau instans eksternal.	Peran IAM instance wadah Amazon ECS
Peran Amazon ECS Anywhere	Peran ini memungkinkan instans eksternal Anda untuk mengakses AWS APIs.	Tugas Anda di-host pada instance eksternal.	Peran IAM Amazon ECS Anywhere
Peran Amazon ECS CodeDeploy	Peran ini memungkinkan CodeDeploy untuk membuat pembaruan pada layanan Anda.	Anda menggunakan jenis penyebaran CodeDeploy biru/hijau untuk menyebarkan layanan.	Peran Amazon ECS CodeDeploy IAM

Peran	Definisi	Saat dibutuhkan	Informasi selengkapnya
Peran Amazon ECS EventBridge	Peran ini memungkinkan EventBridge untuk membuat pembaruan pada layanan Anda.	Anda menggunakan EventBridge aturan dan target untuk menjadwalkan tugas Anda.	Peran Amazon ECS EventBridge IAM
Peran infrastruktur Amazon ECS	Peran ini memungkinkan Amazon ECS mengelola sumber daya infrastruktur di cluster Anda.	• Anda ingin melampirkan volume Amazon EBS ke Fargate Anda EC2 atau meluncurkan tugas Amazon ECS jenis. Peran infrastruktur memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas Anda. • Anda ingin menggunakan Transport Layer Security (TLS) untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda. • Anda ingin membuat grup target VPC Lattice.	Peran IAM infrastruktur Amazon ECS

role tugas

Kami menyarankan Anda menetapkan peran tugas. Perannya dapat dibedakan dari peran EC2 instance Amazon yang sedang dijalankannya. Menetapkan setiap tugas peran selaras dengan prinsip akses yang paling tidak memiliki hak istimewa dan memungkinkan kontrol terperinci yang lebih besar atas tindakan dan sumber daya.

Saat Anda menambahkan peran tugas ke definisi tugas, agen penampung Amazon ECS secara otomatis membuat token dengan ID kredensyal unik (misalnya,`12345678-90ab-cdef-1234-567890abcdef`) untuk tugas tersebut. Token ini dan kredenal peran kemudian ditambahkan ke cache internal agen. Agen mengisi variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` dalam wadah dengan URI ID kredensyal (misalnya,`/v2/credentials/12345678-90ab-cdef-1234-567890abcdef`).

Anda dapat mengambil kredenal peran sementara secara manual dari dalam wadah dengan menambahkan variabel lingkungan ke alamat IP agen penampung Amazon ECS dan menjalankan `curl` perintah pada string yang dihasilkan.

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

Output yang diharapkan adalah sebagai berikut:

```
{
 "RoleArn": "arn:aws:iam::123456789012:role/SSMTaskRole-SSMFargateTaskIAMRole-
DASWWSF2WGD6",
 "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
 "Token": "IQoJb3JpZ2luX2VjEEM/Example==",
 "Expiration": "2021-01-16T00:51:53Z"
}
```

Versi yang lebih baru secara AWS SDKs otomatis mengambil kredensyal ini dari variabel `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` lingkungan saat melakukan panggilan API. AWS Untuk informasi tentang cara memperbarui kredensil, lihat Memperbarui kredensil di RePost AWS.

Outputnya mencakup pasangan kunci akses yang terdiri dari ID kunci akses rahasia dan kunci rahasia yang digunakan aplikasi Anda untuk mengakses AWS sumber daya. Ini juga termasuk token yang AWS digunakan untuk memverifikasi bahwa kredensialnya valid. Secara default, kredenal yang ditetapkan ke tugas menggunakan peran tugas valid selama enam jam. Setelah itu, mereka secara otomatis diputar oleh agen kontainer Amazon ECS.

Peran pelaksanaan tugas

Peran eksekusi tugas digunakan untuk memberikan izin agen penampung Amazon ECS untuk memanggil tindakan AWS API tertentu atas nama Anda. Misalnya, saat Anda menggunakan AWS Fargate, Fargate memerlukan peran IAM yang memungkinkannya menarik gambar dari Amazon ECR dan menulis log ke Log. CloudWatch Peran IAM juga diperlukan saat tugas mereferensikan rahasia yang disimpan AWS Secrets Manager, seperti rahasia tarik gambar.

> **Note**
>
> Jika Anda menarik gambar sebagai pengguna yang diautentikasi, Anda cenderung tidak terpengaruh oleh perubahan yang terjadi pada batas kecepatan tarik Docker Hub. Untuk informasi selengkapnya lihat, Autentikasi registri pribadi untuk instance kontainer. Dengan menggunakan Amazon ECR dan Amazon ECR Public, Anda dapat menghindari batasan yang diberlakukan oleh Docker. Jika Anda menarik gambar dari Amazon ECR, ini juga membantu mempersingkat waktu tarik jaringan dan mengurangi perubahan transfer data saat lalu lintas meninggalkan VPC Anda.

> **Important**
>
> Saat Anda menggunakan Fargate, Anda harus mengautentikasi ke registri gambar pribadi menggunakan. `repositoryCredentials` Tidak mungkin menyetel variabel lingkungan agen penampung Amazon ECS `ECS_ENGINE_AUTH_TYPE` `ECS_ENGINE_AUTH_DATA` atau memodifikasi `ecs.config` file untuk tugas yang dihosting di Fargate. Untuk informasi selengkapnya, lihat Autentikasi registri pribadi untuk tugas.

role instans kontainer

Kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola mencakup izin berikut. Dengan mengikuti saran keamanan standar untuk memberikan hak istimewa paling rendah, kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` dapat digunakan sebagai panduan. Jika salah satu izin yang diberikan dalam kebijakan terkelola tidak diperlukan untuk kasus penggunaan Anda, buat kebijakan kustom dan hanya tambahkan izin yang Anda perlukan.

- `ec2:DescribeTags`— (Opsional) Memungkinkan prinsipal untuk menggambarkan tag yang terkait dengan EC2 instance Amazon. Izin ini digunakan oleh agen penampung Amazon ECS untuk

mendukung propagasi tag sumber daya. Untuk informasi selengkapnya, lihat Bagaimana sumber daya ditandai.

- `ecs:CreateCluster`— (Opsional) Memungkinkan prinsipal untuk membuat cluster Amazon ECS. Izin ini digunakan oleh agen penampung Amazon ECS untuk membuat `default` klaster, jika belum ada.
- `ecs:DeregisterContainerInstance`— (Opsional) Memungkinkan prinsipal untuk membatalkan pendaftaran instans penampung Amazon ECS dari cluster. Agen penampung Amazon ECS tidak memanggil operasi API ini, tetapi izin ini tetap untuk membantu memastikan kompatibilitas mundur.
- `ecs:DiscoverPollEndpoint`— (Diperlukan) Tindakan ini mengembalikan titik akhir yang digunakan agen penampung Amazon ECS untuk melakukan polling untuk pembaruan.
- `ecs:Poll`— (Wajib) Memungkinkan agen penampung Amazon ECS berkomunikasi dengan bidang kontrol Amazon ECS untuk melaporkan perubahan status tugas.
- `ecs:RegisterContainerInstance`— (Wajib) Memungkinkan prinsipal untuk mendaftarkan instance kontainer dengan cluster. Izin ini digunakan oleh agen penampung Amazon ECS untuk mendaftarkan EC2 instans Amazon dengan klaster dan untuk mendukung propagasi tag sumber daya.
- `ecs:StartTelemetrySession`— (Opsional) Memungkinkan agen kontainer Amazon ECS berkomunikasi dengan pesawat kontrol Amazon ECS untuk melaporkan informasi kesehatan dan metrik untuk setiap wadah dan tugas.

  Meskipun izin ini tidak diperlukan, kami menyarankan Anda menambahkannya untuk mengizinkan metrik instance container untuk memulai tindakan skala dan juga menerima laporan yang terkait dengan perintah pemeriksaan kesehatan.
- `ecs:TagResource`— (Opsional) Memungkinkan agen penampung Amazon ECS menandai klaster saat pembuatan dan menandai instance kontainer saat terdaftar ke klaster.
- `ecs:UpdateContainerInstancesState`— Memungkinkan kepala sekolah untuk memodifikasi status instans penampung Amazon ECS. Izin ini digunakan oleh agen kontainer Amazon ECS untuk pengeringan Instans Spot.
- `ecs:Submit*`— (Wajib) Ini termasuk tindakan `SubmitAttachmentStateChangesSubmitContainerStateChange`,, dan `SubmitTaskStateChange` API. Mereka digunakan oleh agen penampung Amazon ECS untuk melaporkan perubahan status untuk setiap sumber daya ke bidang kontrol Amazon ECS. `SubmitContainerStateChange`Izin tersebut tidak lagi digunakan oleh agen penampung Amazon ECS tetapi tetap untuk membantu memastikan kompatibilitas mundur.

- `ecr:GetAuthorizationToken`— (Opsional) Memungkinkan kepala sekolah untuk mengambil token otorisasi. Token otorisasi mewakili kredensi otentikasi IAM Anda dan dapat digunakan untuk mengakses registri ECR Amazon apa pun yang dapat diakses oleh prinsipal IAM. Token otorisasi yang diterima berlaku selama 12 jam.
- `ecr:BatchCheckLayerAvailability`— (Opsional) Saat gambar kontainer didorong ke repositori pribadi Amazon ECR, setiap lapisan gambar diperiksa untuk memverifikasi apakah sudah didorong. Jika sudah didorong, maka lapisan citra dilewati.
- `ecr:GetDownloadUrlForLayer`— (Opsional) Ketika gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk setiap lapisan gambar yang belum di-cache.
- `ecr:BatchGetImage`— (Opsional) Saat gambar kontainer ditarik dari repositori pribadi Amazon ECR, API ini dipanggil sekali untuk mengambil manifes gambar.
- `logs:CreateLogStream`— (Opsional) Memungkinkan prinsipal untuk membuat aliran CloudWatch log Log untuk grup log tertentu.
- `logs:PutLogEvents`— (Opsional) Memungkinkan prinsipal untuk mengunggah sekumpulan peristiwa log ke aliran log tertentu.

Peran terkait layanan

Anda dapat menggunakan peran terkait layanan untuk Amazon ECS untuk memberikan izin layanan Amazon ECS untuk memanggil layanan APIs lain atas nama Anda. Amazon ECS memerlukan izin untuk membuat dan menghapus antarmuka jaringan, mendaftar, dan membatalkan pendaftaran target dengan grup target. Ini juga membutuhkan izin yang diperlukan untuk membuat dan menghapus kebijakan penskalaan. Izin ini diberikan melalui peran terkait layanan. Peran ini dibuat atas nama Anda saat pertama kali Anda menggunakan layanan.

> **Note**
>
> Jika Anda secara tidak sengaja menghapus peran terkait layanan, Anda dapat membuatnya kembali. Untuk petunjuknya, lihat Membuat peran terkait layanan.

Rekomendasi peran

Kami menyarankan Anda melakukan hal berikut saat menyiapkan peran dan kebijakan IAM tugas Anda.

Blokir akses ke EC2 metadata Amazon

Saat menjalankan tugas di EC2 instans Amazon, kami sangat menyarankan Anda memblokir akses ke EC2 metadata Amazon untuk mencegah container mewarisi peran yang ditetapkan ke instance tersebut. Jika aplikasi Anda harus memanggil tindakan AWS API, gunakan peran IAM untuk tugas sebagai gantinya.

Untuk mencegah tugas yang berjalan dalam mode bridge mengakses EC2 metadata Amazon, jalankan perintah berikut atau perbarui data pengguna instans. Untuk instruksi selengkapnya tentang memperbarui data pengguna sebuah instans, lihat Artikel AWS Support ini. Untuk informasi selengkapnya tentang mode jembatan definisi tugas, lihat mode jaringan definisi tugas.

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 169.254.169.254/32 --jump DROP
```

Agar perubahan ini tetap ada setelah reboot, jalankan perintah berikut yang khusus untuk Amazon Machine Image (AMI) Anda:

- Amazon Linux 2

  ```
  sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
   iptables
  ```

- Amazon Linux

  ```
  sudo service iptables save
  ```

Untuk tugas yang menggunakan mode `awsvpc` jaringan, atur variabel lingkungan `ECS_AWSVPC_BLOCK_IMDS` ke `true` dalam `/etc/ecs/ecs.config` file.

Anda harus mengatur `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` variabel ke `false` dalam `ecs-agent config` file untuk mencegah kontainer yang berjalan di dalam `host` jaringan mengakses EC2 metadata Amazon.

Gunakan mode **`awsvpc`** jaringan

Gunakan mode jaringan `awsvpc` jaringan untuk membatasi arus lalu lintas antara tugas yang berbeda atau antara tugas Anda dan layanan lain yang berjalan dalam VPC Amazon Anda. Ini menambahkan lapisan keamanan tambahan. Mode `awsvpc` jaringan menyediakan isolasi jaringan

tingkat tugas untuk tugas yang berjalan di Amazon. EC2 Ini adalah mode default AWS Fargate aktif. itu satu-satunya mode jaringan yang dapat Anda gunakan untuk menetapkan grup keamanan untuk tugas.

Gunakan informasi yang terakhir diakses untuk menyempurnakan peran

Kami menyarankan Anda menghapus tindakan apa pun yang tidak pernah digunakan atau belum digunakan selama beberapa waktu. Ini mencegah akses yang tidak diinginkan terjadi. Untuk melakukan ini, tinjau informasi terakhir yang diakses yang disediakan oleh IAM, lalu hapus tindakan yang tidak pernah digunakan atau belum digunakan baru-baru ini. Anda dapat melakukan ini dengan mengikuti langkah-langkah berikut.

Jalankan perintah berikut untuk menghasilkan laporan yang menampilkan informasi akses terakhir untuk kebijakan yang direferensikan:

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

gunakan `JobId` yang ada di output untuk menjalankan perintah berikut. Setelah Anda melakukan ini, Anda dapat melihat hasil laporan.

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

Untuk informasi selengkapnya, lihat Memperbaiki izin AWS menggunakan informasi yang terakhir diakses.

Pantau AWS CloudTrail aktivitas mencurigakan

Anda dapat memantau AWS CloudTrail aktivitas mencurigakan apa pun. Sebagian besar panggilan AWS API dicatat AWS CloudTrail sebagai peristiwa. Mereka dianalisis oleh AWS CloudTrail Insights, dan Anda diberi tahu tentang perilaku mencurigakan apa pun yang terkait dengan `write` panggilan API. Ini mungkin termasuk lonjakan volume panggilan. Peringatan ini mencakup informasi seperti waktu aktivitas yang tidak biasa terjadi dan identitas teratas ARN yang berkontribusi pada. APIs

Anda dapat mengidentifikasi tindakan yang dilakukan oleh tugas dengan peran IAM AWS CloudTrail dengan melihat `userIdentity` properti acara. Dalam contoh berikut, `arn` termasuk nama peran yang diasumsikan,`s3-write-go-bucket-role`, diikuti dengan nama tugas,`7e9894e088ad416eb5cab92afExample`.

```
"userIdentity": {
```

```
    "type": "AssumedRole",
    "principalId": "AROA36C6WWEJ2YEXAMPLE:7e9894e088ad416eb5cab92afExample",
    "arn": "arn:aws:sts::123456789012:assumed-role/s3-write-go-bucket-
role/7e9894e088ad416eb5cab92afExample",
    ...
}
```

> **Note**
>
> Saat tugas yang mengambil peran dijalankan di instans EC2 penampung Amazon, permintaan dicatat oleh agen penampung Amazon ECS ke log audit agen yang terletak di alamat dalam format. `/var/log/ecs/audit.log.YYYY-MM-DD-HH` Untuk informasi selengkapnya, lihat Log Peran IAM Tugas dan Logging Insights Events for Trails.

Peran IAM eksekusi tugas Amazon ECS

Peran eksekusi tugas memberikan izin kepada kontainer Amazon ECS dan agen Fargate untuk melakukan panggilan AWS API atas nama Anda. IAM role eksekusi tugas diperlukan sesuai dengan persyaratan tugas Anda. Anda dapat memiliki beberapa peran eksekusi tugas untuk berbagai tujuan dan layanan yang terkait dengan akun Anda.

> **Note**
>
> Izin ini tersedia untuk agen yang berjalan di instans Anda oleh Amazon ECS yang secara berkala mengirimkan kredensi sementara peran tersebut, tetapi tidak dapat diakses secara langsung oleh kontainer dalam tugas. Untuk izin IAM yang perlu dijalankan oleh kode aplikasi Anda di dalam container, lihat. Peran IAM tugas Amazon ECS

Berikut ini adalah kasus penggunaan umum untuk peran IAM eksekusi tugas:

- Tugas Anda di-host pada AWS Fargateatau pada instans eksternal dan:
  - menarik gambar kontainer dari repositori pribadi Amazon ECR.
  - menarik gambar kontainer dari repositori pribadi Amazon ECR di akun yang berbeda dari akun yang menjalankan tugas.
  - mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke CloudWatch .

- Tugas Anda di-host di salah satu AWS Fargate atau EC2 instans Amazon dan:
  - menggunakan otentikasi registri pribadi. Untuk informasi selengkapnya, lihat Izin otentikasi registri pribadi.
  - menggunakan Runtime Monitoring.
  - definisi tugas mereferensikan data sensitif menggunakan rahasia Secrets Manager atau parameter AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Izin Secrets Manager atau Systems Manager.

> **Note**
>
> Peran eksekusi tugas didukung oleh agen penampung Amazon ECS versi 1.16.0 dan yang lebih baru.

Amazon ECS menyediakan kebijakan terkelola bernama `AmazonECSTaskExecutionRolePolicy` yang berisi izin yang diperlukan oleh kasus penggunaan umum yang dijelaskan di atas. Untuk informasi selengkapnya, lihat Amazon ECSTask ExecutionRolePolicy di Panduan Referensi Kebijakan AWS Terkelola. Mungkin perlu menambahkan kebijakan sebaris ke peran eksekusi tugas Anda untuk kasus penggunaan khusus

Konsol Amazon ECS membuat peran eksekusi tugas. Anda dapat secara manual melampirkan kebijakan IAM terkelola untuk tugas agar Amazon ECS dapat menambahkan izin untuk fitur dan penyempurnaan future saat diperkenalkan. Anda dapat menggunakan pencarian konsol IAM untuk mencari `ecsTaskExecutionRole` dan melihat apakah akun Anda sudah memiliki peran eksekusi tugas. Untuk informasi selengkapnya, lihat pencarian konsol IAM di panduan pengguna IAM.

Jika Anda menarik gambar sebagai pengguna yang diautentikasi, Anda cenderung tidak terpengaruh oleh perubahan yang terjadi pada Docker Hubpenggunaan dan batasan. Untuk informasi selengkapnya lihat, Autentikasi registri pribadi untuk instance kontainer.

Dengan menggunakan Amazon ECR dan Amazon ECR Public, Anda dapat menghindari batasan yang diberlakukan oleh Docker. Jika Anda menarik gambar dari Amazon ECR, ini juga membantu mempersingkat waktu tarik jaringan dan mengurangi perubahan transfer data saat lalu lintas meninggalkan VPC Anda.

Saat Anda menggunakan Fargate, Anda harus mengautentikasi ke registri gambar pribadi menggunakan. `repositoryCredentials` Tidak mungkin menyetel variabel lingkungan agen

penampung Amazon ECS `ECS_ENGINE_AUTH_TYPE` `ECS_ENGINE_AUTH_DATA` atau memodifikasi `ecs.config` file untuk tugas yang dihosting di Fargate. Untuk informasi selengkapnya, lihat Autentikasi registri pribadi untuk tugas.

Membuat peran eksekusi tugas

Jika akun Anda belum memiliki peran eksekusi tugas, gunakan langkah-langkah berikut untuk membuat peran.

AWS Management Console

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.
6. Di bagian Tambahkan izin, cari Amazon ECSTask ExecutionRolePolicy, lalu pilih kebijakan.
7. Pilih Berikutnya.
8. Untuk nama Peran, masukkan ecsTaskExecutionPeran.
9. Tinjau peran lalu pilih Buat peran.

AWS CLI

Ganti semua *`user input`* dengan informasi Anda sendiri.

1. Buat file bernama `ecs-tasks-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Buat peran IAM bernama `ecsTaskExecutionRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsTaskExecutionRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. Lampirkan `AmazonECSTaskExecutionRolePolicy` kebijakan yang AWS dikelola ke `ecsTaskExecutionRole` peran.

```
aws iam attach-role-policy \
      --role-name ecsTaskExecutionRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

Setelah Anda membuat peran, tambahkan izin tambahan ke peran untuk fitur berikut.

Fitur	Izin tambahan
Tarik gambar kontainer dari pendaftar pribadi di luar AWS (seperti Docker Hub, Quay.io, atau registri pribadi Anda sendiri) menggunakan kredensil Secrets Manager	Izin otentikasi registri pribadi
Lulus data sensitif dengan Systems Manager atau Secrets Manager	Izin Secrets Manager atau Systems Manager
Minta tugas Fargate menarik gambar Amazon ECR melalui titik akhir antarmuka	Tugas Fargate menarik gambar Amazon ECR melalui izin titik akhir antarmuka
Host file konfigurasi dalam bucket Amazon S3	Izin penyimpanan file Amazon S3

Fitur	Izin tambahan
Konfigurasikan Wawasan Kontainer untuk melihat peristiwa siklus hidup Amazon ECS	Izin diperlukan untuk mengaktifkan peristiwa siklus hidup Amazon ECS dalam Container Insights
Melihat peristiwa siklus hidup Amazon ECS di Wawasan Kontainer	Izin diperlukan untuk melihat peristiwa siklus hidup Amazon ECS di Wawasan Kontainer

Izin otentikasi registri pribadi

Autentikasi registri pribadi memungkinkan tugas Amazon ECS Anda menarik gambar kontainer dari pendaftar pribadi di luar AWS (seperti Docker Hub, Quay.io, atau registri pribadi Anda sendiri) yang memerlukan kredensi otentikasi. Fitur ini menggunakan Secrets Manager untuk menyimpan kredenal registri Anda dengan aman, yang kemudian direferensikan dalam definisi tugas Anda menggunakan parameter. `repositoryCredentials`

Untuk informasi selengkapnya tentang mengonfigurasi autentikasi registri pribadi, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.

Untuk menyediakan akses ke rahasia yang berisi kredensil registri pribadi Anda, tambahkan izin berikut sebagai kebijakan inline ke peran eksekusi tugas. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- `secretsmanager:GetSecretValue`—Diperlukan untuk mengambil kredensi registri pribadi dari Secrets Manager.
- `kms:Decrypt`—Wajib hanya jika rahasia Anda menggunakan kunci KMS khusus dan bukan kunci default. Nama Sumber Daya Amazon (ARN) untuk kunci kustom Anda harus ditambahkan sebagai sumber daya.

Berikut ini adalah contoh kebijakan inline yang menambahkan izin.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

Izin Secrets Manager atau Systems Manager

Izin untuk mengizinkan agen kontainer menarik sumber daya Secrets Manager yang diperlukan AWS Systems Manager atau Secrets. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.

Menggunakan Secrets Manager

Untuk memberikan akses ke rahasia Secrets Manager yang Anda buat, tambahkan izin berikut secara manual ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.

Contoh kebijakan berikut menambahkan izin yang diperlukan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
}
```

Menggunakan Systems Manager

> **Important**
>
> Untuk tugas yang menggunakan tipe EC2 peluncuran, Anda harus menggunakan variabel konfigurasi agen ECS `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` untuk menggunakan fitur ini. Anda dapat menambahkannya ke file `./etc/ecs/ecs.config` selama pembuatan instans kontainer atau Anda dapat menambahkannya ke instans yang ada, lalu memulai ulang agen ECS. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Untuk memberikan akses ke parameter Penyimpanan Parameter Systems Manager yang Anda buat, tambahkan izin berikut secara manual sebagai kebijakan ke peran eksekusi tugas. Untuk informasi tentang cara mengelola izin, lihat Menambahkan dan Menghapus izin identitas IAM di Panduan Pengguna IAM.

- `ssm:GetParameters`— Diperlukan jika Anda mereferensikan parameter Systems Manager Parameter Store dalam definisi tugas. Menambahkan izin untuk mengambil parameter Systems Manager.
- `secretsmanager:GetSecretValue`— Diperlukan jika Anda mereferensikan rahasia Secrets Manager baik secara langsung atau jika parameter Parameter Store Systems Manager Anda mereferensikan rahasia Secrets Manager dalam definisi tugas. Menambahkan izin untuk mengambil rahasia dari Secrets Manager.
- `kms:Decrypt`— Diperlukan hanya jika rahasia Anda menggunakan kunci yang dikelola pelanggan dan bukan kunci default. ARN untuk kunci khusus Anda harus ditambahkan sebagai sumber daya. Menambahkan izin untuk mendekripsi kunci yang dikelola pelanggan.

Contoh kebijakan berikut menambahkan izin yang diperlukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
```

```
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

Tugas Fargate menarik gambar Amazon ECR melalui izin titik akhir antarmuka

Saat meluncurkan tugas yang menggunakan tipe peluncuran Fargate yang menarik gambar dari Amazon ECR saat Amazon ECR dikonfigurasi untuk menggunakan titik akhir VPC antarmuka, Anda dapat membatasi akses tugas ke titik akhir VPC atau VPC tertentu. Lakukan ini dengan membuat peran eksekusi tugas untuk tugas yang akan digunakan yang menggunakan kunci kondisi IAM.

Gunakan kunci kondisi global IAM berikut untuk membatasi akses ke titik akhir VPC atau VPC tertentu. Untuk informasi selengkapnya, lihat Kunci Konteks Syarat Global AWS.

- `aws:SourceVpc`—Membatasi akses ke VPC tertentu. Anda dapat membatasi VPC ke VPC yang menampung tugas dan titik akhir.
- `aws:SourceVpce`—Membatasi akses ke titik akhir VPC tertentu.

Kebijakan peran eksekusi tugas berikut memberikan contoh untuk menambahkan kunci syarat:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "arn:aws:ec2:arn:aws::111122223333:vpc-
endpoint/vpce-0123456789abcdef0",
                    "arn:aws:ec2:arn:aws::111122223333:vpc/vpc-0123456789abcdef0"
                }
            }
        }
    ]
}
```

Izin Amazon ECR

Izin berikut diperlukan saat Anda perlu menarik gambar kontainer dari repositori pribadi Amazon ECR. Peran eksekusi tugas harus memiliki izin ini untuk memungkinkan penampung Amazon ECS dan agen Fargate menarik gambar kontainer atas nama Anda. Untuk implementasi ECS dasar, izin ini harus ditambahkan ke peran eksekusi tugas daripada peran IAM tugas.

Kebijakan terkelola peran eksekusi tugas Amazon ECS (`AmazonECSTaskExecutionRolePolicy`) menyertakan izin yang diperlukan untuk menarik gambar dari Amazon ECR. Jika menggunakan kebijakan terkelola, Anda tidak perlu menambahkan izin ini secara terpisah.

Jika Anda membuat kebijakan khusus, sertakan izin berikut untuk mengizinkan pengambilan gambar dari Amazon ECR:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Perhatikan bahwa izin ini berbeda dari izin yang mungkin diperlukan dalam peran IAM tugas jika kode aplikasi Anda perlu berinteraksi dengan Amazon ECR secara langsung. APIs Untuk informasi tentang izin peran IAM tugas untuk Amazon ECR, lihat. Izin Amazon ECR

Izin penyimpanan file Amazon S3

Saat Anda menentukan file konfigurasi yang di-host di Amazon S3, peran eksekusi tugas harus menyertakan `s3:GetObject` izin untuk file konfigurasi dan `s3:GetBucketLocation` izin di bucket Amazon S3 tempat file tersebut berada. Untuk informasi selengkapnya, lihat Tindakan kebijakan untuk Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Kebijakan contoh berikut menambahkan izin yang diperlukan untuk mengambil file dari Amazon S3. Tentukan nama bucket Amazon S3 dan nama file konfigurasi Anda.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    }
  ]
}
```

Peran IAM tugas Amazon ECS

Tugas Amazon ECS Anda dapat memiliki peran IAM yang terkait dengannya. Izin yang diberikan dalam peran IAM dijual ke kontainer yang berjalan dalam tugas. Peran ini memungkinkan kode aplikasi Anda (berjalan dalam wadah) untuk menggunakan AWS layanan lain. Peran tugas diperlukan saat aplikasi Anda mengakses AWS layanan lain, seperti Amazon S3.

> **Note**
>
> Izin ini tidak diakses oleh penampung Amazon ECS dan agen Fargate. Untuk izin IAM yang dibutuhkan Amazon ECS untuk menarik gambar kontainer dan menjalankan tugas, lihat. Peran IAM eksekusi tugas Amazon ECS

Berikut ini adalah manfaat menggunakan peran tugas:

- Pemisahan masalah: Jika Anda menggunakan jenis EC2 peluncuran, peran IAM tugas memungkinkan Anda menentukan izin IAM untuk kontainer Anda tanpa memerlukan izin ini ditentukan menggunakan profil EC2 instans (untuk informasi selengkapnya, lihat Menggunakan profil instans di Panduan Pengguna).AWS Identity and Access Management Oleh karena itu, Anda dapat menerapkan aplikasi Anda secara independen dan seragam pada instance kontainer ECS tanpa perlu mengubah izin IAM yang terkait dengan instance. EC2
- Auditabilitas: Akses dan pencatatan peristiwa tersedia CloudTrail untuk memastikan audit retrospektif. Kredensyal tugas memiliki konteks '`taskArn`' yang dilampirkan ke sesi, jadi CloudTrail log menunjukkan tugas mana yang digunakan untuk kredensyal peran.
- Pengiriman kredensil seragam: ECS memberikan kredensil peran IAM ke kontainer Anda dan membuatnya dapat diakses melalui antarmuka yang terdefinisi dengan baik terlepas dari jenis peluncuran yang terkait dengan tugas Anda. Di ECS Fargate EC2 , profil instans tidak tersedia untuk kontainer dalam tugas Anda. Peran IAM tugas memungkinkan Anda mengaitkan izin IAM

ke kontainer terlepas dari jenis peluncuran saat Anda menggunakan SDK AWS atau CLI AWS di container Anda. Untuk informasi selengkapnya tentang cara AWS SDK mengakses kredensil ini, lihat Penyedia kredensi kontainer.

Important

Kontainer bukan batas keamanan dan penggunaan peran tugas IAM tidak mengubah ini. Setiap tugas yang berjalan di Fargate memiliki batas isolasi sendiri dan tidak berbagi kernel yang mendasarinya, sumber daya CPU, sumber daya memori, atau elastic network interface dengan tugas lain. Untuk EC2 dan Instans Kontainer Eksternal di ECS, tidak ada isolasi tugas (tidak seperti Fargate) dan kontainer berpotensi mengakses kredensyal untuk tugas lain pada instance kontainer yang sama. Mereka juga dapat mengakses izin yang ditetapkan ke peran instance kontainer ECS. Ikuti rekomendasi dalam rekomendasi Peran untuk memblokir akses ke Layanan Metadata EC2 Instans Amazon untuk container (Untuk informasi selengkapnya, lihat Menggunakan Layanan Metadata Instance untuk mengakses metadata instans di Panduan Pengguna Amazon). EC2
Perhatikan bahwa ketika Anda menentukan peran IAM untuk tugas, AWS CLI atau SDKs lainnya dalam wadah untuk tugas tersebut menggunakan AWS kredenal yang disediakan oleh peran tugas secara eksklusif dan mereka tidak mewarisi izin IAM apa pun dari EC2 Amazon atau instance eksternal tempat mereka berjalan.

Membuat peran tugas IAM

Saat membuat kebijakan IAM untuk tugas yang akan digunakan, kebijakan tersebut harus menyertakan izin yang Anda inginkan untuk diasumsikan oleh container dalam tugas Anda. Anda dapat menggunakan kebijakan AWS terkelola yang ada, atau Anda dapat membuat kebijakan khusus dari awal yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Important

Untuk tugas Amazon ECS (untuk semua jenis peluncuran), sebaiknya gunakan kebijakan dan peran IAM untuk tugas Anda. Kredensil ini memungkinkan tugas Anda untuk membuat permintaan AWS API tanpa memanggil `sts:AssumeRole` untuk mengambil peran yang sama yang sudah dikaitkan dengan tugas. Jika tugas Anda mengharuskan peran mengasumsikan dirinya sendiri, Anda harus membuat kebijakan kepercayaan yang secara

eksplisit memungkinkan peran itu untuk mengambil alih dirinya sendiri. Untuk informasi selengkapnya, lihat Memperbarui kebijakan kepercayaan peran di Panduan Pengguna IAM.

Setelah kebijakan IAM dibuat, Anda dapat membuat peran IAM yang menyertakan kebijakan yang Anda referensikan dalam definisi tugas Amazon ECS Anda. Anda dapat membuat peran menggunakan kasus penggunaan Elastic Container Service Task di konsol IAM. Kemudian, Anda dapat melampirkan kebijakan IAM spesifik Anda ke peran yang memberikan kontainer dalam tugas Anda izin yang Anda inginkan. Prosedur di bawah menjelaskan cara melakukannya.

Jika Anda memiliki beberapa definisi tugas atau layanan yang memerlukan izin IAM, Anda harus mempertimbangkan untuk membuat peran untuk setiap definisi tugas atau layanan tertentu dengan izin minimum yang diperlukan untuk tugas yang akan dioperasikan sehingga Anda dapat meminimalkan akses yang Anda berikan untuk setiap tugas.

Untuk informasi tentang titik akhir layanan untuk Wilayah Anda, lihat Titik akhir layanan di Panduan.Referensi Umum Amazon Web

Peran tugas IAM harus memiliki kebijakan kepercayaan yang menentukan layanan. `ecs-tasks.amazonaws.com` `sts:AssumeRole`Izin tersebut memungkinkan tugas Anda untuk mengambil peran IAM yang berbeda dari yang digunakan EC2 instans Amazon. Dengan cara ini, tugas Anda tidak mewarisi peran yang terkait dengan EC2 instans Amazon. Berikut ini adalah contoh kebijakan kepercayaan. Ganti pengenal Wilayah dan tentukan nomor AWS akun yang Anda gunakan saat meluncurkan tugas.

Important

Saat membuat peran IAM tugas Anda, Anda disarankan untuk menggunakan `aws:SourceAccount` atau kunci `aws:SourceArn` kondisi dalam kebijakan hubungan kepercayaan yang terkait dengan peran untuk cakupan izin lebih lanjut untuk mencegah masalah keamanan wakil yang membingungkan. Menggunakan tombol `aws:SourceArn` kondisi untuk menentukan klaster tertentu saat ini tidak didukung, Anda harus menggunakan wildcard untuk menentukan semua cluster. Untuk mempelajari lebih lanjut tentang masalah deputi yang membingungkan dan cara melindungi AWS akun Anda, lihat Masalah wakil yang bingung di Panduan Pengguna IAM.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

Gunakan prosedur berikut untuk membuat kebijakan untuk mengambil objek dari Amazon S3 dengan kebijakan contoh. Ganti semua *user input* dengan nilai Anda sendiri.

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.

4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ]
      }
   ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

AWS CLI

Ganti semua *user input* dengan nilai Anda sendiri.

1. Buat file bernama `s3-policy.json` dengan konten berikut.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ]
      }
   ]
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON. Ganti semua *user input* dengan nilai Anda sendiri.

```
aws iam create-policy \
      --policy-name taskRolePolicy \
      --policy-document file://s3-policy.json
```

Gunakan prosedur berikut untuk membuat peran layanan.

AWS Management Console

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih kasus penggunaan Elastic Container Service Task.
5. Pilih Berikutnya.

6. Untuk Menambahkan izin, cari dan pilih kebijakan yang Anda buat.
7. Pilih Berikutnya.
8. Untuk Nama peran, masukkan nama peran Anda. Untuk contoh ini, ketik `AmazonECSTaskS3BucketRole` untuk memberi nama peran.
9. Tinjau peran lalu pilih Buat peran.

AWS CLI

1. Buat file bernama `ecs-tasks-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM tugas. File harus berisi yang berikut ini. Ganti pengenal Wilayah dan tentukan nomor AWS akun yang Anda gunakan saat meluncurkan tugas.

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

2. Buat peran IAM bernama `ecsTaskRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsTaskRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut. Ganti *taskRolePolicy* dengan nama kebijakan yang Anda buat.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'
```

4. Lampirkan kebijakan IAM yang Anda buat ke `ecsTaskRole` peran. Ganti `policy-arn` dengan ARN dari kebijakan yang Anda buat.

```
aws iam attach-role-policy \
      --role-name ecsTaskRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy
```

Setelah Anda membuat peran, tambahkan izin tambahan ke peran untuk fitur berikut.

Fitur	Izin tambahan
Gunakan ECS Exec	Izin ECS Exec
Menggunakan gambar dari repositori Amazon ECR pribadi	Izin Amazon ECR
Gunakan EC2 instance (Windows dan Linux)	Amazon membuat EC2 instans konfigurasi tambahan
Gunakan instance eksternal	Konfigurasi tambahan contoh eksternal
Gunakan EC2 instance Windows	Konfigurasi tambahan instans Amazon EC2 Windows

Izin Amazon ECR

Izin berikut diperlukan saat kode aplikasi Anda perlu berinteraksi dengan repositori Amazon ECR secara langsung. Perhatikan bahwa untuk implementasi dasar di mana Anda hanya perlu menarik

gambar dari Amazon ECR, izin ini tidak diperlukan pada tingkat peran IAM tugas. Sebagai gantinya, peran eksekusi tugas Amazon ECS harus memiliki izin ini. Untuk informasi selengkapnya tentang peran pelaksanaan tugas, lihat Peran IAM eksekusi tugas Amazon ECS.

Jika kode aplikasi yang berjalan di container perlu berinteraksi dengan Amazon ECR APIs secara langsung, Anda harus menambahkan izin berikut ke peran IAM tugas dan menyertakan peran IAM tugas dalam definisi tugas Anda. Untuk informasi lebih lanjut, lihat Menambahkan dan Menghapus Kebijakan IAM dalam Panduan Pengguna IAM.

Gunakan kebijakan berikut untuk peran IAM tugas Anda untuk menambahkan izin ECR Amazon yang diperlukan untuk aplikasi penampung yang perlu berinteraksi dengan Amazon ECR secara langsung:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Izin ECS Exec

Fitur ECS Exec memerlukan peran IAM tugas untuk memberikan kontainer izin yang diperlukan untuk komunikasi antara agen SSM terkelola (`execute-command`agen) dan layanan SSM. Anda harus menambahkan izin berikut ke peran IAM tugas dan menyertakan peran IAM tugas dalam definisi tugas Anda. Untuk informasi lebih lanjut, lihat Menambahkan dan Menghapus Kebijakan IAM dalam Panduan Pengguna IAM.

Gunakan kebijakan berikut untuk peran IAM tugas Anda untuk menambahkan izin SSM yang diperlukan.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

Amazon membuat EC2 instans konfigurasi tambahan

Sebaiknya Anda membatasi izin dalam peran instance container Anda ke daftar izin minimal yang digunakan dalam kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola.

EC2 Instans Amazon Anda memerlukan setidaknya versi agen `1.11.0` penampung untuk menggunakan peran tugas; namun, sebaiknya gunakan versi agen penampung terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memerlukan setidaknya `1.11.0-1` paket. `ecs-init` Jika instans Anda menggunakan AMI terbaru yang dioptimalkan Amazon ECS, maka instans tersebut berisi versi yang diperlukan dari agen penampung dan. `ecs-init` Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan untuk instance container Anda, tambahkan `--net=host` opsi ke docker run perintah yang memulai agen dan variabel konfigurasi agen berikut untuk konfigurasi yang Anda inginkan (untuk informasi selengkapnya, lihat): Konfigurasi agen kontainer Amazon ECS

`ECS_ENABLE_TASK_IAM_ROLE=true`

Menggunakan peran IAM untuk tugas untuk wadah dengan mode `bridge` dan `default` jaringan.

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

Menggunakan peran IAM untuk tugas untuk kontainer dengan mode `host` jaringan. Variabel ini hanya didukung pada agen versi 1.12.0 dan yang lebih baru.

Untuk contoh run command, lihat Memperbarui agen penampung Amazon ECS secara manual (untuk non-Amazon ECS yang dioptimalkan) AMIs. Anda juga perlu mengatur perintah jaringan berikut pada instance container Anda sehingga container dalam tugas Anda dapat mengambil AWS kredensialnya:

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

Anda harus menyimpan aturan iptables ini pada instans kontainer Anda agar dapat bertahan saat booting ulang. Anda dapat menggunakan perintah iptables-save dan iptables-restore untuk menyimpan aturan iptables dan memulihkannya saat booting. Untuk informasi selengkapnya, konsultasikan dokumentasi sistem operasi tertentu Anda.

Untuk mencegah container yang dijalankan oleh tugas yang menggunakan mode `awsvpc` jaringan mengakses informasi kredensi yang diberikan ke profil EC2 instans Amazon, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, setel variabel konfigurasi `ECS_AWSVPC_BLOCK_IMDS` agen ke `true` dalam file konfigurasi agen dan restart agen. Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS.

Untuk mencegah container yang dijalankan oleh tugas yang menggunakan mode `bridge` jaringan mengakses informasi kredensi yang diberikan ke profil EC2 instans Amazon, sambil tetap mengizinkan izin yang disediakan oleh peran tugas, dengan menjalankan iptables perintah berikut di instans Amazon Anda. EC2 Perintah ini tidak memengaruhi kontainer dalam tugas yang menggunakan mode `host` atau `awsvpc` jaringan. Untuk informasi selengkapnya, lihat Mode jaringan.

- ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER-USER 1 --in-
  interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```

  Anda harus menyimpan iptables aturan ini di EC2 instans Amazon Anda agar dapat bertahan dari reboot. Saat menggunakan AMI Amazon ECS yang dioptimalkan, Anda dapat menggunakan perintah berikut. Untuk sistem operasi lain, lihat dokumentasi untuk sistem operasi tersebut.
  ```

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

Konfigurasi tambahan contoh eksternal

Instance eksternal Anda memerlukan setidaknya versi `1.11.0` agen kontainer untuk menggunakan peran IAM tugas; namun, sebaiknya gunakan versi agen kontainer terbaru. Untuk informasi tentang memeriksa versi agen Anda dan memperbarui ke versi terbaru, lihat Memperbarui agen kontainer Amazon ECS. Jika Anda menggunakan AMI Amazon ECS yang dioptimalkan, instans Anda memerlukan setidaknya `1.11.0-1` paket. `ecs-init` Jika instans Anda menggunakan AMI terbaru yang dioptimalkan Amazon ECS, maka instans tersebut berisi versi yang diperlukan dari agen penampung dan. `ecs-init` Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.

Jika Anda tidak menggunakan AMI Amazon ECS yang dioptimalkan untuk instance container Anda, tambahkan `--net=host` opsi ke docker run perintah yang memulai agen dan variabel konfigurasi agen berikut untuk konfigurasi yang Anda inginkan (untuk informasi selengkapnya, lihat): Konfigurasi agen kontainer Amazon ECS

`ECS_ENABLE_TASK_IAM_ROLE=true`

Menggunakan peran IAM untuk tugas untuk wadah dengan mode `bridge` dan `default` jaringan.

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

Menggunakan peran IAM untuk tugas untuk kontainer dengan mode `host` jaringan. Variabel ini hanya didukung pada agen versi 1.12.0 dan yang lebih baru.

Untuk contoh run command, lihat Memperbarui agen penampung Amazon ECS secara manual (untuk non-Amazon ECS yang dioptimalkan) AMIs. Anda juga perlu mengatur perintah jaringan berikut pada instance container Anda sehingga container dalam tugas Anda dapat mengambil AWS kredensialnya:

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

Anda harus menyimpan aturan iptables ini pada instans kontainer Anda agar dapat bertahan saat booting ulang. Anda dapat menggunakan perintah iptables-save dan iptables-restore untuk menyimpan aturan iptables dan memulihkannya saat booting. Untuk informasi selengkapnya, konsultasikan dokumentasi sistem operasi tertentu Anda.

Konfigurasi tambahan instans Amazon EC2 Windows

> **Important**
>
> Ini hanya berlaku untuk wadah Windows EC2 yang menggunakan peran tugas.

Peran tugas dengan fitur Windows memerlukan konfigurasi tambahan aktif EC2.

- Ketika Anda meluncurkan instans kontainer, Anda harus mengatur opsi `-EnableTaskIAMRole` skrip data pengguna instans kontainer. `EnableTaskIAMRole`Menghidupkan fitur peran IAM Tugas untuk tugas. Misalnya:

  ```
  <powershell>
  Import-Module ECSTools
  Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
  </powershell>
  ```

- Anda harus bootstrap kontainer Anda dengan perintah jaringan yang disediakan di Skrip bootstrap wadah Amazon ECS.
- Anda harus membuat peran dan kebijakan IAM untuk tugas Anda. Untuk informasi selengkapnya, lihat Membuat peran tugas IAM.
- Peran IAM untuk penyedia kredensi tugas menggunakan port 80 pada instance container. Oleh karena itu, jika Anda mengonfigurasi peran IAM untuk tugas pada instance kontainer Anda, kontainer Anda tidak dapat menggunakan port 80 untuk port host di pemetaan port apa pun. Untuk mengekspos kontainer Anda pada port 80, kami sarankan mengonfigurasi layanan untuk kontainer yang menggunakan penyeimbangan beban. Anda dapat menggunakan port 80 pada penyeimbang beban. Dengan demikian, lalu lintas dapat diarahkan ke port host lain pada instans kontainer Anda. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.
- Jika instans Windows Anda dimulai ulang, Anda harus menghapus antarmuka proxy dan menginisialisasi agen penampung Amazon ECS lagi untuk mengembalikan proxy kredensi.

Skrip bootstrap wadah Amazon ECS

Sebelum kontainer dapat mengakses proxy kredensial pada instans kontainer untuk mendapatkan kredensial, kontainer harus di-bootstrap dengan perintah jaringan yang diperlukan. Skrip contoh kode berikut harus dijalankan pada kontainer Anda ketika dimulai.

> **Note**
>
> Anda tidak perlu menjalankan skrip ini saat Anda menggunakan mode jaringan `awsvpc` pada Windows.

Jika Anda menjalankan wadah Windows yang menyertakan Powershell, gunakan skrip berikut:

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

$gateway = (Get-NetRoute | Where { $_.DestinationPrefix -eq '0.0.0.0/0' } | Sort-Object
 RouteMetric | Select NextHop).NextHop
$ifIndex = (Get-NetAdapter -InterfaceDescription "Hyper-V Virtual Ethernet*" | Sort-
Object | Select ifIndex).ifIndex
New-NetRoute -DestinationPrefix 169.254.170.2/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # credentials API
New-NetRoute -DestinationPrefix 169.254.169.254/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # metadata API
```

Jika Anda menjalankan wadah Windows yang hanya memiliki shell Command, maka gunakan skrip berikut:

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
```

```
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

for /f "tokens=1" %i  in ('netsh interface ipv4 show interfaces ^| findstr /x /r
 ".*vEthernet.*"') do set interface=%i
for /f "tokens=3" %i  in ('netsh interface ipv4 show addresses %interface% ^| findstr /
x /r ".*Default.Gateway.*"') do set gateway=%i
netsh interface ipv4 add route prefix=169.254.170.2/32 interface="%interface%"
 nexthop="%gateway%" store=active # credentials API
netsh interface ipv4 add route prefix=169.254.169.254/32 interface="%interface%"
 nexthop="%gateway%" store=active # metadata API
```

Peran IAM instance wadah Amazon ECS

Instans penampung Amazon ECS, termasuk Amazon EC2 dan instans eksternal, menjalankan agen penampung Amazon ECS dan memerlukan peran IAM agar layanan mengetahui bahwa agen tersebut milik Anda. Sebelum meluncurkan instance kontainer dan mendaftarkannya ke klaster, Anda harus membuat peran IAM agar instance container Anda dapat digunakan. Peran dibuat di akun yang Anda gunakan untuk masuk ke konsol atau menjalankan AWS CLI perintah.

> **Important**
>
> Jika Anda mendaftarkan instance eksternal ke klaster, peran IAM yang Anda gunakan juga memerlukan izin Systems Manager. Untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere.

Amazon ECS menyediakan kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola yang berisi izin yang diperlukan untuk menggunakan set fitur Amazon ECS lengkap. Kebijakan terkelola ini dapat dilampirkan ke peran IAM dan dikaitkan dengan instance container Anda. Atau, Anda dapat menggunakan kebijakan terkelola sebagai panduan saat membuat kebijakan kustom untuk digunakan. Peran instance container memberikan izin yang diperlukan untuk memanggil

agen penampung Amazon ECS dan daemon Docker atas nama Anda. AWS APIs Untuk informasi selengkapnya tentang kebijakan terkelola, lihat EC2ContainerServiceforEC2Peran Amazon.

Buat peran instance kontainer

> **Important**
>
> Jika Anda mendaftarkan instans eksternal untuk klaster Anda, lihat Peran IAM Amazon ECS Anywhere.

Anda dapat membuat peran secara manual dan melampirkan kebijakan IAM terkelola untuk instance container agar Amazon ECS dapat menambahkan izin untuk fitur dan penyempurnaan future saat diperkenalkan. Gunakan prosedur berikut untuk melampirkan kebijakan IAM terkelola jika diperlukan.

AWS Management Console

Untuk membuat peran layanan untuk Elastic Container Service (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk Service atau use case, pilih Elastic Container Service, lalu pilih EC2 Role for Elastic Container Service use case.
5. Pilih Berikutnya.
6. Di bagian Kebijakan izin, verifikasi bahwa kebijakan EC2ContainerServiceforEC2Peran Amazon dipilih.

   > **Important**
   >
   > Kebijakan terkelola EC2ContainerServiceforEC2Peran Amazon harus dilampirkan ke peran IAM instance container, jika tidak, Anda akan menerima error menggunakan AWS Management Console untuk membuat cluster.

7. Pilih Berikutnya.
8. Untuk nama Peran, masukkan ecsInstanceRole

9. Tinjau peran lalu pilih Buat peran.

AWS CLI

Ganti semua *user input* dengan nilai Anda sendiri.

1. Buat file bernama `instance-role-trust-policy.json` dengan konten berikut.

   JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Gunakan perintah berikut untuk membuat peran IAM instance menggunakan dokumen kebijakan kepercayaan.

```
aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://instance-role-trust-policy.json
```

3. Buat profil instance bernama `ecsInstanceRole-profile` menggunakan create-instance-profileperintah.

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole-profile
```

   Contoh tanggapan

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJTLBPJLEGREXAMPLE",
        "Roles": [],
        "CreateDate": "2022-04-12T23:53:34.093Z",
```

```
        "InstanceProfileName": "ecsInstanceRole-profile",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole-
profile"
    }
}
```

4. Tambahkan peran *ecsInstanceRole* pada profil instans *ecsInstanceRole-profile*.

```
aws iam add-role-to-instance-profile \
    --instance-profile-name ecsInstanceRole-profile \
    --role-name ecsInstanceRole
```

5. Lampirkan kebijakan `AmazonEC2ContainerServiceRoleForEC2Role` terkelola ke peran menggunakan perintah berikut.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

Setelah Anda membuat peran, tambahkan izin tambahan ke peran untuk fitur berikut.

Fitur	Izin tambahan
Amazon ECR memiliki gambar kontainer	Izin Amazon ECR
Minta CloudWatch Log memantau instance kontainer	Memantau izin instance kontainer
Host file konfigurasi dalam bucket Amazon S3	Akses hanya-baca Amazon S3

Izin Amazon ECR

Peran instans penampung Amazon ECS yang Anda gunakan dengan instance container harus memiliki izin kebijakan IAM berikut untuk Amazon ECR.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Jika Anda menggunakan kebijakan terkelola `AmazonEC2ContainerServiceforEC2Role` untuk instans kontainer Anda, maka peran anda memiliki izin yang tepat. Untuk memeriksa apakah peran Anda mendukung Amazon ECR, lihat Peran IAM Instance Amazon ECS Container di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Izin yang diperlukan untuk menyetel pengaturan akun AWSVPCtrunking

Amazon ECS mendukung peluncuran instans kontainer dengan peningkatan kepadatan ENI menggunakan jenis EC2 instans Amazon yang didukung. Saat Anda menggunakan fitur ini, kami sarankan Anda membuat 2 peran instance container. Aktifkan pengaturan `awsvpcTrunking` akun untuk satu peran dan gunakan peran itu untuk tugas yang memerlukan trunking ENI. Untuk informasi tentang pengaturan `awsvpcTrunking` akun, lihatAkses fitur Amazon ECS dengan pengaturan akun.

Peran instance container yang Anda gunakan dengan instance container harus memiliki izin kebijakan IAM berikut untuk menyetel setelan akun

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ListAccountSettings",
                "ecs:ListAttributes",
                "ecs:PutAccountSetting"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk menggunakan peran instance container, tambahkan yang berikut ini ke data pengguna instance:

```
#!/bin/bash
aws ecs put-account-setting --name awsvpcTrunking --value enabled --region region
ECS_CLUSTER=MyCluster>> /etc/ecs/ecs.config
EOF
```

Untuk informasi selengkapnya tentang menambahkan data pengguna ke EC2 instans Anda, lihat Menjalankan perintah pada instans Linux saat diluncurkan di Panduan EC2 Pengguna Amazon.

Akses hanya-baca Amazon S3

Menyimpan informasi konfigurasi dalam bucket pribadi di Amazon S3 dan memberikan akses hanya-baca ke peran IAM instance container Anda adalah cara yang aman dan nyaman untuk mengizinkan konfigurasi instance container pada waktu peluncuran. Anda dapat menyimpan salinan `ecs.config` file Anda dalam bucket pribadi, menggunakan data EC2 pengguna Amazon untuk menginstal AWS CLI dan kemudian menyalin informasi konfigurasi Anda ke `/etc/ecs/ecs.config` saat instance diluncurkan.

Untuk informasi selengkapnya tentang membuat `ecs.config` file, menyimpannya di Amazon S3, dan meluncurkan instance dengan konfigurasi ini, lihat. Menyimpan konfigurasi instans penampung Amazon ECS di Amazon S3

Anda dapat menggunakan AWS CLI perintah berikut untuk mengizinkan akses hanya-baca Amazon S3 untuk peran instance container Anda. Ganti *ecsInstanceRole* dengan nama peran yang Anda buat.

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

Anda juga dapat menggunakan konsol IAM untuk menambahkan akses `AmazonS3ReadOnlyAccess` () hanya-baca Amazon S3 ke peran Anda. Untuk informasi selengkapnya, lihat Memperbarui izin untuk peran dalam Panduan AWS Identity and Access Management Pengguna.

Izin yang diperlukan untuk menyetel pengaturan akun AWSVPCtrunking

Peran instans penampung Amazon ECS yang Anda gunakan dengan instance container harus memiliki izin kebijakan IAM berikut untuk menyetel setelan akun

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ListAccountSettings",
                "ecs:ListAttributes",
                "ecs:PutAccountSetting"
            ],
            "Resource": "*"
        }
    ]
}
```

Memantau izin instance kontainer

Sebelum instans penampung Anda dapat mengirim data CloudWatch log ke Log, Anda harus membuat kebijakan IAM agar agen Amazon ECS dapat menulis log aplikasi pelanggan CloudWatch (biasanya ditangani melalui driver). `awslogs` Setelah Anda membuat kebijakan, lampirkan kebijakan tersebut`ecsInstanceRole`.

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke peran instance container. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memperbarui izin untuk peran dalam Panduan AWS Identity and Access Management Pengguna.

AWS CLI

1. Buat file bernama `instance-cw-logs.json` dengan konten berikut.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name cwlogspolicy \
      --policy-document file://instance-cw-logs.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut. Ganti *cwlogspolicy* dengan nama kebijakan yang Anda buat.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogspolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran IAM instance container menggunakan ARN kebijakan.

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cwlogspolicy
```

Peran IAM Amazon ECS Anywhere

Saat Anda mendaftarkan server lokal atau mesin virtual (VM) ke klaster Anda, server atau VM memerlukan peran IAM untuk berkomunikasi dengannya. AWS APIs Anda hanya perlu membuat peran IAM ini sekali untuk setiap AWS akun. Namun, peran IAM ini harus dikaitkan dengan setiap server atau VM yang Anda daftarkan ke cluster. Peran ini adalah `ECSAnywhereRole`. Anda dapat membuat peran ini secara manual. Atau, Amazon ECS dapat membuat peran atas nama Anda saat Anda mendaftarkan instans eksternal di AWS Management Console. Anda dapat menggunakan pencarian konsol IAM untuk mencari `ecsAnywhereRole` dan melihat apakah akun Anda sudah memiliki peran tersebut. Untuk informasi selengkapnya, lihat pencarian konsol IAM di panduan pengguna IAM.

AWS menyediakan dua kebijakan IAM terkelola yang dapat digunakan saat membuat peran IAM ECS Anywhere, dan kebijakan. `AmazonSSMManagedInstanceCore` `AmazonEC2ContainerServiceforEC2Role` Kebijakan `AmazonEC2ContainerServiceforEC2Role` meliputi izin yang menyediakan lebih banyak akses daripada yang Anda butuhkan. Oleh karena itu, dengan bergantung pada kasus penggunaan khusus Anda, kami sarankan supaya Anda membuat kebijakan kustom yang hanya menambahkan izin dari kebijakan yang Anda perlukan di dalamnya. Untuk informasi lebih lanjut, lihat Peran IAM instance wadah Amazon ECS.

Peran IAM eksekusi tugas memberikan izin agen penampung Amazon ECS untuk melakukan panggilan AWS API atas nama Anda. Ketika peran IAM eksekusi tugas digunakan, itu harus ditentukan dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.

Peran pelaksanaan tugas diperlukan jika salah satu syarat berikut berlaku:

- Anda mengirim log kontainer ke CloudWatch Log menggunakan driver `awslogs` log.
- Definisi tugas Anda menentukan gambar kontainer yang di-host di repositori pribadi Amazon ECR. Namun, jika `ECSAnywhereRole` peran yang terkait dengan instans eksternal Anda juga menyertakan izin yang diperlukan untuk menarik gambar dari Amazon ECR, maka peran eksekusi tugas Anda tidak perlu menyertakannya.

Membuat peran Amazon ECS Anywhere

Ganti semua *`user input`* dengan informasi Anda sendiri.

1. Buat file lokal bernama `ssm-trust-policy.json` dengan kebijakan kepercayaan berikut.

   JSON

   ```
   {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Allow",
       "Principal": {"Service": [
         "ssm.amazonaws.com"
       ]},
       "Action": "sts:AssumeRole"
     }
   }
   ```

2. Buat peran dan lampirkan kebijakan kepercayaan dengan menggunakan AWS CLI perintah berikut.

   ```
   aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
    file://ssm-trust-policy.json
   ```

3. Lampirkan kebijakan AWS terkelola dengan menggunakan perintah berikut.

   ```
   aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
    arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
   aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
    arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
   ```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom IAM untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Peran IAM infrastruktur Amazon ECS

Peran IAM infrastruktur Amazon ECS memungkinkan Amazon ECS mengelola sumber daya infrastruktur di klaster Anda atas nama Anda, dan digunakan saat:

- Anda ingin melampirkan volume Amazon EBS ke Fargate Anda EC2 atau meluncurkan tugas Amazon ECS jenis. Peran infrastruktur memungkinkan Amazon ECS mengelola volume Amazon EBS untuk tugas Anda.
- Anda ingin menggunakan Transport Layer Security (TLS) untuk mengenkripsi lalu lintas antara layanan Amazon ECS Service Connect Anda.
- Anda ingin membuat grup target Amazon VPC Lattice.

Ketika Amazon ECS mengasumsikan peran ini untuk mengambil tindakan atas nama Anda, acara akan terlihat di. AWS CloudTrail Jika Amazon ECS menggunakan peran untuk mengelola volume Amazon EBS yang dilampirkan ke tugas Anda, CloudTrail log `roleSessionName` akan menjadi. `ECSTaskVolumesForEBS` Jika peran digunakan untuk mengenkripsi lalu lintas antara layanan Service Connect Anda, CloudTrail log `roleSessionName` akan menjadi`ECSServiceConnectForTLS`. Jika peran digunakan untuk membuat grup target untuk VPC Lattice, CloudTrail log `roleSessionName` akan menjadi. `ECSNetworkingWithVPCLattice` Anda dapat menggunakan nama ini untuk mencari peristiwa di CloudTrail konsol dengan memfilter nama Pengguna.

Amazon ECS menyediakan kebijakan terkelola yang berisi izin yang diperlukan untuk lampiran volume dan TLS. Untuk informasi selengkapnya, lihat Amazon ECSInfrastructure RolePolicyForVolumes, Amazon ECSInfrastructure RolePolicyForServiceConnectTransportLayerSecurity, dan Amazon ECSInfrastructure RolePolicyForVpcLattice di Panduan Referensi Kebijakan AWS Terkelola.

Menciptakan peran infrastruktur Amazon ECS

Ganti semua *`user input`* dengan informasi Anda sendiri.

1. Buat file bernama `ecs-infrastructure-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToECSForInfrastructureManagement",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Gunakan AWS CLI perintah berikut untuk membuat peran `ecsInfrastructureRole` bernama menggunakan kebijakan kepercayaan yang Anda buat di langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsInfrastructureRole \
      --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. Bergantung pada kasus penggunaan Anda, lampirkan `AmazonECSInfrastructureRolePolicyForVpcLattice` kebijakan yang AWS dikelola `AmazonECSInfrastructureRolePolicyForVolumesAmazonECSInfrastructureRolePolicyF` atau ke `ecsInfrastructureRole` peran tersebut.

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForVolumes
```

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Important

Jika peran infrastruktur ECS digunakan oleh Amazon ECS untuk mengelola volume Amazon EBS yang dilampirkan ke tugas Anda, pastikan hal berikut sebelum Anda menghentikan tugas yang menggunakan volume Amazon EBS.

- Peran tidak dihapus.
- Kebijakan kepercayaan untuk peran tersebut tidak dimodifikasi untuk menghapus akses Amazon ECS (`ecs.amazonaws.com`).
- Kebijakan terkelola `AmazonECSInfrastructureRolePolicyForVolumes` tidak dihapus. Jika Anda harus mengubah izin peran, setidaknya pertahankan `ec2:DetachVolumeec2:DeleteVolume`, dan `ec2:DescribeVolumes` untuk penghapusan volume.

Menghapus atau memodifikasi peran sebelum menghentikan tugas dengan volume Amazon EBS terlampir akan mengakibatkan tugas macet `DEPROVISIONING` dan volume Amazon EBS terkait gagal dihapus. Amazon ECS akan secara otomatis mencoba lagi secara berkala untuk menghentikan tugas dan menghapus volume hingga izin yang diperlukan dipulihkan. Anda dapat melihat status lampiran volume tugas dan alasan status terkait dengan menggunakan DescribeTasksAPI.

Setelah Anda membuat file, Anda harus memberikan izin pengguna Anda untuk meneruskan peran ke Amazon ECS.

Izin untuk meneruskan peran infrastruktur ke Amazon ECS

Untuk menggunakan peran IAM infrastruktur ECS, Anda harus memberikan izin kepada pengguna untuk meneruskan peran tersebut ke Amazon ECS. Lampirkan `iam:PassRole` izin berikut ke pengguna Anda. Ganti *`ecsInfrastructureRole`* dengan nama peran infrastruktur yang Anda buat.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

Untuk informasi selengkapnya tentang *iam:Passrole* dan memperbarui izin untuk pengguna Anda, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan dan Mengubah izin untuk pengguna IAM di Panduan Pengguna.AWS Identity and Access Management

Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban

Peran IAM infrastruktur Amazon ECS untuk penyeimbang beban memungkinkan Amazon ECS mengelola sumber daya penyeimbang beban di klaster Anda atas nama Anda, dan digunakan saat:

- Anda ingin menggunakan blue/green penerapan dengan Amazon ECS. Peran infrastruktur memungkinkan Amazon ECS mengelola sumber daya penyeimbang beban untuk penerapan Anda.
- Anda memerlukan Amazon ECS untuk membuat, memodifikasi, atau menghapus sumber daya penyeimbang beban seperti grup target dan pendengar selama operasi penerapan.

Ketika Amazon ECS mengasumsikan peran ini untuk mengambil tindakan atas nama Anda, acara akan terlihat di. AWS CloudTrail Jika Amazon ECS menggunakan peran untuk mengelola sumber daya penyeimbang beban untuk penerapan biru/hijau Anda, log akan menjadi atau. CloudTrail `roleSessionName` `ECSNetworkingWithELB` `ecs-service-scheduler` Anda dapat menggunakan nama ini untuk mencari peristiwa di CloudTrail konsol dengan memfilter nama Pengguna.

Amazon ECS menyediakan kebijakan terkelola yang berisi izin yang diperlukan untuk manajemen penyeimbang beban. Untuk informasi selengkapnya, lihat Amazon ECSInfrastructure RolePolicyForLoadBalancers di Panduan Referensi Kebijakan AWS Terkelola.

Menciptakan peran infrastruktur Amazon ECS untuk penyeimbang beban

Ganti semua *user input* dengan informasi Anda sendiri.

1. Buat file bernama `ecs-infrastructure-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

   JSON

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "AllowAccessToECSForInfrastructureManagement",
         "Effect": "Allow",
         "Principal": {
           "Service": "ecs.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

2. Gunakan AWS CLI perintah berikut untuk membuat peran `ecsInfrastructureRoleForLoadBalancers` bernama menggunakan kebijakan kepercayaan yang Anda buat di langkah sebelumnya.

   ```
   aws iam create-role \
         --role-name ecsInfrastructureRoleForLoadBalancers \
         --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
   ```

3. Lampirkan `AmazonECSInfrastructureRolePolicyForLoadBalancers` kebijakan yang AWS dikelola ke `ecsInfrastructureRoleForLoadBalancers` peran.

   ```
   aws iam attach-role-policy \
         --role-name ecsInfrastructureRoleForLoadBalancers \
   ```

```
    --policy-arn arn:aws:iam::aws:policy/
AmazonECSInfrastructureRolePolicyForLoadBalancers
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Important

Jika peran infrastruktur digunakan oleh Amazon ECS untuk mengelola sumber daya penyeimbang beban untuk blue/green penerapan Anda, pastikan hal berikut sebelum Anda menghapus atau memodifikasi peran:

- Peran tidak dihapus saat penerapan aktif sedang berlangsung.
- Kebijakan kepercayaan untuk peran tersebut tidak dimodifikasi untuk menghapus akses Amazon ECS (`ecs.amazonaws.com`).
- Kebijakan terkelola `AmazonECSInfrastructureRolePolicyForLoadBalancers` tidak dihapus saat penerapan aktif sedang berlangsung.

Menghapus atau memodifikasi peran selama blue/green penerapan aktif dapat mengakibatkan kegagalan penerapan dan dapat membuat layanan Anda dalam keadaan tidak konsisten.

Setelah Anda membuat file, Anda harus memberikan izin pengguna Anda untuk meneruskan peran ke Amazon ECS.

Izin untuk meneruskan peran infrastruktur ke Amazon ECS

Untuk menggunakan peran IAM infrastruktur ECS untuk penyeimbang beban, Anda harus memberikan izin kepada pengguna untuk meneruskan peran tersebut ke Amazon ECS. Lampirkan `iam:PassRole` izin berikut ke pengguna Anda. Ganti *`ecsInfrastructureRoleForLoadBalancers`* dengan nama peran infrastruktur yang Anda buat.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource":
 ["arn:aws:iam::*:role/ecsInfrastructureRoleForLoadBalancers"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

Untuk informasi selengkapnya tentang *iam:Passrole* dan memperbarui izin untuk pengguna Anda, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan dan Mengubah izin untuk pengguna IAM di Panduan Pengguna.AWS Identity and Access Management

Peran Amazon ECS CodeDeploy IAM

Sebelum Anda dapat menggunakan jenis penyebaran CodeDeploy biru/hijau dengan Amazon ECS, CodeDeploy layanan memerlukan izin untuk memperbarui layanan Amazon ECS Anda atas nama Anda. Izin ini disediakan oleh peran CodeDeploy IAM ()`ecsCodeDeployRole`.

> **Note**
>
> Pengguna juga memerlukan izin untuk digunakan CodeDeploy; izin ini dijelaskan dalam. Izin IAM yang diperlukan

Ada dua kebijakan terkelola yang disediakan. Untuk informasi selengkapnya, lihat salah satu hal berikut di Panduan Referensi Kebijakan AWS Terkelola:

- AWSCodeDeployRoleForECS - memberikan CodeDeploy izin untuk memperbarui sumber daya apa pun menggunakan tindakan terkait.
- AWSCodeDeployRoleForECSLimited- memberikan izin yang CodeDeploy lebih terbatas.

Menciptakan CodeDeploy peran

Anda dapat menggunakan prosedur berikut untuk membuat CodeDeploy peran untuk Amazon ECS

AWS Management Console

Untuk membuat peran layanan untuk CodeDeploy (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk kasus Layanan atau penggunaan, pilih CodeDeploy, lalu pilih kasus penggunaan CodeDeploy - ECS.
5. Pilih Berikutnya.
6. Di bagian Lampirkan kebijakan izin, pastikan bahwa kebijakan AWSCodeDeployRoleForECS dipilih.
7. Pilih Berikutnya.
8. Untuk nama Peran, masukkan ecsCodeDeployPeran.
9. Tinjau peran lalu pilih Buat peran.

AWS CLI

Ganti semua *user input* dengan informasi Anda sendiri.

1. Buat file bernama `codedeploy-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran CodeDeploy IAM.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": ["codedeploy.amazonaws.com"]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. Buat peran IAM bernama `ecsCodedeployRole` menggunakan kebijakan kepercayaan yang dibuat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsCodedeployRole \
      --assume-role-policy-document file://codedeploy-trust-policy.json
```

3. Lampirkan kebijakan `AWSCodeDeployRoleForECS` atau yang `AWSCodeDeployRoleForECSLimited` dikelola ke `ecsTaskRole` peran.

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam::aws:policy/AWSCodeDeployRoleForECS
```

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam::aws:policy/AWSCodeDeployRoleForECSLimited
```

Jika tugas dalam layanan Anda memerlukan peran eksekusi tugas, Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas atau penggantian peran tugas ke CodeDeploy peran sebagai kebijakan.

Izin peran eksekusi tugas

Jika tugas dalam layanan Anda memerlukan peran eksekusi tugas, Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas atau penggantian peran tugas ke CodeDeploy peran sebagai kebijakan. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS dan Peran IAM tugas Amazon ECS. Kemudian, Anda melampirkan kebijakan itu ke CodeDeploy peran

Buat kebijakan

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

6. Pilih Berikutnya.

   > **Note**
   >
   > Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.
7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.

8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke CodeDeploy peran tersebut. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memperbarui izin untuk peran dalam Panduan AWS Identity and Access Management Pengguna.

AWS CLI

Ganti semua *user input* dengan informasi Anda sendiri.

1. Buat file bernama `blue-green-iam-passrole.json` dengan konten berikut.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                   "arn:arn:aws:iam::111122223333:role/ecs-code-deploy-
role"
            ]
        }
    ]
}
```

2. Gunakan perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name cdTaskExecutionPolicy \
      --policy-document file://blue-green-iam-passrole.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran CodeDeploy IAM.

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cdTaskExecutionPolicy
```

Peran Amazon ECS EventBridge IAM

Sebelum Anda dapat menggunakan tugas terjadwal Amazon ECS dengan EventBridge aturan dan target, EventBridge layanan memerlukan izin untuk menjalankan tugas Amazon ECS atas nama Anda. Izin ini disediakan oleh peran EventBridge IAM ()`ecsEventsRole`.

Kebijakan `AmazonEC2ContainerServiceEventsRole` ditunjukkan di bawah ini.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
```

```
    ]
}
```

Jika tugas terjadwal Anda memerlukan penggunaan peran eksekusi tugas, peran tugas, atau penggantian peran tugas, Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas, peran tugas, atau penggantian peran tugas ke peran IAM. EventBridge Untuk informasi selengkapnya tentang peran eksekusi tugas, lihat Peran IAM eksekusi tugas Amazon ECS.

> **Note**
>
> Tentukan ARN penuh dari peran eksekusi tugas Anda atau penimpaan peran tugas.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::111122223333:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

Anda dapat memilih untuk membiarkan AWS Management Console membuat EventBridge peran untuk Anda ketika Anda mengkonfigurasi tugas terjadwal. Untuk informasi selengkapnya, lihat Menggunakan Amazon EventBridge Scheduler untuk menjadwalkan tugas Amazon ECS .

Menciptakan EventBridge peran

Ganti semua *user input* dengan informasi Anda sendiri.

1. Buat file bernama `eventbridge-trust-policy.json` yang berisi kebijakan kepercayaan yang akan digunakan untuk peran IAM. File tersebut harus berisi hal berikut:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Gunakan perintah berikut untuk membuat peran IAM bernama `ecsEventsRole` menggunakan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya.

```
aws iam create-role \
      --role-name ecsEventsRole \
      --assume-role-policy-document file://eventbridge-trust-policy.json
```

3. Lampirkan yang AWS dikelola `AmazonEC2ContainerServiceEventsRole` ke `ecsEventsRole` peran menggunakan perintah berikut.

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceEventsRole
```

Anda juga dapat menggunakan alur kerja kebijakan kepercayaan kustom konsol IAM (https://console.aws.amazon.com/iam/) untuk membuat peran. Untuk informasi selengkapnya, lihat Membuat peran menggunakan kebijakan kepercayaan khusus (konsol) di Panduan Pengguna IAM.

Melampirkan kebijakan pada peran **`ecsEventsRole`**

Anda dapat menggunakan prosedur berikut untuk menambahkan izin untuk peran eksekusi tugas ke peran EventBridge IAM.

AWS Management Console

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. Pilih Berikutnya.

   > **Note**
   >
   > Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.
7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.

8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Setelah Anda membuat kebijakan, lampirkan kebijakan ke EventBridge peran tersebut. Untuk informasi tentang cara melampirkan kebijakan ke peran, lihat Memperbarui izin untuk peran dalam Panduan AWS Identity and Access Management Pengguna.

AWS CLI

Ganti semua *user input* dengan informasi Anda sendiri.

1. Buat file bernama `ev-iam-passrole.json` dengan konten berikut.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [

 "arn:aws:iam::111122223333:role/ecsTaskExecutionRole_or_TaskRole_name"
            ]
        }
    ]
}
```

2. Gunakan AWS CLI perintah berikut untuk membuat kebijakan IAM menggunakan file dokumen kebijakan JSON.

```
aws iam create-policy \
      --policy-name eventsTaskExecutionPolicy \
      --policy-document file://ev-iam-passrole.json
```

3. Ambil ARN dari kebijakan IAM yang Anda buat menggunakan perintah berikut.

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'
```

4. Gunakan perintah berikut untuk melampirkan kebijakan ke peran EventBridge IAM dengan menggunakan kebijakan ARN.

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/eventsTaskExecutionPolicy
```

Izin diperlukan untuk konsol Amazon ECS

Mengikuti praktik terbaik pemberian hak istimewa paling rendah, Anda dapat menggunakan kebijakan terkelola `AmazonECS_FullAccess` sebagai templat untuk membuat kebijakan kustom Anda sendiri. Dengan begitu, Anda dapat mengambil atau menambahkan izin ke dan dari kebijakan terkelola berdasarkan kebutuhan khusus Anda. Untuk informasi selengkapnya, lihat AmazonECS_ FullAccess di Referensi Kebijakan Terkelola.AWS

Izin untuk membuat peran IAM

Tindakan berikut memerlukan izin tambahan untuk menyelesaikan operasi:

- Mendaftarkan instance eksternal - untuk informasi selengkapnya, lihat Peran IAM Amazon ECS Anywhere
- Mendaftarkan definisi tugas - untuk informasi lebih lanjut, lihat Peran IAM eksekusi tugas Amazon ECS
- Membuat EventBridge aturan yang akan digunakan untuk menjadwalkan tugas - untuk informasi selengkapnya, lihat Peran Amazon ECS EventBridge IAM

Anda dapat menambahkan izin ini dengan membuat peran di IAM sebelum menggunakannya di konsol Amazon ECS. Jika Anda tidak membuat peran, konsol Amazon ECS akan membuatnya atas nama Anda.

Izin yang diperlukan untuk mendaftarkan instance eksternal ke cluster

Anda memerlukan izin tambahan saat mendaftarkan instance eksternal ke klaster dan Anda ingin membuat peran instance eksternal (`ecsExternalInstanceRole`) baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan membuat daftar peran IAM dan kebijakan terlampir mereka.
- `ssm`— Memungkinkan prinsipal untuk mendaftarkan instance eksternal dengan Systems Manager.

> **Note**
>
> Untuk memilih yang sudah ada`ecsExternalInstanceRole`, Anda harus memiliki `iam:GetRole` dan `iam:PassRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `ecsExternalInstanceRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole","ssm:CreateActivation"],
            "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
        }
    ]
}
```

Izin yang diperlukan untuk mendaftarkan definisi tugas

Anda memerlukan izin tambahan saat mendaftarkan definisi tugas dan Anda ingin membuat peran eksekusi tugas (`ecsTaskExecutionRole`) baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan membuat daftar peran IAM dan kebijakan terlampir mereka.

> **Note**
>
> Untuk memilih yang sudah ada`ecsTaskExecutionRole`, Anda harus memiliki `iam:GetRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `ecsTaskExecutionRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
        }
    ]
}
```

Izin diperlukan untuk membuat EventBridge aturan untuk tugas terjadwal

Anda memerlukan izin tambahan saat menjadwalkan tugas dan Anda ingin membuat peran peran (`ecsEventsRole`) CloudWatch Acara baru.

Izin tambahan berikut diperlukan:

- `iam`— Memungkinkan kepala sekolah untuk membuat dan mencantumkan peran IAM dan kebijakan terlampirnya, dan mengizinkan Amazon ECS meneruskan peran tersebut ke layanan lain untuk mengambil peran tersebut.

> **Note**
>
> Untuk memilih yang sudah ada`ecsEventsRole`, Anda harus memiliki `iam:GetRole` dan `iam:PassRole` izin.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan untuk `ecsEventsRole` peran.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole",
              "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsEventsRole"
        }
    ]
}
```

Izin yang diperlukan untuk melihat penerapan layanan

Saat Anda mengikuti praktik terbaik pemberian hak istimewa paling sedikit, Anda perlu menambahkan izin tambahan untuk melihat penerapan layanan di konsol.

Anda memerlukan akses ke tindakan berikut:

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

Anda memerlukan akses ke sumber daya berikut:

- Layanan
- Penyebaran layanan
- Revisi layanan

Kebijakan contoh berikut berisi izin yang diperlukan, dan membatasi tindakan ke layanan tertentu.

Gantiaccount,cluster-name, dan service-name dengan nilai-nilai Anda.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "ecs:ListServiceDeployments",
              "ecs:DescribeServiceDeployments",
              "ecs:DescribeServiceRevisions"
            ],
            "Resource": [
             "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
             "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/
service-name/*",
             "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-
name/*"
            ]
        }
    ]
}
```

Izin diperlukan untuk melihat peristiwa siklus hidup Amazon ECS di Wawasan Kontainer

Izin berikut diperlukan untuk melihat peristiwa siklus hidup. Tambahkan izin berikut sebagai kebijakan sebaris ke peran. Untuk informasi selengkapnya, lihat Menambahkan dan Menghapus Kebijakan IAM.

- peristiwa: DescribeRule
- peristiwa: ListTargetsByRule
- log: DescribeLogGroups

JSON

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

Izin diperlukan untuk mengaktifkan peristiwa siklus hidup Amazon ECS dalam Container Insights

Izin berikut diperlukan untuk mengonfigurasi peristiwa siklus hidup:

- peristiwa: PutRule
- peristiwa: PutTargets
- log: CreateLogGroup

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "logs:CreateLogGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

Izin diperlukan untuk konsol Amazon ECS dengan AWS CloudFormation

Sebelum menggunakan AWS Management Console untuk membuat sumber daya Anda, Anda harus memastikan untuk memiliki izin IAM yang benar. Untuk informasi tentang cara mengatur izin untuk konsol Amazon ECS secara umum terlebih dahulu, lihat Izin yang diperlukan untuk konsol Amazon ECS.

Konsol Amazon ECS didukung oleh AWS CloudFormation dan memerlukan izin IAM tambahan dalam kasus berikut:

- Membuat klaster
- Membuat sebuah layanan
- Menciptakan penyedia kapasitas

Anda dapat membuat kebijakan untuk izin tambahan, lalu melampirkannya ke peran IAM yang Anda gunakan untuk mengakses konsol. Untuk informasi selengkapnya, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Izin yang diperlukan untuk membuat klaster

Saat membuat klaster di konsol, Anda memerlukan izin tambahan yang memberi Anda izin untuk mengelola AWS CloudFormation tumpukan.

Izin tambahan berikut diperlukan:

- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat cluster Amazon ECS menggunakan AWS Management Console dan pengelolaan cluster tersebut selanjutnya.
- `ssm`- Memungkinkan AWS CloudFormation untuk mereferensikan AMI Amazon ECS terbaru yang dioptimalkan. Ini diperlukan saat membuat cluster Amazon ECS menggunakan file. AWS Management Console

Kebijakan berikut berisi AWS CloudFormation izin yang diperlukan, dan membatasi tindakan ke sumber daya yang dibuat di konsol Amazon ECS.

```
{
  "Statement": [
      {
```

```
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
            ]
      },
      {
          "Effect": "Allow",
          "Action": "ssm:GetParameters",
          "Resource": [
            "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-linux-2*/
*",
            "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-
linux-2023*/*"
          ]
      }
   ]
}
```

Jika Anda belum membuat role instance container Amazon ECS (`ecsInstanceRole`), dan Anda membuat klaster yang menggunakan EC2 instans Amazon, konsol akan membuat peran tersebut atas nama Anda.

Selain itu, jika Anda menggunakan grup Auto Scaling, maka Anda memerlukan izin tambahan agar konsol dapat menambahkan tag ke grup penskalaan otomatis saat menggunakan fitur penskalaan otomatis cluster.

Izin tambahan berikut diperlukan:

- `autoscaling`— Memungkinkan konsol untuk menandai grup Amazon EC2 Auto Scaling. Ini diperlukan saat mengelola grup penskalaan EC2 otomatis Amazon saat menggunakan fitur penskalaan otomatis cluster. Tag adalah tag yang dikelola ECS yang secara otomatis ditambahkan konsol ke grup untuk menunjukkan dibuat di konsol.
- `iam`— Memungkinkan kepala sekolah untuk membuat daftar peran IAM dan kebijakan terlampir mereka. Prinsipal juga dapat mencantumkan profil instans yang tersedia untuk instans Amazon Anda. EC2

Kebijakan berikut berisi izin IAM yang diperlukan, dan membatasi tindakan untuk peran. `ecsInstanceRole`

Izin Auto Scaling tidak terbatas.

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateOrUpdateTags",
            "Resource": "*"
        }
    ]
}
```

Izin yang diperlukan untuk membuat layanan

Saat membuat layanan di konsol, Anda memerlukan izin tambahan yang memberi Anda izin untuk mengelola AWS CloudFormation tumpukan. Izin tambahan berikut diperlukan:

- `cloudformation`— Memungkinkan kepala sekolah untuk membuat dan mengelola tumpukan. AWS CloudFormation Ini diperlukan saat membuat layanan Amazon ECS menggunakan AWS Management Console dan pengelolaan selanjutnya dari layanan tersebut.

Kebijakan berikut berisi izin yang diperlukan, dan membatasi tindakan ke sumber daya yang dibuat di konsol Amazon ECS.

```
{
  "Statement": [
```

```
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
            ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
            ]
      }
   ]
}
```

Izin diperlukan untuk fungsi Lambda di penerapan Amazon ECS blue/green

Saat Anda menggunakan fungsi Lambda sebagai kait siklus hidup penerapan di penerapan biru/hijau Amazon ECS, Anda perlu membuat peran IAM dengan izin tertentu. Peran ini memungkinkan Amazon ECS untuk menjalankan fungsi Lambda Anda pada berbagai tahap siklus hidup penerapan.

Izin tambahan berikut diperlukan:

- `lambda:InvokeFunction`— Memungkinkan Amazon ECS untuk memanggil fungsi Lambda yang dikonfigurasi sebagai kait siklus hidup selama proses penerapan.

Kebijakan berikut berisi izin yang diperlukan untuk kait siklus hidup Lambda:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:*:*:function:*"
        }
    ]
```

```
}
```

Untuk kebijakan kepercayaan, Anda perlu mengizinkan layanan untuk mengambil peran ini:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ecs.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Izin IAM diperlukan untuk penskalaan otomatis layanan Amazon ECS

Service Auto Scaling dimungkinkan oleh kombinasi Amazon ECS, CloudWatch, dan Application Auto Scaling. APIs Layanan dibuat dan diperbarui dengan Amazon ECS, alarm dibuat dengan CloudWatch, dan kebijakan penskalaan dibuat dengan Application Auto Scaling.

Selain izin IAM standar untuk membuat dan memperbarui layanan, izin berikut diperlukan untuk berinteraksi dengan pengaturan Auto Scaling Layanan seperti yang ditunjukkan dalam contoh kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ecs:DescribeServices",
```

```
                "ecs:UpdateService",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Contoh kebijakan Perbarui contoh layanan Amazon ECS IAM Buat contoh layanan Amazon ECS dan IAM menunjukkan izin yang diperlukan untuk menggunakan Auto Scaling Service di. AWS Management Console

Layanan Application Auto Scaling juga memerlukan izin untuk menjelaskan layanan dan CloudWatch alarm Amazon ECS Anda, dan izin untuk mengubah jumlah layanan yang diinginkan atas nama Anda. `sns:`Izin adalah untuk notifikasi yang CloudWatch dikirim ke topik Amazon SNS ketika ambang batas telah terlampaui. Jika Anda menggunakan penskalaan otomatis untuk layanan Amazon ECS Anda, itu akan membuat peran terkait layanan bernama. `AWSServiceRoleForApplicationAutoScaling_ECSService` Peran terkait layanan ini memberikan izin Application Auto Scaling untuk menjelaskan alarm kebijakan Anda, memantau jumlah tugas layanan yang sedang berjalan, dan mengubah jumlah layanan yang diinginkan. Peran Amazon ECS yang dikelola asli untuk Application Auto Scaling `ecsAutoscaleRole` adalah, tetapi tidak lagi diperlukan. Peran tertaut layanan adalah peran default untuk Penskalaan Otomatis Aplikasi. Untuk informasi selengkapnya, lihat Peran tertaut layanan untuk Application Auto Scaling dalam Panduan Pengguna Application Auto Scaling.

Jika Anda membuat peran instans penampung Amazon ECS sebelum CloudWatch metrik tersedia untuk Amazon ECS, Anda mungkin perlu menambahkan izin. `ecs:StartTelemetrySession` Untuk informasi selengkapnya, lihat Pertimbangan.

Berikan izin untuk menandai sumber daya pada pembuatan

Tindakan penandaan Amazon ECS API berikut memungkinkan Anda menentukan tag saat membuat sumber daya. Jika tag ditentukan dalam tindakan pembuatan sumber daya, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa izin yang benar ditetapkan untuk membuat tag.

- `CreateCapacityProvider`
- `CreateCluster`
- `CreateService`
- `CreateTaskSet`
- `RegisterContainerInstance`
- `RegisterTaskDefinition`
- `RunTask`
- `StartTask`

Anda dapat menggunakan tanda sumber daya untuk menerapkan pengendalian berbasis atribut (ABAC). Untuk informasi selengkapnya, lihat the section called “Kontrol akses ke sumber daya Amazon ECS menggunakan tag sumber daya” dan Pemberian tag pada sumber daya .

Untuk mengizinkan penandaan pada pembuatan, buat atau ubah kebijakan untuk menyertakan izin untuk menggunakan tindakan yang membuat sumber daya, seperti `ecs:CreateCluster` atau `ecs:RunTask` dan tindakan. `ecs:TagResource`

Contoh berikut menunjukkan kebijakan yang memungkinkan pengguna membuat cluster dan menambahkan tag selama pembuatan klaster. Para pengguna tidak diizinkan untuk memberi tanda pada sumber daya yang sudah ada (mereka tidak dapat memerintahkan tindakan `ecs:TagResource` secara langsung).

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:CreateCluster"
      ],
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
                  "ecs:CreateAction": [
                      "CreateCluster",
                      "CreateCapacityProvider",
                      "CreateService",
                      "CreateTaskSet",
                      "RegisterContainerInstance",
                      "RegisterTaskDefinition",
                      "RunTask",
                      "StartTask"
                  ]
          }
       }
    }
  ]
}
```

Tindakan `ecs:TagResource` akan dievaluasi hanya jika tanda diterapkan selama tindakan pembuatan sumber daya. Oleh karena itu, seorang pengguna yang memiliki izin untuk membuat sumber daya (dengan asumsi tidak ada syarat untuk pemberian tanda) tidak memerlukan izin untuk menggunakan tindakan `ecs:TagResource` jika tidak ada tanda yang ditentukan dalam permintaan. Akan tetapi, jika pengguna tersebut mencoba untuk membuat sumber daya dengan tanda, maka permintaan akan gagal jika pengguna tidak memiliki izin untuk menggunakan tindakan `ecs:TagResource`.

Amazon ECS mengontrol akses ke tag tertentu

Anda dapat menggunakan syarat tambahan dalam elemen `Condition` dari kebijakan IAM Anda untuk mengontrol kunci tanda dan nilai tanda yang dapat diterapkan ke sumber daya.

Kunci syarat berikut dapat digunakan dengan contoh-contoh pada bagian sebelumnya:

- `aws:RequestTag`: Untuk mengindikasikan bahwa kunci tanda tertentu atau kunci dan nilai tanda tertentu harus ada di permintaan. Tanda-tanda yang lain juga dapat ditentukan dalam permintaan.

- Gunakan bersama dengan operator syarat `StringEquals` untuk memberlakukan kombinasi kunci dan nilai tanda tertentu, misalnya, untuk memberlakukan tanda `cost-center=cc123`:

  ```
  "StringEquals": { "aws:RequestTag/cost-center": "cc123" }
  ```

- Gunakan bersama dengan operator syarat `StringLike` untuk memberlakukan kunci tanda tertentu dalam permintaan, misalnya, untuk memberlakukan kunci tanda `purpose`:

  ```
  "StringLike": { "aws:RequestTag/purpose": "*" }
  ```

- `aws:TagKeys`: Untuk memberlakukan kunci tanda yang digunakan dalam permintaan.
  - Gunakan bersama dengan pemodifikasi `ForAllValues` untuk menerapkan kunci tanda tertentu jika disediakan dalam permintaan (jika tanda ditentukan dalam permintaan, hanya kunci tanda tertentu saja yang diperbolehkan; tidak ada tanda lain yang diperbolehkan). Sebagai contoh, kunci tanda `environment` atau `cost-center` diperbolehkan:

    ```
    "ForAllValues:StringEquals": { "aws:TagKeys": ["environment","cost-center"] }
    ```

  - Gunakan pemodifikasi `ForAnyValue` untuk memaksakan keberadaan setidaknya salah satu kunci tanda tertentu dalam permintaan. Sebagai contoh, setidaknya salah satu kunci tanda `environment` atau `webserver` harus ada dalam permintaan:

    ```
    "ForAnyValue:StringEquals": { "aws:TagKeys": ["environment","webserver"] }
    ```

Kunci kondisi ini dapat diterapkan ke tindakan pembuatan sumber daya yang mendukung penandaan, serta tindakan. `ecs:TagResource` Untuk mengetahui apakah tindakan Amazon ECS API mendukung penandaan, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon ECS.

Untuk memaksa para pengguna menentukan tanda pada saat mereka membuat sumber daya, Anda harus menggunakan kunci syarat `aws:RequestTag` atau kunci syarat `aws:TagKeys` dengan pemodifikasi `ForAnyValue` pada tindakan yang digunakan untuk membuat sumber daya. Tindakan `ecs:TagResource` tidak akan dievaluasi jika pengguna tidak menentukan tanda untuk tindakan yang digunakan untuk pembuatan sumber daya.

Untuk syarat, kunci syarat tidak bersifat peka terhadap huruf besar dan kecil dan nilai syarat bersifat peka huruf besar dan kecil. Oleh karena itu, untuk memaksakan sifat peka terhadap huruf besar atau kecil dari kunci tanda, gunakan kunci syarat `aws:TagKeys`, di mana kunci tanda ditetapkan sebagai nilai dalam syarat tersebut.

Untuk informasi selengkapnya tentang kondisi multi-nilai, lihat Ketentuan dengan beberapa kunci konteks atau nilai dalam Panduan Pengguna IAM.

Kontrol akses ke sumber daya Amazon ECS menggunakan tag sumber daya

Saat membuat kebijakan IAM yang memberikan izin kepada pengguna untuk menggunakan sumber daya Amazon ECS, Anda dapat menyertakan informasi tag dalam `Condition` elemen kebijakan untuk mengontrol akses berdasarkan tag. Hal ini dikenal sebagai kendali akses berbasis atribut (ABAC). ABAC memberikan Anda kendali yang lebih baik atas sumber daya mengenai sumber daya mana yang dapat diubah, digunakan, atau dihapus oleh seorang pengguna. Untuk informasi lebih lanjut, lihat Apa fungsi ABAC untuk AWS?

Misalnya, Anda dapat membuat kebijakan yang memungkinkan pengguna menghapus klaster, tetapi menolak tindakan jika klaster memiliki tag`environment=production`. Untuk melakukan hal ini, Anda bisa menggunakan kunci syarat `aws:ResourceTag` untuk mengizinkan atau menolak akses ke sumber daya berdasarkan tanda yang dilampirkan pada sumber daya.

```
"StringEquals": { "aws:ResourceTag/environment": "production" }
```

Untuk mempelajari apakah tindakan Amazon ECS API mendukung pengendalian akses menggunakan kunci `aws:ResourceTag` kondisi, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon ECS. Perhatikan bahwa tindakan `Describe` tidak mendukung izin tingkat sumber daya, sehingga Anda harus menentukannya dalam pernyataan terpisah yang tidak disertai syarat.

Untuk contoh kebijakan IAM, lihat Amazon ECS Contoh kebijakan .

Jika Anda mengizinkan atau menolak akses para pengguna ke sumber daya berdasarkan tanda, maka Anda harus mempertimbangkan untuk menolak secara eksplisit memberikan kemampuan kepada pengguna untuk menambahkan atau menghapus tanda tersebut dari sumber daya yang sama. Jika tidak, pengguna dapat mengakali pembatasan Anda dan mendapatkan akses atas sumber daya dengan melakukan modifikasi pada tanda dari sumber daya tersebut.

Amazon ECS Contoh kebijakan

Anda dapat menggunakan kebijakan IAM untuk memberikan izin kepada pengguna untuk melihat dan bekerja dengan sumber daya tertentu di konsol Amazon ECS. Anda dapat menggunakan contoh kebijakan di bagian sebelumnya; namun, kebijakan tersebut dirancang untuk permintaan yang dibuat dengan AWS CLI atau AWS SDK.

Contoh: Izinkan pengguna menghapus cluster Amazon ECS berdasarkan tag

Kebijakan berikut memungkinkan pengguna untuk menghapus cluster ketika tag memiliki key/value sepasang “Tujuan/Pengujian”.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DeleteCluster"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:ecs:us-east-1:111122223333:cluster/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Testing"
                }
            }
        }
    ]
}
```

Memecahkan masalah identitas dan akses Amazon Elastic Container Service

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon ECS dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon ECS
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECS saya
- Sumber daya pemecahan masalah tambahan

Saya tidak berwenang untuk melakukan tindakan di Amazon ECS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `ecs:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecs:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `ecs:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon ECS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon ECS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECS saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon ECS mendukung fitur-fitur ini, lihatBagaimana Amazon Elastic Container Service bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Sumber daya pemecahan masalah tambahan

Halaman-halaman berikut memberikan informasi tentang kode kesalahan:

- Amazon ECS menghentikan pesan kesalahan tugas
- Melihat pesan acara layanan Amazon ECS

Praktik terbaik IAM untuk Amazon ECS

Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk mengelola dan mengontrol akses ke AWS layanan dan sumber daya Anda melalui kebijakan berbasis aturan untuk tujuan otentikasi dan otorisasi. Lebih khusus lagi, melalui layanan ini, Anda mengontrol akses ke

AWS sumber daya Anda dengan menggunakan kebijakan yang diterapkan pada pengguna, grup, atau peran. Di antara ketiganya, pengguna adalah akun yang dapat memiliki akses ke sumber daya Anda. Dan, peran IAM adalah sekumpulan izin yang dapat diasumsikan oleh identitas yang diautentikasi, yang tidak terkait dengan identitas tertentu di luar IAM. Untuk informasi selengkapnya, lihat ikhtisar Amazon ECS tentang manajemen akses: Izin dan kebijakan.

Ikuti kebijakan akses yang paling tidak memiliki hak istimewa

Buat kebijakan yang dicakup untuk memungkinkan pengguna melakukan pekerjaan yang ditentukan. Misalnya, jika pengembang perlu menghentikan tugas secara berkala, buat kebijakan yang hanya mengizinkan tindakan tertentu tersebut. Contoh berikut hanya memungkinkan pengguna untuk menghentikan tugas yang dimiliki oleh tertentu `task_family` di klaster dengan Nama Sumber Daya Amazon (ARN) tertentu. Mengacu pada ARN dalam suatu kondisi juga merupakan contoh penggunaan izin tingkat sumber daya. Anda dapat menggunakan izin tingkat sumber daya untuk menentukan sumber daya yang Anda inginkan untuk diterapkan tindakan.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StopTask"
            ],
            "Condition": {
                "ArnEquals": {
                       "ecs:cluster": "arn:aws:ecs:us-
east-1:111122223333:cluster/cluster_name"
                }
            },
            "Resource": [
                  "arn:aws:ecs:us-east-1:111122223333:task-
definition/task_family:*"
            ]
        }
    ]
}
```

Memiliki sumber daya cluster berfungsi sebagai batas administratif

Kebijakan yang terlalu sempit cakupan dapat menyebabkan proliferasi peran dan meningkatkan overhead administratif. Daripada membuat peran yang tercakup pada tugas atau layanan tertentu saja, buat peran yang tercakup ke cluster dan gunakan klaster sebagai batas administratif utama Anda.

Buat pipeline otomatis untuk mengisolasi pengguna akhir dari API

Anda dapat membatasi tindakan yang dapat digunakan pengguna dengan membuat pipeline yang secara otomatis mengemas dan menyebarkan aplikasi ke kluster Amazon ECS. Ini secara efektif mendelegasikan tugas membuat, memperbarui, dan menghapus tugas ke pipeline. Untuk informasi selengkapnya, lihat Tutorial: Penerapan standar Amazon ECS dengan CodePipeline di AWS CodePipeline Panduan Pengguna.

Gunakan kondisi kebijakan untuk lapisan keamanan tambahan

Jika Anda membutuhkan lapisan keamanan tambahan, tambahkan kondisi ke kebijakan Anda. Ini dapat berguna jika Anda melakukan operasi istimewa atau ketika Anda perlu membatasi serangkaian tindakan yang dapat dilakukan terhadap sumber daya tertentu. Contoh kebijakan berikut memerlukan otorisasi multi-faktor saat menghapus klaster.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      },
    "Resource": ["*"]
    }
  ]
```

```
}
```

Tag yang diterapkan pada layanan disebarkan ke semua tugas yang merupakan bagian dari layanan itu. Karena itu, Anda dapat membuat peran yang tercakup ke sumber daya Amazon ECS dengan tag tertentu. Dalam kebijakan berikut, prinsipal IAM memulai dan menghentikan semua tugas dengan kunci tag `Department` dan nilai tag. `Accounting`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Resource": "arn:aws:ecs:arn:aws::111122223333:task/my-cluster/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Department": "Accounting"}
            }
        }
    ]
}
```

Secara berkala mengaudit akses ke APIs

Seorang pengguna dapat mengubah peran. Setelah mereka mengubah peran, izin yang sebelumnya diberikan kepada mereka mungkin tidak lagi berlaku. Pastikan Anda mengaudit siapa yang memiliki akses ke Amazon ECS APIs dan apakah akses itu masih diperlukan. Pertimbangkan untuk mengintegrasikan IAM dengan solusi manajemen siklus hidup pengguna yang secara otomatis mencabut akses saat pengguna meninggalkan organisasi. Untuk informasi selengkapnya, lihat pedoman audit AWS keamanan di Panduan AWS Identity and Access Management Pengguna.

Logging dan Monitoring di Amazon Elastic Container Service

Pemantauan merupakan bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon Elastic Container Service dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. AWS menyediakan beberapa alat untuk memantau sumber daya Amazon ECS Anda dan menanggapi potensi insiden:

CloudWatch Alarm Amazon

Melihat metrik tunggal selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama sejumlah periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon Auto EC2 Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pantau Amazon ECS menggunakan CloudWatch.

Untuk layanan dengan tugas yang menggunakan tipe peluncuran Fargate, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil tugas dalam layanan Anda berdasarkan CloudWatch metrik, seperti CPU dan pemanfaatan memori. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.

Untuk cluster dengan tugas atau layanan yang menggunakan tipe EC2 peluncuran, Anda dapat menggunakan CloudWatch alarm untuk menskalakan dan memperkecil instance container berdasarkan CloudWatch metrik, seperti reservasi memori cluster.

CloudWatch Log Amazon

Pantau, simpan, dan akses file log dari kontainer di tugas Amazon ECS Anda dengan menentukan driver `awslogs` log dalam definisi tugas Anda. Untuk informasi selengkapnya, lihat Menggunakan driver awslogs.

Anda juga dapat memantau, menyimpan, dan mengakses sistem operasi dan file log agen penampung Amazon ECS dari instans penampung Amazon ECS Anda. Metode ini untuk mengakses log dapat digunakan untuk kontainer menggunakan tipe EC2 peluncuran..

CloudWatch Acara Amazon

Mencocokkan peristiwa dan merutekan mereka ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif.

Untuk informasi lebih lanjut, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge di panduan ini dan EventBridge merupakan evolusi CloudWatch Acara Amazon di Panduan EventBridge Pengguna Amazon.

AWS CloudTrail Log

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon ECS. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon ECS, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat Log panggilan Amazon ECS API menggunakan AWS CloudTrail.

AWS Trusted Advisor

Trusted Advisor mengacu pada praktik terbaik yang dipelajari dari melayani ratusan ribu AWS pelanggan. Trusted Advisor memeriksa AWS lingkungan Anda dan kemudian membuat rekomendasi ketika ada peluang untuk menghemat uang, meningkatkan ketersediaan dan kinerja sistem, atau membantu menutup kesenjangan keamanan. Semua AWS pelanggan memiliki akses ke lima Trusted Advisor cek. Pelanggan dengan paket dukungan Bisnis atau Perusahaan dapat melihat semua Trusted Advisor pemeriksaan.

Untuk informasi selengkapnya, lihat AWS Trusted Advisor di Dukungan Panduan Pengguna.

AWS Compute Optimizer

AWS Compute Optimizer adalah layanan yang menganalisis konfigurasi dan metrik pemanfaatan sumber daya Anda. AWS Ini melaporkan apakah sumber daya Anda optimal, dan menghasilkan rekomendasi pengoptimalan untuk mengurangi biaya dan meningkatkan kinerja beban kerja Anda.

Untuk informasi selengkapnya, lihat AWS Compute Optimizer rekomendasi untuk Amazon ECS.

Bagian penting lainnya dari pemantauan Amazon ECS melibatkan pemantauan secara manual item-item yang tidak CloudWatch tercakup oleh alarm. Dasbor AWS konsol CloudWatch Trusted Advisor,, dan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda untuk memeriksa file log pada instans kontainer Anda dan kontainer dalam tugas Anda.

Validasi kepatuhan untuk Amazon Elastic Container Service

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Kepatuhan dan Tata Kelola Keamanan – Panduan implementasi solusi ini membahas pertimbangan arsitektur serta memberikan langkah-langkah untuk menerapkan fitur keamanan dan kepatuhan.
- Referensi Layanan yang Memenuhi Syarat HIPAA — Daftar layanan yang memenuhi syarat HIPAA. Tidak semua memenuhi Layanan AWS syarat HIPAA.
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas yang mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan

kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.

- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Praktik terbaik kepatuhan dan keamanan untuk Amazon ECS

Tanggung jawab kepatuhan Anda saat menggunakan Amazon ECS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS)

Penting bagi Anda untuk memahami aliran lengkap data pemegang kartu (PJK) dalam lingkungan saat mengikuti PCI DSS. Aliran PJK menentukan penerapan PCI DSS, mendefinisikan batas dan komponen lingkungan data pemegang kartu (CDE), dan oleh karena itu ruang lingkup penilaian PCI DSS. Penentuan cakupan PCI DSS yang akurat adalah kunci untuk mendefinisikan postur keamanan dan akhirnya penilaian yang berhasil. Pelanggan harus memiliki prosedur untuk penentuan ruang lingkup yang menjamin kelengkapannya dan mendeteksi perubahan atau penyimpangan dari ruang lingkup.

Sifat sementara aplikasi kontainer memberikan kompleksitas tambahan saat mengaudit konfigurasi. Akibatnya, pelanggan perlu menjaga kesadaran akan semua parameter konfigurasi kontainer untuk memastikan persyaratan kepatuhan ditangani di semua fase siklus hidup kontainer.

Untuk informasi tambahan tentang mencapai kepatuhan PCI DSS di Amazon ECS, lihat whitepaper berikut.

- Arsitektur di Amazon ECS untuk kepatuhan PCI DSS

HIPAA (Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS)

Menggunakan Amazon ECS dengan beban kerja yang memproses informasi kesehatan yang dilindungi (PHI) tidak memerlukan konfigurasi tambahan. Amazon ECS bertindak sebagai layanan orkestrasi yang mengoordinasikan peluncuran kontainer di Amazon. EC2 Itu tidak beroperasi dengan atau pada data dalam beban kerja yang diatur. Konsisten dengan peraturan HIPAA dan Adendum Asosiasi AWS Bisnis, PHI harus dienkripsi saat transit dan istirahat saat diakses oleh kontainer yang diluncurkan dengan Amazon ECS.

Berbagai mekanisme untuk mengenkripsi saat istirahat tersedia dengan setiap opsi AWS penyimpanan, seperti Amazon S3, Amazon EBS, dan. AWS KMS Anda dapat menggunakan jaringan overlay (seperti VNS3 atau Weave Net) untuk memastikan enkripsi lengkap PHI yang ditransfer antar kontainer atau untuk menyediakan lapisan enkripsi yang berlebihan. Anda juga harus menggunakan logging lengkap, dan mengarahkan semua log kontainer ke Amazon CloudWatch. Untuk informasi tentang penggunaan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

AWS Security Hub

Gunakan AWS Security Hub. Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.

Amazon GuardDuty dengan Pemantauan Runtime Amazon ECS

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Gunakan Runtime Monitoring GuardDuty untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring melindungi beban kerja yang berjalan di Fargate dan EC2 dengan terus memantau aktivitas AWS log dan jaringan untuk mengidentifikasi perilaku berbahaya atau tidak sah. Runtime Monitoring menggunakan agen GuardDuty keamanan ringan yang dikelola sepenuhnya yang menganalisis perilaku on-host, seperti akses file, eksekusi proses, dan koneksi jaringan. Ini mencakup masalah termasuk peningkatan hak istimewa, penggunaan kredensil yang terbuka, atau komunikasi dengan alamat IP berbahaya, domain, dan keberadaan malware di instans Amazon EC2 dan beban kerja kontainer Anda. Untuk informasi selengkapnya, lihat GuardDuty Runtime Monitoring di Panduan GuardDuty Pengguna.

Rekomendasi kepatuhan

Kami menyarankan Anda melibatkan pemilik program kepatuhan dalam bisnis Anda lebih awal dan menggunakan model tanggung jawab AWS bersama untuk mengidentifikasi kepemilikan kontrol

kepatuhan agar berhasil dengan program kepatuhan yang relevan. Untuk informasi selengkapnya, lihat AWS model tanggung jawab bersama untuk Amazon ECS .

AWS Fargate Standar Pemrosesan Informasi Federal (FIPS-140)

Standar Pemrosesan Informasi Federal (FIPS). FIPS-140 adalah standar pemerintah AS dan Kanada yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. FIPS-140 mendefinisikan satu set fungsi kriptografi yang divalidasi yang dapat digunakan untuk mengenkripsi data dalam transit dan data saat istirahat.

Saat Anda mengaktifkan kepatuhan FIPS-140, Anda dapat menjalankan beban kerja di Fargate dengan cara yang sesuai dengan FIPS-140. Untuk informasi selengkapnya tentang kepatuhan FIPS-140, lihat Federal Information Processing Standard (FIPS) 140-3.

AWS Fargate Pertimbangan FIPS-140

Pertimbangkan hal berikut saat menggunakan kepatuhan FIPS-140 pada Fargate:

- Kepatuhan FIPS-140 hanya tersedia di Wilayah. AWS GovCloud (US)
- Fargate mendukung FIPS-140 versi 140.3
- Kepatuhan FIPS-140 dimatikan secara default. Anda harus menyalakannya.
- Tugas Anda harus menggunakan konfigurasi berikut untuk kepatuhan FIPS-140:
  - `operatingSystemFamily`Harus`LINUX`.
  - `cpuArchitecture`Harus`X86_64`.
  - Versi platform Fargate harus `1.4.0` atau lebih baru.

Gunakan FIPS di Fargate

Gunakan prosedur berikut untuk menggunakan kepatuhan FIPS-140 pada Fargate.

1. Nyalakan kepatuhan FIPS-140. Untuk informasi selengkapnya, lihat the section called “AWS Fargate Kepatuhan Standar Pemrosesan Informasi Federal (FIPS-140)”.
2. Anda dapat menggunakan ECS Exec secara opsional untuk menjalankan perintah berikut untuk memverifikasi status kepatuhan FIPS-140 untuk sebuah cluster.

Ganti *cluster-name* dengan nama cluster Anda, *task-id* dengan ID atau ARN tugas Anda, dan *container-name* dengan nama wadah dalam tugas Anda, Anda ingin menjalankan perintah terhadap.

Nilai pengembalian “1" menunjukkan bahwa Anda menggunakan FIPS.

```
aws ecs execute-command \
    --cluster cluster-name \
    --task task-id \
    --container container-name \
    --interactive \
    --command "cat /proc/sys/crypto/fips_enabled"
```

Gunakan CloudTrail untuk audit Fargate FIPS-140

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Saat aktivitas API dan konsol terjadi di Amazon ECS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Peristiwa. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk peristiwa untuk Amazon ECS, buat jejak yang CloudTrail digunakan untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol, jejak ini diterapkan ke semua Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat the section called “Log panggilan Amazon ECS API menggunakan AWS CloudTrail”.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan tindakan `PutAccountSettingDefault` API:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIV5AJI5LXF5EXAMPLE",
```

```
          "arn": "arn:aws:iam::123456789012:user/jdoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIPWIOFC3EXAMPLE",
    },
    "eventTime": "2023-03-01T21:45:18Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "PutAccountSettingDefault",
    "awsRegion": "us-gov-east-1",
    "sourceIPAddress": "52.94.133.131",
    "userAgent": "aws-cli/2.9.8 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
ecs.put-account-setting",
    "requestParameters": {
        "name": "fargateFIPSMode",
        "value": "enabled"
    },
    "responseElements": {
        "setting": {
            "name": "fargateFIPSMode",
            "value": "enabled",
            "principalArn": "arn:aws:iam::123456789012:user/jdoe"
        }
    },
    "requestID": "acdc731e-e506-447c-965d-f5f75EXAMPLE",
    "eventID": "6afced68-75cd-4d44-8076-0beEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "ecs-fips.us-gov-east-1.amazonaws.com"
    }
}
```


Keamanan Infrastruktur di Amazon Elastic Container Service

Sebagai layanan terkelola, Amazon Elastic Container Service dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan

praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon ECS melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun. Amazon ECS mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber, jadi pastikan kebijakan tersebut memperhitungkan alamat IP untuk lokasi jaringan. Anda juga dapat menggunakan kebijakan Amazon ECS untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud tertentu atau spesifik. VPCs Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon ECS tertentu hanya dari VPC tertentu dalam jaringan. AWS Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink.

Titik akhir VPC antarmuka Amazon ECS ()AWS PrivateLink

Anda dapat meningkatkan postur keamanan VPC Anda dengan mengonfigurasi Amazon ECS untuk menggunakan titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses Amazon ECS secara pribadi APIs dengan menggunakan alamat IP pribadi. AWS PrivateLink membatasi semua lalu lintas jaringan antara VPC Anda dan Amazon ECS ke jaringan Amazon. Anda tidak memerlukan sebuah gateway internet, perangkat NAT, atau gateway privat virtual.

Untuk informasi selengkapnya tentang AWS PrivateLink dan titik akhir VPC, lihat titik akhir VPC di Panduan Pengguna Amazon VPC.

Pertimbangan

Pertimbangan untuk titik akhir di Wilayah diperkenalkan mulai 23 Desember 2023

Sebelum Anda mengatur titik akhir VPC antarmuka untuk Amazon ECS, perhatikan pertimbangan berikut:

- Anda harus memiliki titik akhir VPC khusus Wilayah berikut:

  > **Note**
  >
  > Jika Anda tidak mengonfigurasi semua titik akhir, lalu lintas Anda akan melewati titik akhir publik, bukan titik akhir VPC Anda.

  - `com.amazonaws.region.ecs-agent`
  - `com.amazonaws.region.ecs-telemetry`
  - `com.amazonaws.region.ecs`

  Misalnya, Wilayah Kanada Barat (Calgary) (ca-west-1) membutuhkan titik akhir VPC berikut:

  - `com.amazonaws.ca-west-1.ecs-agent`
  - `com.amazonaws.ca-west-1.ecs-telemetry`
  - `com.amazonaws.ca-west-1.ecs`
- Saat Anda menggunakan templat untuk membuat AWS sumber daya di Wilayah baru dan templat disalin dari Wilayah yang diperkenalkan sebelum 23 Desember 2023, tergantung pada salinan-dari Wilayah, lakukan salah satu operasi berikut.

  Misalnya, copy-from Region adalah US East (Virginia N.) (us-east-1). Wilayah copy-to adalah Kanada Barat (Calgary) (ca-west-1).

Konfigurasi	Tindakan	
Wilayah yang disalin-dari tidak memiliki titik akhir VPC.	Buat ketiga titik akhir VPC untuk Wilayah baru (misalnya ,). `com.amazonaws.ca-west-1.ecs-agent`	

Konfigurasi	Tindakan	
Wilayah yang disalin dari berisi titik akhir VPC khusus Wilayah.	a. Buat ketiga titik akhir VPC untuk Wilayah baru (misalnya,). `com.amazonaws.ca-west-1.ecs-agent` b. Hapus ketiga titik akhir VPC untuk salinan-dari Wilayah (misalnya,). `com.amazonaws.us-east-1.ecs-agent`	

Pertimbangan untuk titik akhir Amazon ECS VPC untuk jenis peluncuran Fargate

Ketika ada titik akhir VPC untuk `ecr.dkr` dan `ecr.api` di VPC yang sama di mana tugas Fargate diterapkan, itu akan menggunakan titik akhir VPC. Jika tidak ada titik akhir VPC, itu akan menggunakan antarmuka Fargate.

Sebelum Anda mengatur titik akhir VPC antarmuka untuk Amazon ECS, perhatikan pertimbangan berikut:

- Tugas yang menggunakan tipe peluncuran Fargate tidak memerlukan titik akhir VPC antarmuka untuk Amazon ECS, tetapi Anda mungkin memerlukan titik akhir VPC antarmuka untuk Amazon ECR, Secrets Manager, atau Amazon Logs yang dijelaskan dalam poin berikut. CloudWatch
  - Untuk memungkinkan tugas Anda menarik gambar pribadi dari Amazon ECR, Anda harus membuat titik akhir VPC antarmuka untuk Amazon ECR. Untuk informasi selengkapnya, lihat Antarmuka VPC Endpoints (AWS PrivateLink) di Panduan Pengguna Amazon Elastic Container Registry.

    > **Important**
    >
    > Jika Anda mengonfigurasi Amazon ECR untuk menggunakan titik akhir VPC antarmuka, Anda dapat membuat peran eksekusi tugas yang menyertakan kunci kondisi untuk membatasi akses ke titik akhir VPC atau VPC tertentu. Untuk informasi selengkapnya, lihat Tugas Fargate menarik gambar Amazon ECR melalui izin titik akhir antarmuka.

  - Untuk memungkinkan tugas Anda menarik data sensitif dari Secrets Manager, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .
  - Jika VPC Anda tidak memiliki gateway internet dan tugas Anda menggunakan driver `awslogs` log untuk mengirim informasi log ke Log, Anda harus membuat antarmuka VPC endpoint untuk CloudWatch Log. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch Logs.
- VPC endpoint saat ini tidak mendukung permintaan lintas Wilayah. Pastikan Anda membuat titik akhir di Wilayah yang sama tempat Anda berencana mengeluarkan panggilan API ke Amazon ECS. Misalnya, asumsikan bahwa Anda ingin menjalankan tugas di AS Timur (Virginia N.). Kemudian, Anda harus membuat titik akhir Amazon ECS VPC di US East (Virginia N.). Titik akhir VPC Amazon ECS yang dibuat di wilayah lain mana pun tidak dapat menjalankan tugas di AS Timur (Virginia Utara).
- Titik akhir VPC hanya mendukung DNS yang disediakan Amazon melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.
- Grup keamanan yang terpasang pada titik akhir VPC harus mengizinkan koneksi masuk pada port TCP 443 dari subnet pribadi VPC.
- Manajemen Service Connect dari proxy Envoy menggunakan titik akhir VPC`com.amazonaws.region.ecs-agent`. Bila Anda tidak menggunakan endpoint VPC, manajemen Service Connect dari proxy Envoy menggunakan endpoint di Region tersebut`ecs-sc`. Untuk daftar titik akhir Amazon ECS di setiap Wilayah, lihat titik akhir dan kuota Amazon ECS.

Pertimbangan untuk titik akhir Amazon ECS VPC untuk jenis peluncuran EC2

Sebelum Anda mengatur titik akhir VPC antarmuka untuk Amazon ECS, perhatikan pertimbangan berikut:

- Tugas yang menggunakan tipe EC2 peluncuran mengharuskan instance container yang diluncurkan untuk menjalankan versi `1.25.1` atau yang lebih baru dari agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Manajemen instans penampung Amazon ECS Linux.
- Untuk memungkinkan tugas Anda menarik data sensitif dari Secrets Manager, Anda harus membuat titik akhir VPC antarmuka untuk Secrets Manager. Untuk informasi selengkapnya, lihat Menggunakan Secrets Manager dengan VPC Endpoint dalam Panduan Pengguna AWS Secrets Manager .

- Jika VPC Anda tidak memiliki gateway internet dan tugas Anda menggunakan driver `awslogs` log untuk mengirim informasi log ke Log, Anda harus membuat antarmuka VPC endpoint untuk CloudWatch Log. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch Logs.
- VPC endpoint saat ini tidak mendukung permintaan lintas Wilayah. Pastikan Anda membuat titik akhir di Wilayah yang sama tempat Anda berencana mengeluarkan panggilan API ke Amazon ECS. Misalnya, asumsikan bahwa Anda ingin menjalankan tugas di AS Timur (Virginia N.). Kemudian, Anda harus membuat titik akhir Amazon ECS VPC di US East (Virginia N.). Titik akhir VPC Amazon ECS yang dibuat di Wilayah lain mana pun tidak dapat menjalankan tugas di AS Timur (Virginia Utara).
- Titik akhir VPC hanya mendukung DNS yang disediakan Amazon melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.
- Grup keamanan yang terpasang pada titik akhir VPC harus mengizinkan koneksi masuk pada port TCP 443 dari subnet pribadi VPC.

Memahami pola penamaan titik akhir Amazon ECS

Penting untuk dipahami bahwa agen Amazon ECS dapat mengajukan permintaan ke titik akhir dengan sufiks bernomor, seperti:

- `ecs-a-1.region.amazonaws.com`,`ecs-a-2.region.amazonaws.com`, dll. untuk titik akhir agen
- `ecs-t-1.region.amazonaws.com`,`ecs-t-2.region.amazonaws.com`, dll. untuk titik akhir telemetri

Perilaku ini terjadi karena agen Amazon ECS menggunakan DiscoverPollEndpointAPI untuk secara dinamis menentukan titik akhir spesifik mana yang akan disambungkan. Jika titik akhir VPC Anda tidak menangani variasi titik akhir bernomor ini dengan benar, agen akan kembali menggunakan titik akhir publik, bahkan jika Anda telah mengonfigurasi titik akhir VPC untuk nama dasar.

Peran DiscoverPollEndpoint API

DiscoverPollEndpointAPI digunakan oleh agen Amazon ECS untuk menemukan titik akhir yang sesuai untuk polling untuk tugas. Ketika agen memanggil API ini, ia menerima URL titik akhir tertentu yang mungkin menyertakan akhiran bernomor. Untuk memastikan titik akhir VPC Anda berfungsi dengan benar, konfigurasi jaringan Anda harus mengizinkan agen untuk:

1. Akses DiscoverPollEndpoint API
2. Connect ke endpoint yang dikembalikan URLs, termasuk yang memiliki sufiks bernomor

Jika Anda memecahkan masalah konektivitas titik akhir VPC, verifikasi bahwa agen Anda dapat mencapai titik akhir dasar dan variasi bernomor apa pun yang mungkin dikembalikan oleh API. DiscoverPollEndpoint

Membuat Endpoint VPC untuk Amazon ECS

Untuk membuat titik akhir VPC untuk layanan Amazon ECS, gunakan prosedur Access an AWS service menggunakan antarmuka VPC endpoint di Panduan Pengguna Amazon VPC untuk membuat titik akhir berikut. Jika Anda memiliki instans kontainer yang berada dalam VPC Anda, Anda harus membuat titik akhir dalam urutan terdaftar mereka. Urutan langkah tidak membuat masalah jika Anda berencana untuk membuat VPC endpoint terlebih dahulu kemudian instans kontainer Anda.

> **Note**
>
> Jika Anda tidak mengonfigurasi semua titik akhir, lalu lintas Anda akan melewati titik akhir publik, bukan titik akhir VPC Anda.
> Saat Anda membuat titik akhir, Amazon ECS juga membuat nama DNS pribadi untuk titik akhir. Misalnya, untuk ecs-agent dan `ecs-a.region.amazonaws.com` `ecs-t.region.amazonaws.com` untuk ecs-telemetri.

- `com.amazonaws.region.ecs-agent`
- `com.amazonaws.region.ecs-telemetry`
- `com.amazonaws.region.ecs`

> **Note**
>
> *region*mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh Amazon ECS, seperti `us-east-2` untuk Wilayah AS Timur (Ohio).

`ecs-agent`Titik akhir menggunakan `ecs:poll` API, dan `ecs-telemetry` titik akhir menggunakan API `ecs:poll` dan`ecs:StartTelemetrySession`.

Jika Anda memiliki tugas yang ada yang menggunakan tipe EC2 peluncuran, setelah Anda membuat titik akhir VPC, setiap instance container perlu mengambil konfigurasi baru. Agar hal ini terjadi, Anda harus me-reboot setiap instance kontainer atau memulai ulang agen penampung Amazon ECS pada setiap instance container. Untuk memulai ulang agen kontainer, lakukan hal berikut.

Untuk memulai ulang agen kontainer Amazon ECS

1. Masuk ke instans kontainer Anda melalui SSH.
2. Hentikan kontainer agen.

```
sudo docker stop ecs-agent
```

3. Mulai kontainer agen.

```
sudo docker start ecs-agent
```

Setelah Anda membuat titik akhir VPC dan memulai ulang agen penampung Amazon ECS di setiap instance container, semua tugas yang baru diluncurkan akan mengambil konfigurasi baru.

Membuat kebijakan titik akhir VPC untuk Amazon ECS

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC yang mengontrol akses ke Amazon ECS. Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan Amazon ECS

Berikut ini adalah contoh kebijakan endpoint untuk Amazon ECS. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke izin untuk membuat dan mencantumkan klaster. `ListClusters`Tindakan `CreateCluster` dan tidak menerima sumber daya apa pun, sehingga definisi sumber daya diatur ke* untuk semua sumber daya.

```
{
   "Statement":[
    {
      "Principal":"*",
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

AWS model tanggung jawab bersama untuk Amazon ECS

Keamanan dan Kepatuhan adalah tanggung jawab bersama antara AWS dan pelanggan. Model bersama ini dapat membantu meringankan beban operasional pelanggan saat AWS mengoperasikan, mengelola, dan mengontrol komponen dari sistem operasi host dan lapisan virtualisasi hingga keamanan fisik fasilitas tempat layanan beroperasi. Pelanggan memikul tanggung jawab dan pengelolaan sistem operasi tamu (termasuk pembaruan dan patch keamanan), perangkat lunak aplikasi terkait lainnya serta konfigurasi firewall grup keamanan yang AWS disediakan. Pelanggan harus dengan cermat mempertimbangkan layanan yang mereka pilih karena tanggung jawab mereka sangat bergantung pada layanan yang digunakan, integrasi dari layanan tersebut ke dalam lingkungan IT mereka, serta undang-undang dan regulasi yang berlaku. Pada dasarnya, tanggung jawab bersama ini juga menyediakan kontrol pelanggan dan fleksibilitas yang mengizinkan deployment.

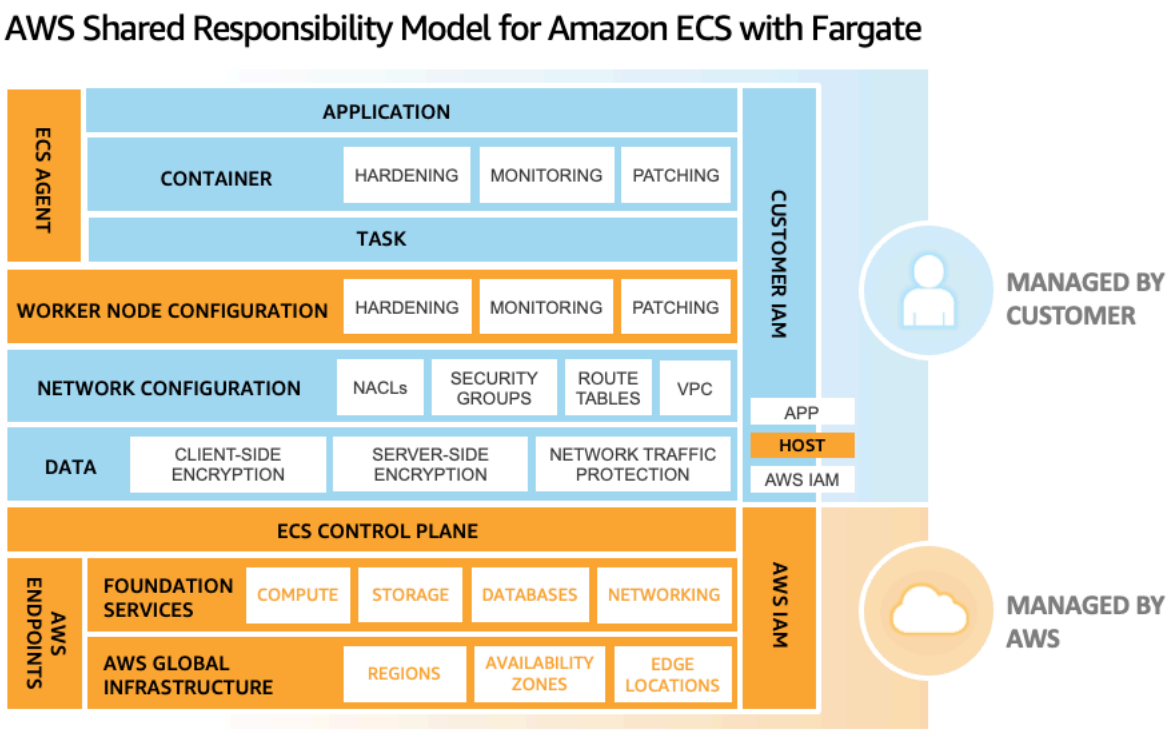
Jenis peluncuran Fargate

Ilustrasi berikut menunjukkan model tanggung jawab bersama untuk jenis peluncuran Fargate. Fargate menjalankan setiap beban kerja dalam lingkungan virtual perangkat keras yang terisolasi. Akibatnya, setiap tugas mendapat kapasitas infrastruktur khusus. Beban kerja kontainer yang berjalan di Fargate tidak berbagi sistem operasi, kernel Linux, antarmuka jaringan, penyimpanan sementara, CPU, atau memori dengan tugas lain. Saat menggunakan Fargate, pelanggan tidak bertanggung jawab untuk mengamankan infrastruktur komputasi yang menjalankan kontainer mereka. Fargate akan menyediakan dan menambal infrastruktur tempat beban kerja pelanggan

berjalan. Untuk informasi selengkapnya, lihat Tugas pensiun dan pemeliharaan untuk AWS Fargate di Amazon ECS .

Anda bertanggung jawab untuk mengelola sumber daya berikut:

- Konfigurasi jaringan termasuk VPC, NACLs, grup keamanan, dan tabel rute
- Enkripsi penyimpanan klien dan layanan. Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.
- Gambar kontainer. Untuk informasi selengkapnya, lihat Praktik terbaik tugas Amazon ECS dan keamanan kontainer.
- Izin IAM untuk aplikasi dengan menggunakan peran tugas. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.

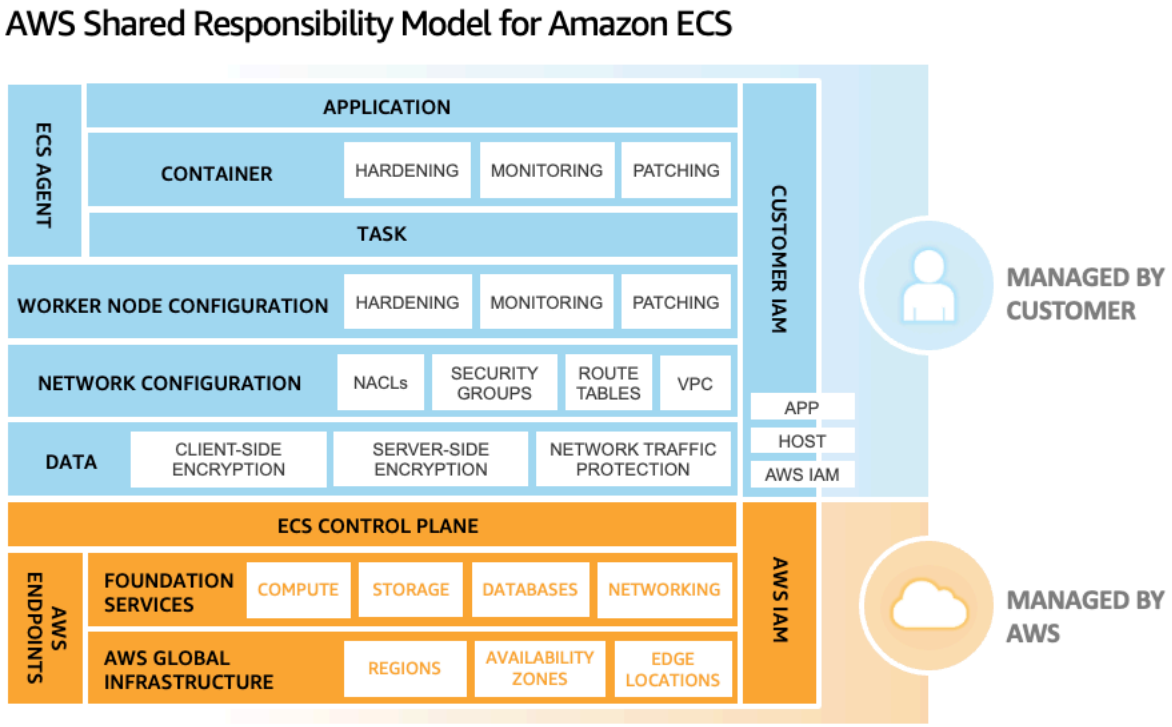


EC2 jenis peluncuran

Ilustrasi berikut menunjukkan tanggung jawab bersama untuk jenis EC2 peluncuran. Saat menjalankan tugas pada EC2 instans, Anda bertanggung jawab untuk menjaga EC2 instans Anda selain sumber daya berikut:

- Agen Amazon ECS.

- • EC2 Contoh AMI, termasuk penambalan dan pengerasan.
- Konfigurasi jaringan termasuk VPC, NACLs, grup keamanan, dan tabel rute.
- Enkripsi penyimpanan klien dan layanan. Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.
- Gambar kontainer. Untuk informasi selengkapnya, lihat Praktik terbaik tugas Amazon ECS dan keamanan kontainer.
- Izin IAM untuk aplikasi dengan menggunakan peran tugas. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.



Praktik terbaik keamanan jaringan untuk Amazon ECS

Keamanan jaringan adalah topik luas yang mencakup beberapa subtopik. Ini termasuk encryption-in-transit, segmentasi dan isolasi jaringan, firewall, perutean lalu lintas, dan observabilitas.

Enkripsi bergerak

Mengenkripsi lalu lintas jaringan mencegah pengguna yang tidak sah mencegat dan membaca data ketika data tersebut ditransmisikan melalui jaringan. Dengan Amazon ECS, enkripsi jaringan dapat diimplementasikan dengan salah satu cara berikut.

- Menggunakan instance Nitro:

  Secara default, lalu lintas secara otomatis dienkripsi antara jenis instans Nitro berikut: C5n, G4, i3en, M5dn, M5n, P3dn, R5dn, dan R5n. Lalu lintas tidak dienkripsi saat dialihkan melalui gateway transit, penyeimbang beban, atau perantara serupa.

  - Enkripsi dalam perjalanan
  - Apa pengumuman baru dari 2019
  - Pembicaraan ini dari re:Inforce 2019
- Menggunakan Server Name Indication (SNI) dengan Application Load Balancer:

  Application Load Balancer (ALB) dan Network Load Balancer (NLB) mendukung Server Name Indication (SNI). Dengan menggunakan SNI, Anda dapat menempatkan beberapa aplikasi aman di belakang satu pendengar. Untuk ini, masing-masing memiliki sertifikat TLS sendiri. Kami menyarankan Anda memberikan sertifikat untuk load balancer using AWS Certificate Manager (ACM) dan kemudian menambahkannya ke daftar sertifikat pendengar. AWS Load balancer menggunakan algoritma pemilihan sertifikat cerdas dengan SNI. Jika nama host yang disediakan oleh klien cocok dengan satu sertifikat dalam daftar sertifikat, penyeimbang beban memilih sertifikat tersebut. Jika nama host yang disediakan oleh klien cocok dengan beberapa sertifikat dalam daftar, penyeimbang beban memilih sertifikat yang dapat didukung klien. Contohnya termasuk sertifikat yang ditandatangani sendiri atau sertifikat yang dihasilkan melalui ACM.

  - SNI dengan Application Load Balancer
  - SNI dengan Network Load Balancer
- End-to-end enkripsi dengan sertifikat TLS:

  Ini melibatkan penerapan sertifikat TLS dengan tugas. Ini bisa berupa sertifikat yang ditandatangani sendiri atau sertifikat dari otoritas sertifikat tepercaya. Anda dapat memperoleh sertifikat dengan mereferensikan rahasia untuk sertifikat. Jika tidak, Anda dapat memilih untuk menjalankan kontainer yang mengeluarkan Permintaan Penandatanganan Sertifikat (CSR) ke ACM dan kemudian memasang rahasia yang dihasilkan ke volume bersama.

  - Menjaga keamanan lapisan transport sampai ke kontainer Anda menggunakan Network Load Balancer dengan Amazon ECS bagian 1
  - Mempertahankan Transport Layer Security (TLS) sampai ke container Anda bagian 2: Menggunakan AWS Private Certificate Authority

Jaringan tugas

Rekomendasi berikut ini mempertimbangkan cara kerja Amazon ECS. Amazon ECS tidak menggunakan jaringan overlay. Sebagai gantinya, tugas dikonfigurasi untuk beroperasi dalam mode jaringan yang berbeda. Misalnya, tugas yang dikonfigurasi untuk menggunakan `bridge` mode memperoleh alamat IP yang tidak dapat dirutekan dari jaringan Docker yang berjalan di setiap host. Tugas yang dikonfigurasi untuk menggunakan mode `awsvpc` jaringan memperoleh alamat IP dari subnet host. Tugas yang dikonfigurasi dengan `host` jaringan menggunakan antarmuka jaringan host. `awsvpc`adalah mode jaringan yang disukai. Ini karena ini adalah satu-satunya mode yang dapat Anda gunakan untuk menetapkan grup keamanan ke tugas. Ini juga satu-satunya mode yang tersedia untuk AWS Fargate tugas-tugas di Amazon ECS.

Grup keamanan untuk tugas

Kami menyarankan Anda mengonfigurasi tugas Anda untuk menggunakan mode `awsvpc` jaringan. Setelah Anda mengonfigurasi tugas Anda untuk menggunakan mode ini, agen Amazon ECS secara otomatis menyediakan dan melampirkan Antarmuka Jaringan Elastis (ENI) ke tugas tersebut. Ketika ENI disediakan, tugas terdaftar dalam kelompok keamanan. AWS Grup keamanan bertindak sebagai firewall virtual yang dapat Anda gunakan untuk mengontrol lalu lintas masuk dan keluar.

Jika Anda menggunakan firewall khusus dengan tugas atau layanan, tambahkan aturan keluar untuk mengizinkan lalu lintas untuk titik akhir manajemen agen Amazon ECS (“”), titik akhir telemetri (`ecs-a-*.`*`region`*`.amazonaws.com`“”), dan titik akhir manajemen Utusan Layanan Connect (`ecs-t-*.`*`region`*`.amazonaws.com`“”). `ecs-sc.`*`region`*`.api.aws`

AWS PrivateLink dan Amazon ECS

AWS PrivateLink adalah teknologi jaringan yang memungkinkan Anda membuat titik akhir pribadi untuk berbagai AWS layanan, termasuk Amazon ECS. Titik akhir diperlukan di lingkungan kotak pasir di mana tidak ada Internet Gateway (IGW) yang terpasang ke VPC Amazon dan tidak ada rute alternatif ke Internet. Menggunakan AWS PrivateLink memastikan bahwa panggilan ke layanan Amazon ECS tetap berada dalam VPC Amazon dan tidak melintasi internet. Untuk petunjuk tentang cara membuat AWS PrivateLink titik akhir untuk Amazon ECS dan layanan terkait lainnya, lihat Antarmuka Amazon ECS Titik akhir Amazon VPC.

> **Important**
>
> AWS Fargate tugas tidak memerlukan AWS PrivateLink titik akhir untuk Amazon ECS.

Amazon ECR dan Amazon ECS keduanya mendukung kebijakan titik akhir. Kebijakan ini memungkinkan Anda untuk memperbaiki akses ke layanan. APIs Misalnya, Anda dapat membuat kebijakan endpoint untuk Amazon ECR yang hanya mengizinkan gambar didorong ke pendaftar di akun tertentu. AWS Kebijakan seperti ini dapat digunakan untuk mencegah data diekstraksi melalui gambar kontainer sambil tetap memungkinkan pengguna untuk mendorong ke pendaftar ECR Amazon resmi. Untuk informasi selengkapnya, lihat Menggunakan kebijakan titik akhir VPC.

Kebijakan berikut memungkinkan semua AWS prinsipal di akun Anda untuk melakukan semua tindakan terhadap hanya repositori Amazon ECR Anda:

```
{
  "Statement": [
    {
      "Sid": "LimitECRAccess",
      "Principal": "*",
      "Action": "*",
      "Effect": "Allow",
      "Resource": "arn:aws:ecr:region:account_id:repository/*"
    },
  ]
}
```

Anda dapat meningkatkan ini lebih lanjut dengan menetapkan kondisi yang menggunakan `PrincipalOrgID` properti baru. Ini mencegah mendorong dan menarik gambar oleh prinsip IAM yang bukan bagian dari Anda AWS Organizations. Untuk informasi selengkapnya, lihat aws: PrincipalOrg ID.

Kami merekomendasikan menerapkan kebijakan yang sama untuk kedua titik akhir `com.amazonaws.region.ecr.dkr` dan `com.amazonaws.region.ecr.api` titik akhir.

Pengaturan agen kontainer

File konfigurasi agen penampung Amazon ECS mencakup beberapa variabel lingkungan yang berhubungan dengan keamanan jaringan. `ECS_AWSVPC_BLOCK_IMDS`dan `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` digunakan untuk memblokir akses tugas ke EC2 metadata Amazon. `HTTP_PROXY`digunakan untuk mengkonfigurasi agen untuk rute melalui proxy HTTP untuk terhubung ke internet. Untuk petunjuk tentang mengonfigurasi agen dan runtime Docker untuk merutekan melalui proxy, lihat Konfigurasi Proxy HTTP.

> **Important**
>
> Pengaturan ini tidak tersedia saat Anda menggunakan AWS Fargate.

Rekomendasi keamanan jaringan

Kami menyarankan Anda melakukan hal berikut saat menyiapkan VPC Amazon, penyeimbang beban, dan jaringan Anda.

Gunakan enkripsi jaringan jika berlaku dengan Amazon ECS

Anda harus menggunakan enkripsi jaringan jika berlaku. Program kepatuhan tertentu, seperti PCI DSS, mengharuskan Anda mengenkripsi data dalam perjalanan jika data berisi data pemegang kartu. Jika beban kerja Anda memiliki persyaratan serupa, konfigurasikan enkripsi jaringan.

Browser modern memperingatkan pengguna saat menghubungkan ke situs yang tidak aman. Jika layanan Anda dipimpin oleh penyeimbang beban yang menghadap publik, gunakan TLS/SSL untuk mengenkripsi lalu lintas dari browser klien ke penyeimbang beban dan mengenkripsi ulang ke backend jika diperlukan.

Gunakan mode **`awsvpc`** jaringan dan grup keamanan untuk mengontrol lalu lintas antara tugas dan sumber daya lainnya di Amazon ECS

Anda harus menggunakan mode `awsvpc` jaringan dan grup keamanan ketika Anda perlu mengontrol lalu lintas antara tugas dan antara tugas dan sumber daya jaringan lainnya. Jika layanan Anda berada di belakang ALB, gunakan grup keamanan untuk hanya mengizinkan lalu lintas masuk dari sumber daya jaringan lain menggunakan grup keamanan yang sama dengan ALB Anda. Jika aplikasi Anda berada di belakang NLB, konfigurasikan grup keamanan tugas untuk hanya mengizinkan lalu lintas masuk dari rentang CIDR VPC Amazon dan alamat IP statis yang ditetapkan ke NLB.

Grup keamanan juga harus digunakan untuk mengontrol lalu lintas antara tugas dan sumber daya lain dalam VPC Amazon seperti database Amazon RDS.

Buat cluster Amazon ECS di Amazon terpisah VPCs saat lalu lintas jaringan perlu diisolasi secara ketat

Anda harus membuat cluster di Amazon terpisah VPCs ketika lalu lintas jaringan perlu diisolasi secara ketat. Hindari menjalankan beban kerja yang memiliki persyaratan keamanan ketat pada

cluster dengan beban kerja yang tidak harus mematuhi persyaratan tersebut. Jika isolasi jaringan yang ketat adalah wajib, buat cluster di Amazon terpisah VPCs dan secara selektif mengekspos layanan ke Amazon lain menggunakan titik akhir VPCs Amazon VPC. Untuk informasi selengkapnya, lihat titik akhir VPC.

Konfigurasikan AWS PrivateLink titik akhir saat diperlukan untuk Amazon ECS

Anda harus mengonfigurasi AWS PrivateLink titik akhir saat diperlukan. Jika kebijakan keamanan mencegah Anda melampirkan Internet Gateway (IGW) ke Amazon VPCs, konfigurasikan titik AWS PrivateLink akhir untuk Amazon ECS dan layanan lain seperti Amazon ECR, dan Amazon. AWS Secrets Manager CloudWatch

Gunakan Amazon VPC Flow Logs untuk menganalisis lalu lintas ke dan dari tugas yang berjalan lama di Amazon ECS

Anda harus menggunakan Amazon VPC Flow Logs untuk menganalisis lalu lintas ke dan dari tugas yang berjalan lama. Tugas yang menggunakan mode `awsvpc` jaringan mendapatkan ENI mereka sendiri. Dengan melakukan ini, Anda dapat memantau lalu lintas yang masuk ke dan dari tugas individual menggunakan Amazon VPC Flow Logs. Pembaruan terbaru untuk Amazon VPC Flow Logs (v3), memperkaya log dengan metadata lalu lintas termasuk ID vpc, ID subnet, dan ID instans. Metadata ini dapat digunakan untuk membantu mempersempit penyelidikan. Untuk informasi selengkapnya, lihat Amazon VPC Flow Logs.

> **Note**
>
> Karena sifat kontainer sementara, log aliran mungkin tidak selalu menjadi cara yang efektif untuk menganalisis pola lalu lintas antara kontainer atau kontainer yang berbeda dan sumber daya jaringan lainnya.

Praktik terbaik tugas Amazon ECS dan keamanan kontainer

Anda harus mempertimbangkan gambar kontainer sebagai garis pertahanan pertama Anda terhadap serangan. Gambar yang tidak aman dan dibangun dengan buruk dapat memungkinkan penyerang melarikan diri dari batas-batas wadah dan mendapatkan akses ke host. Anda harus melakukan hal berikut untuk mengurangi risiko terjadinya hal ini.

Kami menyarankan Anda melakukan hal berikut saat menyiapkan tugas dan wadah Anda.

Buat minimal atau gunakan gambar distroless

Mulailah dengan menghapus semua binari asing dari gambar kontainer. Jika Anda menggunakan gambar asing dari Amazon ECR Public Gallery, periksa gambar untuk merujuk ke konten setiap layer container. Anda dapat menggunakan aplikasi seperti Dive untuk melakukan ini.

Atau, Anda dapat menggunakan gambar distroless yang hanya menyertakan aplikasi Anda dan dependensi runtime-nya. Mereka tidak berisi manajer paket atau shell. Gambar distroless meningkatkan "sinyal ke noise pemindai dan mengurangi beban menetapkan asal sesuai dengan apa yang Anda butuhkan." Untuk informasi lebih lanjut, lihat GitHub dokumentasi tentang distroless.

Docker memiliki mekanisme untuk membuat gambar dari gambar minimal yang dicadangkan yang dikenal sebagai scratch. Untuk informasi selengkapnya, lihat Membuat gambar induk sederhana menggunakan scratch dalam dokumentasi Docker. Dengan bahasa seperti Go, Anda dapat membuat biner tertaut statis dan mereferensikannya di Dockerfile Anda. Contoh berikut menunjukkan bagaimana Anda dapat mencapai ini.

```
############################
# STEP 1 build executable binary
############################
FROM golang:alpine AS builder
# Install git.
# Git is required for fetching the dependencies.
RUN apk update && apk add --no-cache git
WORKDIR $GOPATH/src/mypackage/myapp/
COPY . .
# Fetch dependencies.
# Using go get.
RUN go get -d -v
# Build the binary.
RUN go build -o /go/bin/hello
############################
# STEP 2 build a small image
############################
FROM scratch
# Copy our static executable.
COPY --from=builder /go/bin/hello /go/bin/hello
# Run the hello binary.
ENTRYPOINT ["/go/bin/hello"]
This creates a container image that consists of your application and nothing else,
 making it extremely secure.
```

Contoh sebelumnya juga merupakan contoh build multi-tahap. Jenis build ini menarik dari sudut pandang keamanan karena Anda dapat menggunakannya untuk meminimalkan ukuran gambar akhir yang didorong ke registri kontainer Anda. Gambar kontainer tanpa alat build dan binari asing lainnya meningkatkan postur keamanan Anda dengan mengurangi permukaan serangan gambar. Untuk informasi selengkapnya tentang build multi-tahap, lihat Build multi-tahap dalam dokumentasi. Docker

Pindai gambar Anda untuk kerentanan

Mirip dengan rekan mesin virtual mereka, gambar kontainer dapat berisi binari dan pustaka aplikasi dengan kerentanan atau mengembangkan kerentanan dari waktu ke waktu. Cara terbaik untuk melindungi terhadap eksploitasi adalah dengan memindai gambar Anda secara teratur dengan pemindai gambar.

Amazon ECR menyediakan dua versi pemindaian dasar yang menggunakan database Common Vulnerabilities and Exposures ()CVEs:

- AWS pemindaian dasar AWS asli — Menggunakan teknologi asli, yang sekarang GA dan direkomendasikan. Pemindaian dasar yang ditingkatkan ini dirancang untuk memberi pelanggan hasil pemindaian yang lebih baik dan deteksi kerentanan di seluruh rangkaian luas sistem operasi populer. Hal ini memungkinkan pelanggan untuk lebih memperkuat keamanan gambar kontainer mereka. Semua pendaftar pelanggan baru dipilih ke dalam versi yang ditingkatkan ini secara default.
- Clair basic scanning — Versi pemindaian dasar sebelumnya yang menggunakan proyek Clair open-source dan tidak digunakan lagi. Untuk informasi lebih lanjut tentang Clair, lihat Clair di. GitHub

Pemindaian dasar AWS asli dan Clair didukung di semua wilayah yang terdaftar di AWS Layanan menurut Wilayah, kecuali yang ditambahkan setelah September 2024. Karena dukungan Clair tidak digunakan lagi, Clair tidak akan didukung di wilayah baru karena ditambahkan dan tidak akan lagi didukung di semua wilayah mulai 1 Oktober 2025.

Amazon ECR menggunakan tingkat keparahan untuk CVE dari sumber distribusi hulu jika tersedia. Jika tidak, skor Common Vulnerability Scoring System (CVSS) digunakan. Skor CVSS dapat digunakan untuk mengetahui tingkat kepelikan kelemahan NVD. Untuk informasi selengkapnya, lihat Tingkat kepelikan kelemahan NVD.

Anda juga dapat melihat hasil pemindaian dari dalam konsol Amazon ECR atau dengan memanggil DescribeImageScanFindingsAPI. Gambar dengan `CRITICAL` kerentanan `HIGH` atau harus dihapus

atau dibangun kembali. Jika gambar yang telah digunakan mengembangkan kerentanan, itu harus diganti sesegera mungkin.

Docker Desktop Edge versi 2.3.6.0 atau yang lebih baru dapat memindai gambar lokal. Pemindaian ini didukung oleh Snyk, layanan keamanan aplikasi. Ketika kerentanan ditemukan, Snyk mengidentifikasi lapisan dan dependensi dengan kerentanan di Dockerfile. Ini juga merekomendasikan alternatif yang aman seperti menggunakan gambar dasar yang lebih ramping dengan kerentanan yang lebih sedikit atau meningkatkan paket tertentu ke versi yang lebih baru. Dengan menggunakan pemindaian Docker, pengembang dapat menyelesaikan masalah keamanan potensial sebelum mendorong gambar mereka ke registri.

- Mengotomatiskan kepatuhan gambar menggunakan Amazon ECR dan AWS Security Hub menjelaskan cara memunculkan informasi kerentanan dari Amazon ECR AWS Security Hub dan mengotomatiskan remediasi dengan memblokir akses ke gambar yang rentan.

Hapus izin khusus dari gambar Anda

Hak akses menandai `setuid` dan `setgid` memungkinkan menjalankan executable dengan izin pemilik atau grup yang dapat dieksekusi. Hapus semua binari dengan hak akses ini dari gambar Anda karena binari ini dapat digunakan untuk meningkatkan hak istimewa. Pertimbangkan untuk menghapus semua shell dan utilitas seperti `nc` dan `curl` yang dapat digunakan untuk tujuan jahat. Anda dapat menemukan file dengan `setuid` dan hak `setgid` akses dengan menggunakan perintah berikut.

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

Untuk menghapus izin khusus ini dari file-file ini, tambahkan arahan berikut ke gambar kontainer Anda.

```
RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true
```

Buat satu set gambar yang dikuratori

Daripada mengizinkan pengembang untuk membuat gambar mereka sendiri, buat satu set gambar yang diperiksa untuk tumpukan aplikasi yang berbeda di organisasi Anda. Dengan demikian,

pengembang dapat mengabaikan belajar bagaimana menulis Dockerfiles dan berkonsentrasi pada penulisan kode. Saat perubahan digabungkan ke dalam basis kode Anda, pipeline CI/CD dapat secara otomatis mengkompilasi aset dan kemudian menyimpannya dalam repositori artefak. Dan, terakhir, salin artefak ke gambar yang sesuai sebelum mendorongnya ke registri Docker seperti Amazon ECR. Setidaknya Anda harus membuat satu set gambar dasar yang dapat dibuat oleh pengembang Dockerfiles mereka sendiri. Anda harus menghindari menarik gambar dari Docker Hub. Anda tidak selalu tahu apa yang ada dalam gambar dan sekitar seperlima dari 1000 gambar teratas memiliki kerentanan. Daftar gambar-gambar tersebut dan kerentanannya dapat ditemukan di https://vulnerablecontainers.org/.

Pindai paket aplikasi dan pustaka untuk mencari kerentanan

Penggunaan pustaka open source sekarang umum. Seperti halnya sistem operasi dan paket OS, pustaka ini dapat memiliki kerentanan. Sebagai bagian dari siklus hidup pengembangan, pustaka ini harus dipindai dan diperbarui saat kerentanan kritis ditemukan.

Docker Desktop melakukan pemindaian lokal menggunakan Snyk. Ini juga dapat digunakan untuk menemukan kerentanan dan potensi masalah lisensi di perpustakaan open source. Ini dapat diintegrasikan langsung ke dalam alur kerja pengembang yang memberi Anda kemampuan untuk mengurangi risiko yang ditimbulkan oleh pustaka open source. Untuk informasi selengkapnya, lihat topik berikut:

- Open Source Application Security Tools mencakup daftar alat untuk mendeteksi kerentanan dalam aplikasi.

Lakukan analisis kode statis

Anda harus melakukan analisis kode statis sebelum membuat gambar kontainer. Ini dilakukan terhadap kode sumber Anda dan digunakan untuk mengidentifikasi kesalahan pengkodean dan kode yang dapat dieksploitasi oleh aktor jahat, seperti suntikan kesalahan. Anda dapat menggunakan Amazon Inspector. Untuk informasi selengkapnya, lihat Memindai gambar penampung Amazon ECR dengan Amazon Inspector di Panduan Pengguna Amazon Inspector.

Jalankan kontainer sebagai pengguna non-root

Anda harus menjalankan container sebagai pengguna non-root. Secara default, container berjalan sebagai `root` pengguna kecuali `USER` direktif disertakan dalam Dockerfile Anda. Kemampuan default Linux yang ditetapkan oleh Docker membatasi tindakan yang dapat dijalankan sebagai`root`, tetapi

hanya sedikit. Misalnya, wadah yang berjalan seperti `root` masih tidak diizinkan untuk mengakses perangkat.

Sebagai bagian dari CI/CD pipeline Anda, Anda harus lint Dockerfiles untuk mencari `USER` arahan dan gagal membangun jika tidak ada. Untuk informasi selengkapnya, lihat topik berikut:

- DockerFile-Lint adalah alat sumber terbuka RedHat yang dapat digunakan untuk memeriksa apakah file tersebut sesuai dengan praktik terbaik.
- Hadolint adalah alat lain untuk membangun gambar Docker yang sesuai dengan praktik terbaik.

Gunakan sistem file root read-only

Anda harus menggunakan sistem file root read-only. Sistem file root container dapat ditulis secara default. Saat Anda mengonfigurasi wadah dengan sistem file root `RO` (hanya-baca), itu memaksa Anda untuk secara eksplisit menentukan di mana data dapat disimpan. Ini mengurangi permukaan serangan Anda karena sistem file container tidak dapat ditulis kecuali izin diberikan secara khusus.

> **Note**
>
> Memiliki sistem file root read-only dapat menyebabkan masalah dengan paket OS tertentu yang diharapkan dapat menulis ke sistem file. Jika Anda berencana untuk menggunakan sistem file root read-only, uji secara menyeluruh sebelumnya.

Konfigurasikan tugas dengan batas CPU dan Memori (Amazon EC2)

Anda harus mengkonfigurasi tugas dengan CPU dan batas memori untuk meminimalkan risiko berikut. Batas sumber daya tugas menetapkan batas atas untuk jumlah CPU dan memori yang dapat dicadangkan oleh semua wadah dalam tugas. Jika tidak ada batasan yang ditetapkan, tugas memiliki akses ke CPU dan memori host. Hal ini dapat menyebabkan masalah di mana tugas yang diterapkan pada host bersama dapat membuat tugas sumber daya sistem lainnya kelaparan.

> **Note**
>
> Amazon ECS pada AWS Fargate tugas mengharuskan Anda menentukan batas CPU dan memori karena menggunakan nilai ini untuk tujuan penagihan. Satu tugas memonopoli semua sumber daya sistem tidak menjadi masalah bagi Amazon ECS Fargate karena setiap tugas dijalankan pada instance khusus sendiri. Jika Anda tidak menentukan batas memori,

Amazon ECS mengalokasikan minimal 4MB untuk setiap kontainer. Demikian pula, jika tidak ada batas CPU yang ditetapkan untuk tugas tersebut, agen penampung Amazon ECS menetapkannya minimal 2. CPUs

Gunakan tag yang tidak dapat diubah dengan Amazon ECR

Dengan Amazon ECR, Anda dapat dan harus menggunakan konfigurasi gambar dengan tag yang tidak dapat diubah. Ini mencegah mendorong versi gambar yang diubah atau diperbarui ke repositori gambar Anda dengan tag yang identik. Ini melindungi terhadap penyerang yang mendorong versi gambar yang dikompromikan di atas gambar Anda dengan tag yang sama. Dengan menggunakan tag yang tidak dapat diubah, Anda secara efektif memaksa diri Anda untuk mendorong gambar baru dengan tag berbeda untuk setiap perubahan.

Hindari menjalankan kontainer sebagai hak istimewa (Amazon EC2)

Anda harus menghindari menjalankan kontainer sebagai hak istimewa. Untuk latar belakang, kontainer dijalankan seperti `privileged` yang dijalankan dengan hak istimewa yang diperluas pada host. Ini berarti wadah mewarisi semua kemampuan Linux yang ditetapkan `root` pada host. Penggunaannya harus sangat dibatasi atau dilarang. Kami menyarankan untuk menyetel variabel lingkungan agen penampung Amazon ECS `ECS_DISABLE_PRIVILEGED` `true` untuk mencegah kontainer berjalan seperti `privileged` pada host tertentu jika `privileged` tidak diperlukan. Atau Anda dapat menggunakan AWS Lambda untuk memindai definisi tugas Anda untuk penggunaan `privileged` parameter.

> **Note**
>
> Menjalankan wadah seperti yang `privileged` tidak didukung di Amazon ECS aktif. AWS Fargate

Hapus kemampuan Linux yang tidak perlu dari wadah

Berikut ini adalah daftar kemampuan Linux default yang ditetapkan untuk kontainer Docker. Untuk informasi selengkapnya tentang setiap kemampuan, lihat Ikhtisar Kemampuan Linux.

```
CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL,
```

```
CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_NET_BIND_SERVICE,
CAP_NET_RAW, CAP_SYS_CHROOT, CAP_MKNOD, CAP_AUDIT_WRITE,
CAP_SETFCAP
```

Jika sebuah container tidak memerlukan semua kemampuan kernel Docker yang tercantum di atas, pertimbangkan untuk menjatuhkannya dari container. Untuk informasi selengkapnya tentang setiap kemampuan kernel Docker, lihat KernelCapabilities. Anda dapat mengetahui kemampuan mana yang digunakan dengan melakukan hal berikut:

- Instal paket OS libcap-ng dan jalankan `pscap` utilitas untuk membuat daftar kemampuan yang digunakan setiap proses.
- Anda juga dapat menggunakan capsh untuk menguraikan kemampuan mana yang digunakan suatu proses.

Gunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi gambar yang didorong ke Amazon ECR

Anda harus menggunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi gambar yang didorong ke Amazon ECR. Gambar yang didorong ke Amazon ECR secara otomatis dienkripsi saat istirahat dengan kunci terkelola AWS Key Management Service (AWS KMS). Jika Anda lebih suka menggunakan kunci Anda sendiri, Amazon ECR sekarang mendukung AWS KMS enkripsi dengan kunci terkelola pelanggan (CMK). Sebelum mengaktifkan enkripsi sisi server dengan CMK, tinjau Pertimbangan yang tercantum dalam dokumentasi tentang enkripsi saat istirahat.

Tutorial untuk Amazon ECS

Tutorial berikut menunjukkan cara melakukan tugas-tugas umum saat menggunakan Amazon ECS.

Anda dapat menggunakan salah satu tutorial berikut untuk mempelajari lebih lanjut tentang memulai dengan Amazon ECS.

Gambaran umum tutorial	Pelajari selengkapnya
Mulailah dengan Amazon ECS di Fargate.	Pelajari cara membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate
Memulai dengan wadah Windows di Fargate.	Pelajari cara membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate
Memulai dengan wadah Windows untuk jenis EC2 peluncuran.	Pelajari cara membuat tugas Amazon ECS Windows untuk jenis EC2 peluncuran

Anda dapat menggunakan salah satu tutorial berikut untuk menyebarkan tugas di Amazon ECS menggunakan AWS CLI

Gambaran umum tutorial	Pelajari selengkapnya
Buat tugas Linux untuk jenis peluncuran Fargate.	Membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate dengan AWS CLI
Buat tugas Windows untuk jenis peluncuran Fargate.	Membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate dengan AWS CLI

Gambaran umum tutorial	Pelajari selengkapnya	
Buat tugas Linux untuk jenis EC2 peluncuran.	Membuat tugas Amazon ECS untuk jenis EC2 peluncuran dengan AWS CLI	

Anda dapat menggunakan salah satu tutorial berikut untuk mempelajari lebih lanjut tentang pemantauan dan pencatatan.

Gambaran umum tutorial	Pelajari selengkapnya	
Siapkan fungsi Lambda sederhana yang mendengar kan peristiwa tugas dan menuliskannya ke aliran CloudWatch log Log.	Mengonfigurasi Amazon ECS untuk mendengarkan acara Acara CloudWatch	
Konfigurasikan aturan EventBridge peristiwa Amazon yang hanya menangkap peristiwa tugas di mana tugas telah berhenti berjalan karena salah satu wadah pentingnya telah dihentikan.	Mengirim peringatan Amazon Simple Notification Service untuk tugas Amazon ECS menghentikan peristiwa	
Menggabungkan pesan log yang awalnya milik satu konteks tetapi dibagi menjadi beberapa catatan atau baris log.	Menggabungkan pesan log Amazon ECS multiline atau stack-trace	
Terapkan container Fluent Bit pada instance Windows mereka yang berjalan di Amazon ECS untuk mengalirk an log yang dihasilkan oleh tugas Windows ke Amazon	Menyebarkan Fluent Bit di Amazon ECS Windows container	

Gambaran umum tutorial	Pelajari selengkapnya	
CloudWatch untuk pencatatan terpusat.		

Anda dapat menggunakan salah satu tutorial berikut untuk mempelajari lebih lanjut tentang cara menggunakan otentikasi Direktori Aktif dengan Akun Layanan Terkelola grup di Amazon ECS.

Gambaran umum tutorial	Pelajari selengkapnya	
Gunakan Akun Layanan Terkelola grup dengan wadah Linux aktif EC2.	Menggunakan gMSA untuk EC2 Linux wadah di Amazon ECS	
Gunakan Akun Layanan Terkelola grup dengan wadah Windows aktif EC2.	Pelajari cara menggunakan g MSAs untuk wadah EC2 Windows untuk Amazon ECS	
Gunakan Akun Layanan Terkelola grup dengan wadah Linux di Fargate.	Menggunakan gMSA untuk Linux wadah di Fargate	
Buat tugas yang menjalankan wadah Windows yang memiliki kredensyal untuk mengakses Active Directory dengan akun layanan terkelola grup tanpa domain.	Menggunakan Amazon ECS Windows container dengan domainless gMSA menggunakan AWS CLI	

Membuat tugas Amazon ECS Linux untuk jenis peluncuran Fargate dengan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas Linux, dan melakukan skenario umum lainnya di Amazon ECS dengan. AWS CLI Gunakan versi terbaru dari AWS CLI. Untuk informasi selengkapnya tentang cara memutakhirkan ke versi terbaru, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Topik

- Prasyarat
- Langkah 1: Buat Klaster
- Langkah 2: Daftarkan Definisi Tugas Linux
- Langkah 3: Cantumkan Ketentuan tugas
- Langkah 4: Buat Layanan
- Langkah 5: Cantumkan Layanan
- Langkah 6: Jelaskan Layanan yang Berjalan
- Langkah 7: Uji
- Langkah 8: Bersihkan

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan.

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi lebih lanjut tentang menginstal atau memutakhirkan AWS CLI, Menginstal atau memperbarui ke versi terbaru. AWS CLI
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC dan grup keamanan yang dibuat untuk digunakan. Tutorial ini menggunakan gambar kontainer yang dihosting di Amazon ECR Public sehingga tugas Anda harus memiliki akses internet. Untuk memberikan tugas Anda rute ke internet, gunakan salah satu pilihan berikut ini.
  - Gunakan subnet privat dengan gateway NAT yang memiliki alamat IP elastis.
  - Gunakan subnet publik dan tetapkan alamat IP publik untuk tugas tersebut.

Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

Untuk informasi tentang grup dan aturan keamanan, lihat, Grup keamanan default untuk aturan Anda VPCs dan Contoh di Panduan Pengguna Amazon Virtual Private Cloud.

- Jika Anda mengikuti tutorial ini menggunakan subnet pribadi, Anda dapat menggunakan Amazon ECS Exec untuk berinteraksi langsung dengan penampung Anda dan menguji penerapan. Anda akan perlu untuk membuat peran tugas IAM untuk menggunakan ECS Exec. Untuk informasi selengkapnya tentang peran IAM tugas dan prasyarat lainnya, lihat Memantau kontainer Amazon ECS dengan Amazon ECS Exec.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat EC2 instance mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat Klaster

Secara default, akun Anda menerima klaster `default`.

> **Note**
>
> Manfaat dari penggunaan klaster `default` yang disediakan untuk Anda adalah bahwa Anda tidak perlu menentukan properti pada pilihan `--cluster` *`cluster_name`* di perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name fargate-cluster
```

Output:

```
{
    "cluster": {
        "status": "ACTIVE",
        "defaultCapacityProviderStrategy": [],
        "statistics": [],
```

```
        "capacityProviders": [],
        "tags": [],
        "clusterName": "fargate-cluster",
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

Langkah 2: Daftarkan Definisi Tugas Linux

Sebelum Anda dapat menjalankan tugas di klaster ECS, Anda harus mendaftarkan ketentuan tugas. Ketentuan tugas merupakan daftar kontainer yang dikelompokkan bersama-sama. Contoh berikut adalah ketentuan tugas sederhana yang menciptakan sebuah aplikasi web PHP dengan menggunakan citra kontainer httpd yang di-hosting di Docker Hub. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS. Untuk tutorial ini, hanya `taskRoleArn` diperlukan jika Anda menerapkan tugas di subnet pribadi dan ingin menguji penerapan. Ganti `taskRoleArn` dengan peran tugas IAM yang Anda buat untuk menggunakan ECS Exec seperti yang disebutkan dalam. Prasyarat

```
{
        "family": "sample-fargate",
        "networkMode": "awsvpc",
        "taskRoleArn": "arn:aws:iam::aws_account_id:role/execCommandRole",
        "containerDefinitions": [
            {
                "name": "fargate-app",
                "image": "public.ecr.aws/docker/library/httpd:latest",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
```

```
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

Simpan definisi tugas JSON sebagai file dan berikan dengan `--cli-input-json file://path_to_file.json` opsi.

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

Parameter perintah register-task-definition mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 3: Cantumkan Ketentuan tugas

Anda dapat mencantumkan daftar ketentuan tugas untuk akun Anda kapan saja dengan perintah list-task-definitions. Output dari perintah ini menunjukkan nilai `family` dan `revision` yang dapat Anda gunakan bersama saat memanggil run-task atau start-task.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:1"
    ]
}
```

Langkah 4: Buat Layanan

Setelah Anda telah mendaftarkan tugas untuk akun Anda, Anda dapat membuat layanan untuk tugas yang telah terdaftar di klaster Anda. Pada contoh ini, Anda membuat layanan dengan satu instans dari ketentuan tugas `sample-fargate:1` yang berjalan di klaster Anda. Tugas tersebut membutuhkan rute menuju internet, maka terdapat dua cara untuk Anda agar bisa mencapainya. Salah satu caranya adalah dengan menggunakan subnet privat yang dikonfigurasikan dengan gateway NAT dengan alamat IP elastis di subnet publik. Cara lainnya adalah dengan menggunakan subnet publik dan menetapkan alamat IP publik untuk tugas Anda. Kami menyediakan kedua contoh di bawah ini.

Contoh menggunakan subnet privat. `enable-execute-command` Opsi ini diperlukan untuk menggunakan Amazon ECS Exec.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}" --enable-execute-command
```

Contoh menggunakan subnet publik.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234],assignPublicIp=ENABLED}"
```

Perintah create-service mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 5: Cantumkan Layanan

Cantumkan layanan untuk klaster Anda. Anda seharusnya dapat melihat layanan yang Anda buat di bagian sebelumnya. Anda dapat mengambil nama layanan atau ARN penuh yang telah dikembalikan dari perintah ini serta menggunakannya untuk menjelaskan layanan nantinya.

```
aws ecs list-services --cluster fargate-cluster
```

Output:

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-cluster/fargate-service"
    ]
}
```

Langkah 6: Jelaskan Layanan yang Berjalan

Jelaskan layanan yang menggunakan nama layanan yang telah diambil sebelumnya untuk mendapatkan informasi lebih lanjut tentang tugas.

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

Jika berhasil, tugas akan mengembalikan deskripsi dari layanan yang gagal juga layanan. Misalnya, di `services` bagian ini, Anda akan menemukan informasi tentang penerapan, seperti status tugas sebagai berjalan atau tertunda. Anda juga dapat menemukan informasi tentang ketentuan tugas, konfigurasi jaringan dan peristiwa stempel waktu. Pada bagian kegagalan, Anda akan menemukan informasi tentang kegagalan, jika ada kegagalan tersebut, yang terkait dengan panggilan tersebut. Untuk pemecahan masalah, lihat Pesan Peristiwa Layanan. Untuk informasi selengkapnya tentang deskripsi layanan, lihat Jelaskan layanan.

```
{
    "services": [
        {
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "ENABLED"
                }
            },
            "launchType": "FARGATE",
```

```
            "enableECSManagedTags": false,
            "loadBalancers": [],
            "deploymentController": {
                "type": "ECS"
            },
            "desiredCount": 1,
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            },
            "createdAt": 1692283199.771,
            "schedulingStrategy": "REPLICA",
            "placementConstraints": [],
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "ENABLED"
                        }
                    },
                    "pendingCount": 0,
                    "launchType": "FARGATE",
                    "createdAt": 1692283199.771,
                    "desiredCount": 1,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate:1",
                    "updatedAt": 1692283199.771,
                    "platformVersion": "1.4.0",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "serviceName": "fargate-service",
            "events": [
                {
```

```
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "status": "ACTIVE",
            "serviceRegistries": [],
            "pendingCount": 0,
            "createdBy": "arn:aws:iam::aws_account_id:user/user_name",
            "platformVersion": "LATEST",
            "placementStrategy": [],
            "propagateTags": "NONE",
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1"
        }
    ],
    "failures": []
}
```

Langkah 7: Uji

Tugas pengujian dikerahkan menggunakan subnet publik

Jelaskan tugas dalam layanan sehingga Anda bisa mendapatkan Elastic Network Interface (ENI) untuk tugas tersebut.

Pertama, dapatkan tugas ARN.

```
aws ecs list-tasks --cluster fargate-cluster --service fargate-service
```

Outputnya berisi tugas ARN.

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE
    ]
}
```

Jelaskan tugas dan temukan ID ENI. Gunakan tugas ARN untuk parameter. `tasks`

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/service/EXAMPLE
```

Informasi lampiran tercantum dalam output.

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                    ]
                }
…
}
```

Jelaskan ENI untuk mendapatkan alamat IP publik.

```
aws ec2 describe-network-interfaces --network-interface-id  eni-0fa40520aeEXAMPLE
```

Alamat IP publik ada di output.

```
{
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-229-42-222.compute-1.amazonaws.com",
                "PublicIp": "198.51.100.2"
            },
…
}
```

Masukkan alamat IP publik di browser web Anda dan Anda akan melihat halaman web yang menampilkan contoh aplikasi Amazon ECS.

Tugas pengujian yang digunakan menggunakan subnet pribadi

Jelaskan tugas dan temukan `managedAgents` untuk memverifikasi bahwa `ExecuteCommandAgent` sedang berjalan. Perhatikan `privateIPv4Address` untuk digunakan nanti.

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/fargate-service/EXAMPLE
```

Informasi agen terkelola tercantum dalam output.

```
{
     "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
```

```
                        "name": "networkInterfaceId",
                        "value": "eni-0fa40520aeEXAMPLE"
                    },
                    {
                        "name": "privateIPv4Address",
                        "value": "10.0.143.156"
                    }
                ]
            }
        ],
    ...
    "containers": [
        {
        ...
       "managedAgents": [
                    {
                        "lastStartedAt": "2023-08-01T16:10:13.002000+00:00",
                        "name": "ExecuteCommandAgent",
                        "lastStatus": "RUNNING"
                    }
            ],
        ...
   }
```

Setelah memverifikasi bahwa `ExecuteCommandAgent` sedang berjalan, Anda dapat menjalankan perintah berikut untuk menjalankan shell interaktif pada wadah dalam tugas.

```
aws ecs execute-command --cluster fargate-cluster \
    --task  arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE  \
    --container  fargate-app \
    --interactive \
    --command "/bin/sh"
```

Setelah shell interaktif berjalan, jalankan perintah berikut untuk menginstal cURL.

```
apt update
```

```
apt install curl
```

Setelah menginstal cURL, jalankan perintah berikut menggunakan alamat IP pribadi yang Anda peroleh sebelumnya.

```
curl 10.0.143.156
```

Anda akan melihat HTML yang setara dengan halaman web aplikasi sampel Amazon ECS.

```
<html>
    <head>
     <title>Amazon ECS Sample App</title>
     <style>body {margin-top: 40px; background-color: #333;} </style>
    </head>
      <body>
      <div style=color:white;text-align:center>
      <h1>Amazon ECS Sample App</h1>
      <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>
      </div>
      </body>
</html>
```

Langkah 8: Bersihkan

Setelah Anda selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari biaya sumber daya yang tidak terpakai.

Hapus layanan.

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

Hapus klaster.

```
aws ecs delete-cluster --cluster fargate-cluster
```

Membuat tugas Amazon ECS Windows untuk jenis peluncuran Fargate dengan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas Windows, dan melakukan skenario umum lainnya di Amazon ECS dengan file. AWS CLI Pastikan bahwa Anda menggunakan versi terbaru AWS CLI. Untuk informasi selengkapnya

tentang cara memutakhirkan ke versi terbaru, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Topik

- Prasyarat
- Langkah 1: Buat Klaster
- Langkah 2: Daftarkan Definisi Tugas Windows
- Langkah 3: Daftar definisi tugas
- Langkah 4: Buat layanan
- Langkah 5: Daftar layanan
- Langkah 6: Jelaskan Layanan yang Berjalan
- Langkah 7: Bersihkan

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan.

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal atau memperbarui ke versi terbaru. AWS CLI
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki VPC dan grup keamanan yang telah dibuat untuk digunakan. Tutorial ini menggunakan citra kontainer yang di-host pada Docker Hub sehingga tugas Anda harus memiliki akses kepada internet. Untuk memberikan tugas Anda rute ke internet, gunakan salah satu pilihan berikut ini.
  - Gunakan subnet privat dengan gateway NAT yang memiliki alamat IP elastis.

- Gunakan subnet publik dan tetapkan alamat IP publik untuk tugas tersebut.

  Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.

  Untuk informasi tentang grup dan aturan keamanan, lihat, Grup keamanan default untuk aturan Anda VPCs dan Contoh di Panduan Pengguna Amazon Virtual Private Cloud.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat EC2 instance mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat Klaster

Secara default, akun Anda menerima klaster `default`.

> **Note**
>
> Manfaat dari penggunaan klaster `default` yang disediakan untuk Anda adalah bahwa Anda tidak perlu menentukan properti pada pilihan `--cluster` *`cluster_name`* di perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name fargate-cluster
```

Output:

```
{
    "cluster": {
        "status": "ACTIVE",
        "statistics": [],
        "clusterName": "fargate-cluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
```

```
    }
}
```

Langkah 2: Daftarkan Definisi Tugas Windows

Sebelum Anda dapat menjalankan tugas Windows di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah definisi tugas sederhana yang membuat aplikasi web. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
```

```
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

Contoh di atas JSON dapat diteruskan ke AWS CLI dalam dua cara: Anda dapat menyimpan definisi tugas JSON sebagai file dan meneruskannya dengan opsi. `--cli-input-json file://path_to_file.json`

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

Parameter perintah register-task-definition mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 3: Daftar definisi tugas

Anda dapat mencantumkan daftar ketentuan tugas untuk akun Anda kapan saja dengan perintah list-task-definitions. Output dari perintah ini menunjukkan nilai `family` dan `revision` yang dapat Anda gunakan bersama saat memanggil run-task atau start-task.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate-windows:1"
    ]
}
```

Langkah 4: Buat layanan

Setelah Anda telah mendaftarkan tugas untuk akun Anda, Anda dapat membuat layanan untuk tugas yang telah terdaftar di klaster Anda. Pada contoh ini, Anda membuat layanan dengan satu instans dari ketentuan tugas `sample-fargate:1` yang berjalan di klaster Anda. Tugas tersebut membutuhkan rute menuju internet, maka terdapat dua cara untuk Anda agar bisa mencapainya. Salah satu caranya adalah dengan menggunakan subnet privat yang dikonfigurasikan dengan gateway NAT dengan alamat IP elastis di subnet publik. Cara lainnya adalah dengan menggunakan

subnet publik dan menetapkan alamat IP publik untuk tugas Anda. Kami menyediakan kedua contoh di bawah ini.

Contoh menggunakan subnet privat.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234]}"
```

Contoh menggunakan subnet publik.

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

Perintah create-service mengembalikan penjelasan pada ketentuan tugas setelah menyelesaikan pendaftarannya.

Langkah 5: Daftar layanan

Cantumkan layanan untuk klaster Anda. Anda seharusnya dapat melihat layanan yang Anda buat di bagian sebelumnya. Anda dapat mengambil nama layanan atau ARN penuh yang telah dikembalikan dari perintah ini serta menggunakannya untuk menjelaskan layanan nantinya.

```
aws ecs list-services --cluster fargate-cluster
```

Output:

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-service"
    ]
}
```

Langkah 6: Jelaskan Layanan yang Berjalan

Jelaskan layanan yang menggunakan nama layanan yang telah diambil sebelumnya untuk mendapatkan informasi lebih lanjut tentang tugas.

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

Jika berhasil, tugas akan mengembalikan deskripsi dari layanan yang gagal juga layanan. Sebagai contoh, pada bagian layanan, Anda akan menemukan informasi tentang deployment, seperti menjelaskan status pada tugas berjalan atau tertundanya tugas tersebut. Anda juga dapat menemukan informasi tentang ketentuan tugas, konfigurasi jaringan dan peristiwa stempel waktu. Pada bagian kegagalan, Anda akan menemukan informasi tentang kegagalan, jika ada kegagalan tersebut, yang terkait dengan panggilan tersebut. Untuk pemecahan masalah, lihat Pesan Peristiwa Layanan. Untuk informasi selengkapnya tentang deskripsi layanan, lihat Jelaskan layanan.

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate-windows:1",
            "pendingCount": 2,
            "launchType": "FARGATE",
            "loadBalancers": [],
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "placementConstraints": [],
            "createdAt": 1510811361.128,
            "desiredCount": 2,
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "DISABLED"
                }
            },
            "platformVersion": "LATEST",
            "serviceName": "fargate-service",
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
```

```
            },
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "DISABLED"
                        }
                    },
                    "pendingCount": 2,
                    "launchType": "FARGATE",
                    "createdAt": 1510811361.128,
                    "desiredCount": 2,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate-windows:1",
                    "updatedAt": 1510811361.128,
                    "platformVersion": "0.0.1",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
```

```
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "placementStrategy": []
        }
    ],
    "failures": []
}
```

Langkah 7: Bersihkan

Setelah Anda selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari biaya sumber daya yang tidak terpakai.

Hapus layanan.

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

Hapus klaster.

```
aws ecs delete-cluster --cluster fargate-cluster
```

Membuat tugas Amazon ECS untuk jenis EC2 peluncuran dengan AWS CLI

Langkah-langkah berikut membantu Anda menyiapkan klaster, mendaftarkan definisi tugas, menjalankan tugas, dan melakukan skenario umum lainnya di Amazon ECS dengan. AWS CLI Gunakan versi terbaru dari AWS CLI. Untuk informasi selengkapnya tentang cara memutakhirkan ke versi terbaru, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Topik

- Prasyarat
- Membuat klaster
- Luncurkan instance kontainer dengan Amazon ECS AMI
- Daftar contoh kontainer
- Jelaskan contoh kontainer Anda
- Daftarkan definisi tugas
- Daftar definisi tugas
- Membuat layanan
- Daftar layanan
- Jelaskan layanannya
- Jelaskan tugas yang sedang berjalan
- Uji server web
- Pembersihan sumber daya

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal atau memperbarui ke versi terbaru. AWS CLI
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Anda memiliki peran IAM instance container yang dibuat untuk digunakan. Untuk informasi lebih lanjut, lihat. Peran IAM instance wadah Amazon ECS
- Anda memiliki VPC yang dibuat untuk digunakan. Untuk informasi selengkapnya, lihat the section called “Buat virtual private cloud”.
- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat EC2 instance mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Membuat klaster

Secara default, akun Anda menerima klaster `default` ketika Anda meluncurkan instans kontainer pertama Anda.

> **Note**
>
> Manfaat menggunakan klaster `default` yang disediakan untuk Anda adalah Anda tidak perlu menentukan opsi `--cluster` *`cluster_name`* dalam perintah berikutnya. Jika Anda membuat klaster non-default sendiri, Anda harus menentukan `--cluster` *`cluster_name`* untuk setiap perintah yang ingin Anda gunakan dengan klaster itu.

Buat klaster Anda sendiri dengan nama yang unik menggunakan perintah berikut:

```
aws ecs create-cluster --cluster-name MyCluster
```

Output:

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

Luncurkan instance kontainer dengan Amazon ECS AMI

Instance kontainer adalah EC2 instance yang menjalankan agen penampung Amazon ECS dan telah terdaftar ke dalam klaster. Di bagian ini, Anda akan meluncurkan EC2 instance menggunakan AMI yang dioptimalkan ECS.

Untuk meluncurkan instance kontainer dengan AWS CLI

1. Ambil ID AMI Amazon Linux 2 terbaru yang dioptimalkan ECS untuk Wilayah AWS Anda menggunakan perintah berikut. Perintah ini menggunakan AWS Systems Manager Parameter Store untuk mendapatkan ID AMI terbaru yang dioptimalkan ECS. AMI mencakup agen kontainer Amazon ECS dan runtime Docker yang sudah diinstal sebelumnya.

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --query 'Parameters[0].Value' --output text | jq -r '.image_id'
```

Output:

```
ami-abcd1234
```

2. Buat grup keamanan yang memungkinkan akses SSH untuk mengelola instance container Anda dan akses HTTP untuk server web.

```
aws ec2 create-security-group --group-name ecs-tutorial-sg --description "ECS
 tutorial security group"
```

Output:

```
{
    "GroupId": "sg-abcd1234"
}
```

3. Tambahkan aturan masuk ke grup keamanan dengan menjalankan perintah berikut.

```
aws ec2 authorize-security-group-ingress --group-id sg-abcd1234 --protocol tcp --
port 80 --cidr 0.0.0.0/0
```

Output:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-efgh5678",
            "GroupId": "sg-abcd1234",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
```

```
}
```

Grup keamanan sekarang memungkinkan akses SSH dari rentang IP yang ditentukan dan akses HTTP dari mana saja. Dalam lingkungan produksi, Anda harus membatasi akses SSH ke alamat IP spesifik Anda dan mempertimbangkan untuk membatasi akses HTTP sesuai kebutuhan.

4. Buat EC2 key pair untuk akses SSH ke instance container Anda.

```
aws ec2 create-key-pair --key-name ecs-tutorial-key --query 'KeyMaterial' --output
 text > ecs-tutorial-key.pem
chmod 400 ecs-tutorial-key.pem
```

Kunci pribadi disimpan ke mesin lokal Anda dengan izin yang sesuai untuk akses SSH.

5. Luncurkan EC2 instance menggunakan AMI yang dioptimalkan ECS dan konfigurasikan untuk bergabung dengan cluster Anda.

```
aws ec2 run-instances --image-id ami-abcd1234 --instance-type t3.micro --key-
name ecs-tutorial-key --security-group-ids sg-abcd1234 --iam-instance-profile
 Name=ecsInstanceRole --user-data '#!/bin/bash
echo ECS_CLUSTER=MyCluster >> /etc/ecs/ecs.config'
{
    "Instances": [
        {
            "InstanceId": "i-abcd1234",
            "ImageId": "ami-abcd1234",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "PrivateDnsName": "",
            "PublicDnsName": "",
            "StateReason": {
                "Code": "pending",
                "Message": "pending"
            },
            "InstanceType": "t3.micro",
            "KeyName": "ecs-tutorial-key",
            "LaunchTime": "2025-01-13T10:30:00.000Z"
        }
    ]
}
```

Skrip data pengguna mengonfigurasi agen Amazon ECS untuk mendaftarkan instans ke instans Anda. `MyCluster` Instance menggunakan peran `ecsInstanceRole` IAM, yang memberikan izin yang diperlukan untuk agen.

Daftar contoh kontainer

Dalam beberapa menit setelah meluncurkan instans penampung Anda, agen Amazon ECS mendaftarkan instans tersebut dengan klaster default Anda. Anda dapat membuat daftar instans kontainer dalam klaster dengan menjalankan perintah berikut:

```
aws ecs list-container-instances --cluster default
```

Output:

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

Jelaskan contoh kontainer Anda

Setelah Anda memiliki ARN atau ID instans kontainer, Anda dapat menggunakan perintah describe-container-instances untuk mendapatkan informasi berharga terkait instans, seperti sumber daya CPU dan memori yang tersisa dan terdaftar.

```
aws ecs describe-container-instances --cluster MyCluster --container-
instances container_instance_ID
```

Output:

```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
                    "integerValue": 1024,
```

```
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "ec2InstanceId": "instance_id",
            "agentConnected": true,
            "containerInstanceArn": "arn:aws:ecs:us-west-2:aws_account_id:container-
instance/container_instance_ID",
            "pendingTasksCount": 0,
            "remainingResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
```

```
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "runningTasksCount": 0,
            "attributes": [
                {
                    "name": "com.amazonaws.ecs.capability.privileged-container"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
                },
                {
```

```
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
                }
            ],
            "versionInfo": {
                "agentVersion": "1.5.0",
                "agentHash": "b197edd",
                "dockerVersion": "DockerVersion: 1.7.1"
            }
        }
    ]
}
```

Anda juga dapat menemukan ID EC2 instans Amazon yang dapat Anda gunakan untuk memantau instance di EC2 konsol Amazon atau dengan aws ec2 describe-instances --instance-id ***instance_id*** perintah.

Daftarkan definisi tugas

Sebelum Anda dapat menjalankan tugas di cluster Amazon ECS Anda, Anda harus mendaftarkan definisi tugas. Ketentuan tugas adalah daftar kontainer yang dikelompokkan bersama. Contoh berikut adalah definisi tugas sederhana yang menggunakan `nginx` gambar. Untuk informasi selengkapnya tentang parameter ketentuan tugas yang tersedia, lihat Definisi tugas Amazon ECS.

```
{
    "family": "nginx-task",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
```

```
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge"
}
```

Contoh di atas JSON dapat diteruskan ke AWS CLI dalam dua cara: Anda dapat menyimpan definisi tugas JSON sebagai file dan meneruskannya dengan opsi. `--cli-input-json file://`*`path_to_file.json`* Atau, Anda dapat menghindari tanda kutip di JSON dan meneruskan definisi wadah JSON pada baris perintah. Jika Anda memilih untuk meneruskan ketentuan kontainer pada baris perintah, maka perintah Anda juga memerlukan parameter `--family` yang digunakan untuk menjaga beberapa versi ketentuan tugas Anda yang terkait satu sama lain.

Untuk menggunakan file JSON untuk ketentuan kontainer:

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/nginx.json
```

register-task-definition mengembalikan deskripsi ketentuan tugas setelah menyelesaikan pendaftarannya.

```
{
    "taskDefinition": {
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/nginx-
task:1",
        "family": "nginx-task",
        "revision": 1,
        "status": "ACTIVE",
        "containerDefinitions": [
            {
                "name": "nginx",
                "image": "public.ecr.aws/docker/library/nginx:latest",
                "cpu": 256,
                "memory": 512,
                "essential": true,
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
```

```
                    "protocol": "tcp"
                }
            ],
            "environment": [],
            "mountPoints": [],
            "volumesFrom": []
        }
    ],
    "volumes": [],
    "networkMode": "bridge",
    "compatibilities": [
        "EC2"
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
}
```

Daftar definisi tugas

Anda dapat membuat daftar ketentuan tugas untuk akun Anda kapan pun dengan perintah list-task-definitions. Output dari perintah ini menunjukkan `revision` nilai `family` dan yang dapat Anda gunakan bersama saat memanggilcreate-service.

```
aws ecs list-task-definitions
```

Output:

```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/nginx-task:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```

Membuat layanan

Setelah mendaftarkan tugas untuk akun dan meluncurkan instans penampung yang terdaftar ke klaster, Anda dapat membuat layanan Amazon ECS yang menjalankan dan mengelola sejumlah tugas yang diinginkan secara bersamaan menggunakan definisi tugas yang Anda daftarkan. Untuk contoh ini, Anda menempatkan satu instans ketentuan tugas `nginx:1` di klaster default Anda.

```
aws ecs create-service --cluster default --service-name nginx-service --task-
definition nginx-task:1 --desired-count 1
```

Output:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:aws_account_id:service/MyCluster/nginx-
service",
        "serviceName": "nginx-service",
        "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
        "taskDefinition": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/nginx-
task:1",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "status": "ACTIVE",
        "createdAt": "2025-01-13T10:45:00.000Z"
    }
}
```

Daftar layanan

Cantumkan layanan untuk klaster Anda. Anda seharusnya dapat melihat layanan yang Anda buat di bagian sebelumnya. Anda dapat mencatat ID layanan atau ARN lengkap yang dikembalikan dari perintah ini dan menggunakannya untuk menggambarkan layanan nanti.

```
aws ecs list-services --cluster MyCluster
```

Output:

```
{
```

```
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```

Jelaskan layanannya

Jelaskan layanan menggunakan perintah berikut untuk mendapatkan informasi lebih lanjut tentang layanan ini.

```
aws ecs describe-services --cluster MyCluster --services nginx-service
```

Output:

```
{
    "services": [
        {
            "serviceArn": "arn:aws:ecs:us-east-1:aws_account_id:service/MyCluster/
nginx-service",
            "serviceName": "nginx-service",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
            "taskDefinition": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
nginx-task:1",
            "desiredCount": 1,
            "runningCount": 1,
            "pendingCount": 0,
            "launchType": "EC2",
            "status": "ACTIVE",
            "createdAt": "2025-01-13T10:45:00.000Z",
            "events": [
                {
                    "id": "abcd1234-5678-90ab-cdef-1234567890ab",
                    "createdAt": "2025-01-13T10:45:30.000Z",
                    "message": "(service nginx-service) has started 1 tasks: (task
 abcd1234-5678-90ab-cdef-1234567890ab)."
                }
            ]
        }
    ]
}
```

Jelaskan tugas yang sedang berjalan

Setelah menjelaskan layanan, jalankan perintah berikut untuk mendapatkan informasi lebih lanjut tentang tugas yang berjalan sebagai bagian dari layanan Anda.

```
aws ecs list-tasks --cluster MyCluster --service-name nginx-service
```

Output:

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/MyCluster/
abcd1234-5678-90ab-cdef-1234567890ab",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
nginx-task:1",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/MyCluster/abcd1234-5678-90ab-cdef-1234567890ab",
            "lastStatus": "RUNNING",
            "desiredStatus": "RUNNING",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-east-1:aws_account_id:container/
MyCluster/abcd1234-5678-90ab-cdef-1234567890ab/abcd1234-5678-90ab-cdef-1234567890ab",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/MyCluster/
abcd1234-5678-90ab-cdef-1234567890ab",
                    "name": "nginx",
                    "lastStatus": "RUNNING",
                    "networkBindings": [
                        {
                            "bindIP": "0.0.0.0",
                            "containerPort": 80,
                            "hostPort": 80,
                            "protocol": "tcp"
                        }
                    ]
                }
            ],
            "createdAt": "2025-01-13T10:45:00.000Z",
            "startedAt": "2025-01-13T10:45:30.000Z"
        }
```

```
    ]
}
```

Uji server web

Untuk menguji server web

1. Ambil alamat IP publik instance container Anda dengan menjalankan perintah berikut.

   ```
   aws ec2 describe-instances --instance-ids i-abcd1234 --query
    'Reservations[0].Instances[0].PublicIpAddress' --output text
   ```

   Output:

   ```
   203.0.113.25
   ```

2. Setelah mengambil alamat IP, jalankan `curl` perintah berikut dengan alamat IP.

   ```
   curl http://203.0.113.25
   ```

   Output:

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   ...
   </head>
   <body>
   <h1>Welcome to nginx!</h1>
   <p>If you can see this page, the nginx web server is successfully installed and
    working.</p>
   ...
   </body>
   </html>
   ```

   Halaman selamat datang nginx mengonfirmasi bahwa layanan Anda berjalan dengan sukses dan dapat diakses dari internet.

Pembersihan sumber daya

Untuk menghindari biaya yang timbul, bersihkan sumber daya yang Anda buat dalam tutorial ini.

Untuk membersihkan sumber daya

1. Perbarui layanan agar tidak memiliki tugas yang diinginkan, lalu hapus layanan.

```
aws ecs update-service --cluster MyCluster --service nginx-service --desired-count
 0
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/nginx-
service",
        "serviceName": "nginx-service",
        "desiredCount": 0,
        "runningCount": 1,
        "pendingCount": 0,
        "status": "ACTIVE"
    }
}
```

2. Tunggu tugas yang berjalan berhenti, lalu hapus layanan.

```
aws ecs delete-service --cluster MyCluster --service nginx-service
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/nginx-
service",
        "serviceName": "nginx-service",
        "status": "DRAINING"
    }
}
```

3. Hentikan instance kontainer yang Anda buat.

```
aws ec2 terminate-instances --instance-ids i-abcd1234
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-abcd1234",
            "CurrentState": {
                "Code": 32,
```

```
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

4. Bersihkan grup keamanan dan key pair yang Anda buat.

```
aws ec2 delete-security-group --group-id sg-abcd1234
aws ec2 delete-key-pair --key-name ecs-tutorial-key
rm ecs-tutorial-key.pem
```

5. Hapus cluster Amazon ECS.

```
aws ecs delete-cluster --cluster MyCluster
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "INACTIVE"
    }
}
```

Mengonfigurasi Amazon ECS untuk mendengarkan acara Acara CloudWatch

Pelajari cara menyiapkan fungsi Lambda sederhana yang mendengarkan peristiwa tugas dan menuliskannya ke aliran CloudWatch log Log.

Prasyarat: Atur klaster pengujian

Jika Anda tidak memiliki klaster yang berjalan untuk menangkap peristiwa, ikuti langkah-langkah dalam the section called “Membuat cluster untuk tipe peluncuran Fargate” untuk membuat klaster. Di akhir tutorial ini, Anda menjalankan tugas di cluster ini untuk menguji apakah Anda telah mengonfigurasi fungsi Lambda Anda dengan benar.

Langkah 1: Buat fungsi Lambda

Dalam prosedur ini, Anda membuat fungsi Lambda sederhana untuk berfungsi sebagai target pesan aliran acara Amazon ECS.

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pada layar Tulis dari awal, lakukan hal berikut:
   a. Untuk Nama, masukkan nilai.
   b. Untuk Runtime, pilih versi Python Anda, misalnya, Python 3.9.
   c. Untuk Peran, pilih Buat peran baru dengan izin Lambda dasar.
4. Pilih Buat fungsi.
5. Di bagian Kode fungsi, edit kode sampel untuk mencocokkan contoh berikut:

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source
 type of: aws.ecs")

    print('Here is the event:')
    print(json.dumps(event))
```

   Ini adalah fungsi Python 3.9 sederhana yang mencetak acara yang dikirim oleh Amazon ECS. Jika semuanya dikonfigurasi dengan benar, di akhir tutorial ini, Anda melihat bahwa detail peristiwa muncul di aliran CloudWatch log Log yang terkait dengan fungsi Lambda ini.
6. Pilih Simpan.

Langkah 2: Mendaftarkan aturan peristiwa

Selanjutnya, Anda membuat aturan CloudWatch acara Acara yang menangkap peristiwa tugas yang berasal dari kluster Amazon ECS Anda. Aturan ini menangkap semua peristiwa dari semua klaster dalam akun yang ditentukan. Pesan tugas sendiri berisi informasi sumber peristiwa, termasuk klaster tempat klaster berada, yang dapat Anda gunakan untuk memfilter dan mengurutkan peristiwa secara terprogram.

> **Note**
>
> Saat Anda menggunakan aturan AWS Management Console untuk membuat acara, konsol secara otomatis menambahkan izin IAM yang diperlukan untuk memberikan izin CloudWatch Acara untuk memanggil fungsi Lambda Anda. Jika Anda membuat aturan acara menggunakan AWS CLI, Anda harus memberikan izin ini secara eksplisit. Untuk informasi selengkapnya, lihat Peristiwa di Amazon EventBridge dan pola EventBridge acara Amazon di Panduan EventBridge Pengguna Amazon.

Untuk merutekan acara ke fungsi Lambda Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Peristiwa, Aturan, Buat aturan.
3. Untuk Sumber Peristiwa, pilih ECS sebagai sumber peristiwa. Secara default, aturan ini berlaku untuk semua peristiwa Amazon ECS untuk semua grup Amazon ECS Anda. Atau, Anda dapat memilih acara tertentu atau grup Amazon ECS tertentu.
4. Untuk Target, pilih Tambah target, untuk jenis Target, pilih fungsi Lambda, lalu pilih fungsi Lambda Anda.
5. Pilih Konfigurasikan detail.
6. Untuk Definisi aturan, tuliskan nama dan deskripsi untuk aturan Anda dan pilih Buat aturan.

Langkah 3: Membuat sebuah penetapan tugas

Buat definisi tugas.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Ketentuan Tugas.
3. Pilih Buat Definisi Tugas baru, Buat revisi baru dengan JSON.
4. Salin dan tempel contoh ketentuan tugas berikut ke dalam kotak dan kemudian pilih Simpan.

```
{
   "containerDefinitions": [
      {
         "command": [
```

```
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "public.ecr.aws/docker/library/httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

5. Pilih Buat.

Langkah 4: Uji aturan Anda

Terakhir, Anda membuat aturan CloudWatch acara Acara yang menangkap peristiwa tugas yang berasal dari kluster Amazon ECS Anda. Aturan ini menangkap semua peristiwa dari semua klaster dalam akun yang ditentukan. Pesan tugas berisi informasi tentang sumber peristiwa, termasuk klaster tempat klaster berada, yang dapat Anda gunakan untuk memfilter dan mengurutkan peristiwa secara terprogram.

Menguji aturan Anda

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pilih Definisi tugas.
3. Pilih console-sample-app-static, lalu pilih Deploy, Run new task.
4. Untuk Cluster, pilih default, lalu pilih Deploy.
5. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
6. Pada panel navigasi, pilih Log dan pilih grup log untuk fungsi Lambda Anda (misalnya, *my-function* /aws/lambda/).
7. Pilih stream log untuk melihat data peristiwa.

Mengirim peringatan Amazon Simple Notification Service untuk tugas Amazon ECS menghentikan peristiwa

Konfigurasikan aturan EventBridge peristiwa Amazon yang hanya menangkap peristiwa tugas di mana tugas telah berhenti berjalan karena salah satu wadah pentingnya telah dihentikan. Acara hanya mengirimkan peristiwa tugas dengan `stoppedReason` properti tertentu ke topik Amazon SNS yang ditentukan.

Prasyarat: Atur klaster pengujian

Jika Anda tidak memiliki cluster yang sedang berjalan untuk menangkap peristiwa, ikuti langkah-langkah di Memulai konsol menggunakan kontainer Linux AWS Fargate untuk membuatnya. Di akhir tutorial ini, Anda menjalankan tugas di cluster ini untuk menguji bahwa Anda telah mengonfigurasi topik dan EventBridge aturan Amazon SNS Anda dengan benar.

Prasyarat: Konfigurasikan izin untuk Amazon SNS

Untuk memungkinkan mempublikasikan EventBridge ke topik Amazon SNS, gunakan perintah aws sns get-topic-attributes dan aws sns. set-topic-attributes

Untuk informasi tentang cara menambahkan izin, lihat izin Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon

Tambahkan izin berikut:

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sns: Publish",
  "Resource": "arn:aws:sns:region:account-id:TaskStoppedAlert",
}
```

Langkah 1: Buat dan berlangganan ke topik Amazon SNS

Untuk tutorial ini, Anda mengonfigurasi topik Amazon SNS yang berfungsi sebagai target peristiwa untuk aturan peristiwa baru Anda.

Untuk informasi tentang cara membuat dan berlangganan topik Amazon SNS, lihat Memulai Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Tipe	Standar
Nama	TaskStoppedAlert
Protokol	Email
Titik Akhir	Alamat email yang saat ini Anda miliki akses

Langkah 2: Mendaftarkan aturan peristiwa

Selanjutnya, Anda mendaftarkan aturan peristiwa yang hanya menangkap peristiwa yang dihentikan tugas untuk tugas dengan kontainer yang dihentikan.

Untuk informasi tentang cara membuat dan berlangganan topik Amazon SNS, lihat Membuat aturan di Amazon EventBridge di Panduan EventBridge Pengguna Amazon dan gunakan tabel berikut untuk menentukan opsi apa yang harus dipilih.

Opsi	Nilai
Jenis aturan	Aturan dengan pola acara
Sumber peristiwa	AWS acara atau acara EventBridge mitra
Pola peristiwa	Pola kustom (editor JSON)

Pola peristiwa	<pre>{
	"source":[
	"aws.ecs"
	],
	"detail-type":[
	"ECS Task State Change"
	],
	"detail":{
	"lastStatus":[
	"STOPPED"
	],
	"stoppedReason":[
	"Essential container in task exited"
	]
	}
	}</pre>

Tipe target	AWS layanan
Target	Topik SNS

Opsi	Nilai	
Topik	TaskStoppedAlert (Topik yang Anda buat di Langkah 1)	

Langkah 3: Uji aturan Anda

Verifikasi bahwa aturan bekerja dengan menjalankan tugas yang keluar segera setelah dimulai. Jika aturan peristiwa Anda dikonfigurasi dengan benar, Anda menerima pesan email dalam beberapa menit dengan teks peristiwa. Jika Anda memiliki ketentuan tugas yang sudah ada yang dapat memenuhi persyaratan aturan, jalankan tugas dengan menggunakannya. Jika tidak memilikinya, langkah-langkah berikut akan memandu Anda mendaftarkan ketentuan tugas Fargate dan menjalankannya.

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih Buat definisi tugas baru, Buat definisi tugas baru dengan JSON.
4. Di kotak editor JSON, edit file JSON Anda, salin yang berikut ini ke editor.

```
{
   "containerDefinitions":[
      {
         "command":[
            "sh",
            "-c",
            "sleep 5"
         ],
         "essential":true,
         "image":"public.ecr.aws/amazonlinux/amazonlinux:latest",
         "name":"test-sleep"
      }
   ],
   "cpu":"256",
   "executionRoleArn":"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family":"fargate-task-definition",
   "memory":"512",
   "networkMode":"awsvpc",
   "requiresCompatibilities":[
      "FARGATE"
```

```
    ]
}
```

5. Pilih Buat.

Untuk menjalankan tugas dari konsol

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada halaman Clusters, pilih cluster yang Anda buat dalam prasyarat.
3. Dari tab Tugas, pilih Jalankan tugas baru.
4. Untuk Jenis aplikasi, pilih Tugas.
5. Untuk definisi Tugas, pilih fargate-task-definition.
6. Untuk tugas yang diinginkan, masukkan jumlah tugas yang akan diluncurkan.
7. Pilih Buat.

Menggabungkan pesan log Amazon ECS multiline atau stack-trace

Dimulai dengan AWS untuk Fluent Bit versi 2.22.0, filter multiline disertakan. Filter multiline membantu menggabungkan pesan log yang awalnya milik satu konteks tetapi dibagi menjadi beberapa catatan atau baris log. Untuk informasi selengkapnya tentang filter multiline, lihat dokumentasi Fluent Bit.

Contoh umum pesan log terpisah adalah:

- Jejak tumpukan.
- Aplikasi yang mencetak log pada beberapa baris.
- Log pesan yang dibagi karena lebih panjang dari ukuran buffer maks runtime yang ditentukan. Anda dapat menggabungkan pesan log yang dibagi berdasarkan runtime container dengan mengikuti contoh di GitHub: FireLens Contoh: Concatenate Container Logs. Partial/Split

Izin IAM yang diperlukan

Anda memiliki izin IAM yang diperlukan untuk agen penampung untuk menarik gambar kontainer dari Amazon ECR dan wadah untuk merutekan log ke Log. CloudWatch

Untuk izin ini, Anda harus memiliki peran berikut:

- Peran tugas IAM.
- Peran IAM eksekusi tugas.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Tentukan kapan harus menggunakan pengaturan log multiline

Berikut ini adalah contoh cuplikan log yang Anda lihat di konsol CloudWatch Log dengan pengaturan log default. Anda dapat melihat garis yang dimulai dengan `log` untuk menentukan apakah Anda memerlukan filter multiline. Ketika konteksnya sama, Anda dapat menggunakan pengaturan log multiline, Dalam contoh ini, konteksnya adalah “com.myproject.model. MyProject”.

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-
app", "source=": "stdout", "log": ": "     at com.myproject.modele.
(MyProject.badMethod.java:22)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at com.myproject.model.MyProject.badMethod(MyProject.java:22)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app", "stdout",
 "log": ": "     at com.myproject.modele.(MyProject.oneMoreMethod.java:18)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at
 com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE,
```

```
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

Setelah Anda menggunakan pengaturan log multiline, output akan terlihat mirip dengan contoh di bawah ini.

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app",
 "stdout",...
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log:    "September 20, 2022 06:41:48 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
    at com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)
 com.myproject.module.MyProject.main(MyProject.java:6)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:2"
     }
```

Parse dan concatenate pilihan

Untuk mengurai log dan menggabungkan baris yang terbagi karena baris baru, Anda dapat menggunakan salah satu dari dua opsi ini.

- Gunakan file parser Anda sendiri yang berisi aturan untuk mengurai dan menggabungkan baris yang termasuk dalam pesan yang sama.
- Gunakan parser bawaan Fluent Bit. Untuk daftar bahasa yang didukung oleh parser bawaan Fluent Bit, lihat Dokumentasi Bit Lancar.

Tutorial berikut memandu Anda melalui langkah-langkah untuk setiap kasus penggunaan. Langkah-langkah menunjukkan cara menggabungkan multiline dan mengirim log ke Amazon. CloudWatch Anda dapat menentukan tujuan yang berbeda untuk log Anda.

Contoh: Gunakan parser yang Anda buat

Dalam contoh ini, Anda akan menyelesaikan langkah-langkah berikut:

1. Buat dan unggah gambar untuk wadah Fluent Bit.
2. Buat dan unggah gambar untuk aplikasi multiline demo yang berjalan, gagal, dan menghasilkan jejak tumpukan multiline.
3. Buat definisi tugas dan jalankan tugas.
4. Lihat log untuk memverifikasi bahwa pesan yang menjangkau beberapa baris tampak digabungkan.

Buat dan unggah gambar untuk wadah Fluent Bit

Gambar ini akan menyertakan file parser tempat Anda menentukan ekspresi reguler dan file konfigurasi yang mereferensikan file parser.

1. Buat folder dengan nama`FluentBitDockerImage`.
2. Di dalam folder, buat file parser yang berisi aturan untuk mengurai log dan menggabungkan baris yang termasuk dalam pesan yang sama.

    a. Tempel konten berikut di file parser:

```
[MULTILINE_PARSER]
    name          multiline-regex-test
    type          regex
    flush_timeout 1000
    #
    # Regex rules for multiline parsing
    # ---------------------------------
    #
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules |   state name  | regex pattern                  | next state
    # ------|---------------|--------------------------------------------
    rule      "start_state"   "/(Dec \d+ \d+\:\d+\:\d+)(.*)/"  "cont"
    rule      "cont"          "/^\s+at.*/"                     "cont"
```

Saat Anda menyesuaikan pola regex Anda, kami sarankan Anda menggunakan editor ekspresi reguler untuk menguji ekspresi.

b. Simpan file sebagai `parsers_multiline.conf`.

3. Di dalam `FluentBitDockerImage` folder, buat file konfigurasi khusus yang mereferensikan file parser yang Anda buat di langkah sebelumnya.

   Untuk informasi selengkapnya tentang file konfigurasi kustom, lihat Menentukan file konfigurasi kustom di Panduan Pengembang Layanan Amazon Elastic Container

   a. Tempel konten berikut dalam file:

```
[SERVICE]
    flush                1
    log_level            info
    parsers_file         /parsers_multiline.conf

[FILTER]
    name                 multiline
    match                *
    multiline.key_content log
    multiline.parser     multiline-regex-test
```

> **Note**
>
> Anda harus menggunakan jalur absolut parser.

   b. Simpan file sebagai `extra.conf`.

4. Di dalam `FluentBitDockerImage` folder, buat Dockerfile dengan gambar Fluent Bit dan parser dan file konfigurasi yang Anda buat.

   a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest

ADD parsers_multiline.conf /parsers_multiline.conf
ADD extra.conf /extra.conf
```

   b. Simpan file sebagai `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar Fluent Bit kustom dengan parser dan file konfigurasi khusus disertakan.

   > **Note**
   >
   > Anda dapat menempatkan file parser dan file konfigurasi di mana saja di gambar Docker kecuali `/fluent-bit/etc/fluent-bit.conf` karena jalur file ini digunakan oleh. FireLens

   a. Bangun gambar: `docker build -t fluent-bit-multiline-image.`

      Dimana: `fluent-bit-multiline-image` adalah nama untuk gambar dalam contoh ini.

   b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=fluent-bit-multiline-image`

      Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar Fluent Bit kustom ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

      Dimana: `fluent-bit-multiline-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

      Output memberi Anda rincian repositori baru.

   b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag` *`fluent-bit-multiline-image repositoryUri`*

      Contoh: `docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

   c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

      Dalam output, nama repositori berubah dari fluent-bit-multiline-repo ke. `repositoryUri`

   d. Otentikasi ke Amazon ECR dengan menjalankan `aws ecr get-login-password` perintah dan menentukan ID registri yang ingin Anda autentikasi: `aws ecr get-login-`

`password | docker login --username AWS --password-stdin registry ID.dkr.ecr.region.amazonaws.com`

Contoh: `ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

Pesan login yang berhasil muncul.

e. Dorong gambar ke Amazon ECR: `docker push registry ID.dkr.ecr.region.amazonaws.com/repository name`

Contoh: `docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

Membangun dan meng-upload gambar untuk aplikasi demo multiline

Gambar ini akan mencakup file skrip Python yang menjalankan aplikasi dan file log sampel.

Saat Anda menjalankan tugas, aplikasi mensimulasikan berjalan, lalu gagal dan membuat jejak tumpukan.

1. Buat folder bernama`multiline-app`: `mkdir multiline-app`
2. Buat file skrip Python.

   a. Di dalam `multiline-app` folder, buat file dan beri nama`main.py`.

   b. Tempel konten berikut dalam file:

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
```

```
print(count)
print("app terminated.")
```

c. Simpan file `main.py`.

3. Buat file log sampel.

a. Di dalam `multiline-app` folder, buat file dan beri nama`test.log`.

b. Tempel konten berikut dalam file:

```
single line...
Dec 14 06:41:08 Exception in thread "main" java.lang.RuntimeException:
 Something has gone wrong, aborting!
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)
    at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
    at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)
    at com.myproject.module.MyProject.someMethod(MyProject.java:10)
    at com.myproject.module.MyProject.main(MyProject.java:6)
another line...
```

c. Simpan file `test.log`.

4. Di dalam `multiline-app` folder, buat Dockerfile.

a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. Simpan file `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar.

a. Bangun gambar: `docker build -t multiline-app-image`

Dimana: `multiline-app-image` adalah nama untuk gambar dalam contoh ini.

b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=multiline-app-image`

   Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

      Dimana: `multiline-app-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

      Output memberi Anda rincian repositori baru. Perhatikan `repositoryUri` nilainya karena Anda akan membutuhkannya di langkah selanjutnya.

   b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag multiline-app-image repositoryUri`

      Contoh: `docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

   c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=repositoryUri`

      Dalam output, nama repositori berubah dari `multiline-app-repo` ke nilai. `repositoryUri`

   d. Dorong gambar ke Amazon ECR: `docker push aws_account_id.dkr.ecr.region.amazonaws.com/repository name`

      Contoh: `docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

Buat definisi tugas dan jalankan tugas

1. Buat file definisi tugas dengan nama file`multiline-task-definition.json`.
2. Tempel konten berikut dalam `multiline-task-definition.json` file:

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN,
```

```
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

Ganti yang berikut ini dalam definisi `multiline-task-definition.json` tugas:

a. *task role ARN*

Untuk menemukan peran tugas ARN, buka konsol IAM. Pilih Peran dan temukan peran `ecs-task-role-for-firelens` tugas yang Anda buat. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

b. *`execution role ARN`*

Untuk menemukan peran eksekusi ARN, buka konsol IAM. Pilih Peran dan temukan `ecsTaskExecutionRole` perannya. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

c. *`aws_account_id`*

Untuk menemukan Anda`aws_account_id`, masuk ke AWS Management Console. Pilih nama pengguna Anda di kanan atas dan salin ID Akun Anda.

d. *`us-east-1`*

Ganti wilayah jika perlu.

3. Daftarkan file definisi tugas: `aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region` *`region`*
4. Buka konsol di https://console.aws.amazon.com/ecs/v2.
5. Di panel navigasi, pilih Definisi Tugas dan kemudian pilih `firelens-example-multiline` keluarga karena kami mendaftarkan definisi tugas ke keluarga ini di baris pertama definisi tugas di atas.
6. Pilih versi terbaru.
7. Pilih tugas Deploy, Run.
8. Pada halaman Jalankan Tugas, Untuk Cluster, pilih cluster, dan kemudian di bawah Jaringan, untuk Subnet, pilih subnet yang tersedia untuk tugas Anda.
9. Pilih Buat.

Verifikasi bahwa pesan log multiline di Amazon CloudWatch tampak digabungkan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dari panel navigasi, perluas Log dan pilih Grup log.
3. Pilih grup `multiline-test/applicatio` log.
4. Pilih log. Lihat pesan. Baris yang cocok dengan aturan dalam file parser digabungkan dan muncul sebagai pesan tunggal.

Cuplikan log berikut menunjukkan baris yang digabungkan dalam satu peristiwa jejak tumpukan Java:

```
{
    "container_id": "xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
 at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
 com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)\n
 at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)\n
 at com.myproject.module.MyProject.someMethod(MyProject.java:10)\n    at
 com.myproject.module.MyProject.main(MyProject.java:6)",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

Cuplikan log berikut menunjukkan bagaimana pesan yang sama muncul hanya dengan satu baris jika Anda menjalankan wadah Amazon ECS yang tidak dikonfigurasi untuk menggabungkan pesan log multiline.

```
{
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
}
```

Contoh: Gunakan parser bawaan Fluent Bit

Dalam contoh ini, Anda akan menyelesaikan langkah-langkah berikut:

1. Buat dan unggah gambar untuk wadah Fluent Bit.

2. Buat dan unggah gambar untuk aplikasi multiline demo yang berjalan, gagal, dan menghasilkan jejak tumpukan multiline.
3. Buat definisi tugas dan jalankan tugas.
4. Lihat log untuk memverifikasi bahwa pesan yang menjangkau beberapa baris tampak digabungkan.

Buat dan unggah gambar untuk wadah Fluent Bit

Gambar ini akan menyertakan file konfigurasi yang mereferensikan parser Fluent Bit.

1. Buat folder dengan nama`FluentBitDockerImage`.
2. Di dalam `FluentBitDockerImage` folder, buat file konfigurasi khusus yang mereferensikan file parser bawaan Fluent Bit.

   Untuk informasi selengkapnya tentang file konfigurasi kustom, lihat Menentukan file konfigurasi kustom di Panduan Pengembang Layanan Amazon Elastic Container

   a. Tempel konten berikut dalam file:

   ```
   [FILTER]
       name                  multiline
       match                 *
       multiline.key_content log
       multiline.parser      go
   ```

   b. Simpan file sebagai `extra.conf`.
3. Di dalam `FluentBitDockerImage` folder, buat Dockerfile dengan gambar Fluent Bit dan parser dan file konfigurasi yang Anda buat.

   a. Tempel konten berikut dalam file:

   ```
   FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
   ADD extra.conf /extra.conf
   ```

   b. Simpan file sebagai `Dockerfile`.
4. Menggunakan Dockerfile, buat gambar Fluent Bit kustom dengan file konfigurasi khusus yang disertakan.

> **Note**
>
> Anda dapat menempatkan file konfigurasi di mana saja di gambar Docker kecuali `/fluent-bit/etc/fluent-bit.conf` karena jalur file ini digunakan oleh FireLens.

a. Bangun gambar: `docker build -t fluent-bit-multiline-image.`

   Dimana: `fluent-bit-multiline-image` adalah nama untuk gambar dalam contoh ini.

b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=fluent-bit-multiline-image`

   Jika berhasil, output menunjukkan gambar dan `latest` tag.

5. Unggah gambar Fluent Bit kustom ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

      Dimana: `fluent-bit-multiline-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

      Output memberi Anda rincian repositori baru.

   b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag` *`fluent-bit-multiline-image repositoryUri`*

      Contoh: `docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

   c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

      Dalam output, nama repositori berubah dari fluent-bit-multiline-repo ke. `repositoryUri`

   d. Otentikasi ke Amazon ECR dengan menjalankan `aws ecr get-login-password` perintah dan menentukan ID registri yang ingin Anda autentikasi: `aws ecr get-login-password | docker login --username AWS --password-stdin` *`registry ID`*`.dkr.ecr.`*`region`*`.amazonaws.com`

Contoh: `ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

Pesan login yang berhasil muncul.

e. Dorong gambar ke Amazon ECR: `docker push` *registry ID*`.dkr.ecr.`*region*`.amazonaws.com/`*repository name*

Contoh: `docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

Membangun dan meng-upload gambar untuk aplikasi demo multiline

Gambar ini akan mencakup file skrip Python yang menjalankan aplikasi dan file log sampel.

1. Buat folder bernama`multiline-app`: `mkdir multiline-app`
2. Buat file skrip Python.

   a. Di dalam `multiline-app` folder, buat file dan beri nama`main.py`.

   b. Tempel konten berikut dalam file:

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

   c. Simpan file `main.py`.

3. Buat file log sampel.

   a. Di dalam `multiline-app` folder, buat file dan beri nama`test.log`.

   b. Tempel konten berikut dalam file:

```
panic: my panic

goroutine 4 [running]:
panic(0x45cb40, 0x47ad70)
  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8 sp=0xc42003f710
 pc=0x422f7c
main.main.func1(0xc420024120)
  foo.go:6 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0
 sp=0xc42003f7d8 pc=0x44b4d1
created by main.main
  foo.go:5 +0x58

goroutine 1 [chan receive]:
runtime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00
 pc=0x42503c
runtime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e
runtime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4
runtime.chanrecv1(0xc420024120, 0x0)
  /usr/local/go/src/runtime/chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20
 pc=0x40439b
main.main()
  foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef
runtime.main()
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc420053fe8
 sp=0xc420053fe0 pc=0x44b4d1

goroutine 2 [force gc (idle)]:
runtime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)
```

```
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c
runtime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e
runtime.forcegchelper()
  /usr/local/go/src/runtime/proc.go:238 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8
 pc=0x424e5c
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8
 sp=0xc42003e7e0 pc=0x44b4d1
created by runtime.init.4
  /usr/local/go/src/runtime/proc.go:227 +0x35

goroutine 3 [GC sweep wait]:
runtime.gopark(0x4739b8, 0x4ad7e0, 0x46fdd2, 0xd, 0x419914, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003ef60 sp=0xc42003ef30
 pc=0x42503c
runtime.goparkunlock(0x4ad7e0, 0x46fdd2, 0xd, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003efa0 sp=0xc42003ef60
 pc=0x42512e
runtime.bgsweep(0xc42001e150)
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8
 sp=0xc42003efa0 pc=0x419973
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003efe0
 sp=0xc42003efd8 pc=0x44b4d1
created by runtime.gcenable
  /usr/local/go/src/runtime/mgc.go:216 +0x58
one more line, no multiline
```

c. Simpan file `test.log`.

4. Di dalam `multiline-app` folder, buat Dockerfile.

   a. Tempel konten berikut dalam file:

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin
```

```
COPY main.py .

CMD ["python3", "main.py"]
```

b. Simpan file `Dockerfile`.

5. Menggunakan Dockerfile, buat gambar.

   a. Bangun gambar: `docker build -t multiline-app-image`

      Dimana: `multiline-app-image` adalah nama untuk gambar dalam contoh ini.

   b. Verifikasi bahwa gambar telah dibuat dengan benar: `docker images –filter reference=multiline-app-image`

      Jika berhasil, output menunjukkan gambar dan `latest` tag.

6. Unggah gambar ke Amazon Elastic Container Registry.

   a. Buat repositori Amazon ECR untuk menyimpan gambar: `aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

      Dimana: `multiline-app-repo` adalah nama untuk repositori dan `us-east-1` merupakan wilayah dalam contoh ini.

      Output memberi Anda rincian repositori baru. Perhatikan `repositoryUri` nilainya karena Anda akan membutuhkannya di langkah selanjutnya.

   b. Tandai gambar Anda dengan `repositoryUri` nilai dari output sebelumnya: `docker tag` *`multiline-app-image repositoryUri`*

      Contoh: `docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

   c. Jalankan docker image untuk memverifikasi itu berjalan dengan benar: `docker images –filter reference=`*`repositoryUri`*

      Dalam output, nama repositori berubah dari `multiline-app-repo` ke nilai. `repositoryUri`

   d. Dorong gambar ke Amazon ECR: `docker push` *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository name`*

      Contoh: `docker push` *`xxxxxxxxxxxx`*`.dkr.ecr.`*`us-east-1`*`.amazonaws.com/multiline-app-repo`

Buat definisi tugas dan jalankan tugas

1. Buat file definisi tugas dengan nama file`multiline-task-definition.json`.
2. Tempel konten berikut dalam `multiline-task-definition.json` file:

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN",
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
```

```
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

Ganti yang berikut ini dalam definisi `multiline-task-definition.json` tugas:

a. *`task role ARN`*

Untuk menemukan peran tugas ARN, buka konsol IAM. Pilih Peran dan temukan peran `ecs-task-role-for-firelens` tugas yang Anda buat. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

b. *`execution role ARN`*

Untuk menemukan peran eksekusi ARN, buka konsol IAM. Pilih Peran dan temukan `ecsTaskExecutionRole` perannya. Pilih peran dan salin ARN yang muncul di bagian Ringkasan.

c. *`aws_account_id`*

Untuk menemukan Anda`aws_account_id`, masuk ke AWS Management Console. Pilih nama pengguna Anda di kanan atas dan salin ID Akun Anda.

d. *`us-east-1`*

Ganti wilayah jika perlu.

3. Daftarkan file definisi tugas: `aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region us-east-1`
4. Buka konsol di https://console.aws.amazon.com/ecs/v2.
5. Di panel navigasi, pilih Definisi Tugas dan kemudian pilih `firelens-example-multiline` keluarga karena kami mendaftarkan definisi tugas ke keluarga ini di baris pertama definisi tugas di atas.
6. Pilih versi terbaru.
7. Pilih tugas Deploy, Run.
8. Pada halaman Jalankan Tugas, Untuk Cluster, pilih cluster, dan kemudian di bawah Jaringan, untuk Subnet, pilih subnet yang tersedia untuk tugas Anda.
9. Pilih Buat.

Verifikasi bahwa pesan log multiline di Amazon CloudWatch tampak digabungkan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dari panel navigasi, perluas Log dan pilih Grup log.
3. Pilih grup `multiline-test/applicatio` log.
4. Pilih log dan lihat pesan. Baris yang cocok dengan aturan dalam file parser digabungkan dan muncul sebagai pesan tunggal.

   Cuplikan log berikut menunjukkan jejak tumpukan Go yang digabungkan menjadi satu peristiwa:

```
{
    "log": "panic: my panic\n\ngoroutine 4 [running]:\npanic(0x45cb40,
 0x47ad70)\n  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8
 sp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n  foo.go:6
 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8
 pc=0x44b4d1\ncreated by main.main\n  foo.go:5 +0x58\n\ngoroutine 1 [chan receive]:
\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n  /usr/
local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c
\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n  /usr/local/go/src/runtime/
chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n
 foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef\nruntime.main()\n
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337
 +0x1 fp=0xc420053fe8 sp=0xc420053fe0 pc=0x44b4d1\n\ngoroutine 2 [force gc
 (idle)]:\nruntime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)\n  /
usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c\nruntime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e\nruntime.forcegchelper()\n  /usr/local/go/src/runtime/proc.go:238
 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8 pc=0x424e5c\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8 sp=0xc42003e7e0
 pc=0x44b4d1\ncreated by runtime.init.4\n  /usr/local/go/src/runtime/proc.go:227
 +0x35\n\ngoroutine 3 [GC sweep wait]:\nruntime.gopark(0x4739b8, 0x4ad7e0,
 0x46fdd2, 0xd, 0x419914, 0x1)\n  /usr/local/go/src/runtime/proc.go:280 +0x12c
 fp=0xc42003ef60 sp=0xc42003ef30 pc=0x42503c\nruntime.goparkunlock(0x4ad7e0,
 0x46fdd2, 0xd, 0x14, 0x1)\n  /usr/local/go/src/runtime/proc.go:286 +0x5e
 fp=0xc42003efa0 sp=0xc42003ef60 pc=0x42512e\nruntime.bgsweep(0xc42001e150)\n
```

```
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8 sp=0xc42003efa0
 pc=0x419973\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1
 fp=0xc42003efe0 sp=0xc42003efd8 pc=0x44b4d1\ncreated by runtime.gcenable\n  /usr/
local/go/src/runtime/mgc.go:216 +0x58",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

Cuplikan log berikut menunjukkan bagaimana peristiwa yang sama muncul jika Anda menjalankan wadah ECS yang tidak dikonfigurasi untuk menggabungkan pesan log multiline. Bidang log berisi satu baris.

```
{
    "log": "panic: my panic",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
```

> **Note**
>
> Jika log Anda masuk ke file log alih-alih output standar, kami sarankan untuk menentukan parameter `multiline.parser` dan `multiline.key_content` konfigurasi di plugin input Tail alih-alih Filter.

Menyebarkan Fluent Bit di Amazon ECS Windows container

Fluent Bit adalah prosesor log dan router yang cepat dan fleksibel yang didukung oleh berbagai sistem operasi. Ini dapat digunakan untuk merutekan log ke berbagai AWS tujuan seperti Amazon CloudWatch Log, Firehose Amazon S3, dan Amazon Service. OpenSearch Fluent Bit mendukung solusi mitra umum seperti Datadog, Splunk, dan server HTTP kustom. Untuk informasi lebih lanjut tentang Fluent Bit, lihat Fluent Bit situs web.

Gambar AWS for Fluent Bit tersedia di Amazon ECR di Galeri Publik Amazon ECR dan di repositori ECR Amazon di sebagian besar Wilayah untuk ketersediaan tinggi. Untuk informasi lebih lanjut, lihat aws-for-fluent-bit di GitHub situs web.

Tutorial ini memandu Anda melalui cara menerapkan container Fluent Bit pada instance Windows mereka yang berjalan di Amazon ECS untuk mengalirkan log yang dihasilkan oleh tugas Windows ke Amazon CloudWatch untuk pencatatan terpusat.

Tutorial ini menggunakan pendekatan berikut:

- Fluent Bit berjalan sebagai layanan dengan strategi penjadwalan Daemon. Strategi ini memastikan bahwa satu instance Fluent Bit selalu berjalan pada instance container di cluster.
  - Mendengarkan pada port 24224 menggunakan plug-in input forward.
  - Ekspos port 24224 ke host sehingga runtime docker dapat mengirim log ke Fluent Bit menggunakan port yang terbuka ini.
  - Memiliki konfigurasi yang memungkinkan Fluent Bit untuk mengirim catatan log ke tujuan tertentu.
- Luncurkan semua wadah tugas Amazon ECS lainnya menggunakan driver logging lancar. Untuk informasi selengkapnya, lihat Driver logging fluentd di situs web dokumentasi Docker.
  - Docker terhubung ke soket TCP 24224 di localhost di dalam namespace host.
  - Agen Amazon ECS menambahkan label ke wadah yang mencakup nama cluster, nama keluarga definisi tugas, nomor revisi definisi tugas, ARN tugas, dan nama penampung. Informasi yang sama ditambahkan ke catatan log menggunakan opsi label dari driver logging docker fluentd. Untuk informasi selengkapnya, lihat label, labels-regex, env, dan env-regex di situs web dokumentasi Docker.
  - Karena `async` opsi driver logging fluentd disetel `ketrue`, ketika wadah Fluent Bit dimulai ulang, docker menyangga log hingga wadah Fluent Bit dimulai ulang. Anda dapat meningkatkan batas buffer dengan mengatur fluentd-buffer-limit opsi. Untuk informasi selengkapnya, lihat fluentd-buffer-limitdi situs web dokumentasi Docker.

Alur kerja adalah sebagai berikut:

- Kontainer Fluent Bit dimulai dan mendengarkan pada port 24224 yang diekspos ke host.
- Fluent Bit menggunakan kredensyal peran IAM tugas yang ditentukan dalam definisi tugasnya.
- Tugas lain yang diluncurkan pada instance yang sama menggunakan driver logging docker fluentd untuk terhubung ke wadah Fluent Bit pada port 24224.

- Saat wadah aplikasi menghasilkan log, docker runtime menandai catatan tersebut, menambahkan metadata tambahan yang ditentukan dalam label, dan kemudian meneruskannya di port 24224 di namespace host.
- Fluent Bit menerima catatan log pada port 24224 karena terkena namespace host.
- Fluent Bit melakukan pemrosesan internal dan merutekan log seperti yang ditentukan.

Tutorial ini menggunakan konfigurasi CloudWatch Fluent Bit default yang melakukan hal berikut:

- Membuat grup log baru untuk setiap cluster dan keluarga definisi tugas.
- Membuat aliran log baru untuk setiap wadah tugas di grup log yang dihasilkan di atas setiap kali tugas baru diluncurkan. Setiap aliran akan ditandai dengan id tugas tempat wadah berada.
- Menambahkan metadata tambahan termasuk nama cluster, ARN tugas, nama wadah tugas, keluarga definisi tugas, dan nomor revisi definisi tugas di setiap entri log.

  Misalnya, jika Anda memiliki `task_1` dengan `container_1` dan `container_2` dan t `ask_2` dengan`container_3`, maka berikut ini adalah aliran CloudWatch log:

  - `/aws/ecs/windows.ecs_task_1`

    `task-out.`*`TASK_ID`*`.container_1`

    `task-out.`*`TASK_ID`*`.container_2`
  - `/aws/ecs/windows.ecs_task_2`

    `task-out.`*`TASK_ID`*`.container_3`

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

Langkah-langkah

- Prasyarat
- Langkah 1: Buat peran akses IAM
- Langkah 2: Buat instance penampung Amazon ECS Windows

- Langkah 3: Konfigurasikan Bit Lancar
- Langkah 4: Daftarkan definisi tugas Windows Fluent Bit yang merutekan log CloudWatch
- Langkah 5: Jalankan definisi ecs-windows-fluent-bit tugas sebagai layanan Amazon ECS menggunakan strategi penjadwalan daemon
- Langkah 6: Daftarkan definisi tugas Windows yang menghasilkan log
- Langkah 7: Jalankan definisi windows-app-task tugas
- Langkah 8: Verifikasi log CloudWatch
- Langkah 9: Membersihkan

Prasyarat

Tutorial ini mengasumsikan bahwa prasyarat berikut telah diselesaikan:

- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru AWS CLI.
- Gambar `aws-for-fluent-bit` kontainer tersedia untuk sistem operasi Windows berikut:
  - Windows Server 2019 Core
  - Windows Server 2019 Full
  - Windows Server 2022 Inti
  - Windows Server 2022 Lengkap
- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah diselesaikan.
- Anda memiliki cluster. Dalam tutorial ini, nama cluster adalah FluentBit-cluster.
- Anda memiliki VPC dengan subnet publik tempat EC2 instance akan diluncurkan. Anda dapat menggunakan VPC default Anda. Anda juga dapat menggunakan subnet pribadi yang memungkinkan CloudWatch titik akhir Amazon mencapai subnet. Untuk informasi selengkapnya tentang CloudWatch titik akhir Amazon, lihat CloudWatch titik akhir dan kuota Amazon di. Referensi Umum AWS Untuk informasi tentang cara menggunakan wizard VPC Amazon untuk membuat VPC, lihat. the section called “Buat virtual private cloud”

Langkah 1: Buat peran akses IAM

Buat peran Amazon ECS IAM.

1. Buat peran instans penampung Amazon ECS bernama "ecsInstanceRole”. Untuk informasi selengkapnya, lihat peran IAM instans penampung Amazon ECS.
2. Buat peran IAM untuk tugas Fluent Bit bernama. `fluentTaskRole` Untuk informasi selengkapnya, lihat the section called “Tugas peran IAM”.

   Izin IAM yang diberikan dalam peran IAM ini diasumsikan oleh wadah tugas. Untuk memungkinkan Bit Lancar mengirim log CloudWatch, Anda harus melampirkan izin berikut ke peran IAM tugas.

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "logs:CreateLogStream",
               "logs:CreateLogGroup",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
           ],
           "Resource": "*"
       }
       ]
   }
   ```

3. Lampirkan kebijakan pada peran tersebut.

   a. Simpan konten di atas dalam file bernama`fluent-bit-policy.json`.

   b. Jalankan perintah berikut untuk melampirkan kebijakan inline ke peran `fluentTaskRole` IAM.

   ```
   aws iam put-role-policy --role-name fluentTaskRole --policy-name
    fluentTaskPolicy --policy-document file://fluent-bit-policy.json
   ```

Langkah 2: Buat instance penampung Amazon ECS Windows

Buat instance penampung Amazon ECS Windows.

Untuk membuat instans Amazon ECS

1. Gunakan `aws ssm get-parameters` perintah untuk mengambil ID AMI untuk Wilayah yang menghosting VPC Anda. Untuk informasi selengkapnya, lihat Mengambil metadata AMI yang dioptimalkan Amazon ECS.
2. Gunakan EC2 konsol Amazon untuk meluncurkan instance.
   a. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
   b. Dari bilah navigasi, pilih Wilayah untuk digunakan.
   c. Dari EC2 Dasbor, pilih Launch instance.
   d. Untuk Nama, masukkan nama unik.
   e. Untuk Gambar Aplikasi dan OS (Amazon Machine Image), pilih AMI yang Anda ambil pada langkah pertama.
   f. Untuk jenis Instance, pilih`t3.xlarge`.
   g. Untuk Key pair (login), pilih key pair.
   h. Di bawah Pengaturan jaringan, untuk grup Keamanan, pilih grup keamanan yang ada, atau buat yang baru.
   i. Di bawah Pengaturan jaringan, untuk Auto-assign IP Publik, pilih Aktifkan.
   j. Di bawah Detail lanjutan, untuk profil instans IAM, pilih ecsInstanceRole.
   k. Konfigurasikan instans penampung Amazon ECS Anda dengan data pengguna berikut. Di bawah Detail Lanjutan, tempelkan skrip berikut ke bidang Data pengguna, ganti *`cluster_name`* dengan nama cluster Anda.

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster cluster-name -EnableTaskENI -EnableTaskIAMRole -
LoggingDrivers '["awslogs","fluentd"]'
</powershell>
```

   l. Saat Anda siap, pilih bidang pengakuan, lalu pilih Luncurkan Instans.
   m. Halaman konfirmasi memberi tahu Anda bahwa instans Anda akan diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol tersebut.

Langkah 3: Konfigurasikan Bit Lancar

Anda dapat menggunakan konfigurasi default berikut yang disediakan oleh AWS untuk memulai dengan cepat:

- Amazon CloudWatch yang didasarkan pada plug-in Fluent Bit untuk Amazon CloudWatch pada Manual Resmi Fluent Bit.

Atau, Anda dapat menggunakan konfigurasi default lain yang disediakan oleh AWS. Untuk informasi selengkapnya, lihat Mengganti titik masuk untuk gambar Windows di situs web Github`aws-for-fluent-bit`.

Konfigurasi Amazon CloudWatch Fluent Bit default ditunjukkan di bawah ini.

Ganti variabel berikut:

- *`region`*dengan Wilayah tempat Anda ingin mengirim CloudWatch log Amazon.

```
[SERVICE]
    Flush               5
    Log_Level           info
    Daemon              off

[INPUT]
    Name                forward
    Listen              0.0.0.0
    Port                24224
    Buffer_Chunk_Size   1M
    Buffer_Max_Size     6M
    Tag_Prefix          ecs.

# Amazon ECS agent adds the following log keys as labels to the docker container.
# We would use fluentd logging driver to add these to log record while sending it to
 Fluent Bit.
[FILTER]
    Name                modify
    Match               ecs.*
    Rename              com.amazonaws.ecs.cluster ecs_cluster
    Rename              com.amazonaws.ecs.container-name ecs_container_name
    Rename              com.amazonaws.ecs.task-arn ecs_task_arn
```

```
    Rename              com.amazonaws.ecs.task-definition-family
 ecs_task_definition_family
    Rename              com.amazonaws.ecs.task-definition-version
 ecs_task_definition_version

[FILTER]
    Name                rewrite_tag
    Match               ecs.*
    Rule                $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9]+)$
 out.$3.$ecs_container_name false
    Emitter_Name        re_emitted

[OUTPUT]
    Name                cloudwatch_logs
    Match               out.*
    region              region
    log_group_name      fallback-group
    log_group_template  /aws/ecs/$ecs_cluster.$ecs_task_definition_family
    log_stream_prefix   task-
    auto_create_group   On
```

Setiap log yang masuk ke Fluent Bit memiliki tag yang Anda tentukan, atau dibuat secara otomatis ketika Anda tidak menyediakannya. Tag dapat digunakan untuk merutekan log yang berbeda ke tujuan yang berbeda. Untuk informasi tambahan, lihat Tag di Manual Resmi Fluent Bit.

Konfigurasi Fluent Bit yang dijelaskan di atas memiliki properti berikut:

- Plug-in input maju mendengarkan lalu lintas masuk pada port TCP 24224.
- Setiap entri log yang diterima pada port tersebut memiliki tag yang dimodifikasi oleh plug-in input forward untuk mengawali catatan dengan string. `ecs.`
- Pipeline internal Fluent Bit merutekan entri log untuk memodifikasi filter menggunakan regex Match. Filter ini menggantikan kunci dalam catatan log JSON ke format yang dapat dikonsumsi Fluent Bit.
- Entri log yang dimodifikasi kemudian dikonsumsi oleh filter rewrite_tag. Filter ini mengubah tag catatan log ke format keluar. *`TASK_ID`*. *`CONTAINER_NAME`*.
- Tag baru akan dirutekan ke plug-in cloudwatch_logs keluaran yang membuat grup log dan aliran seperti yang dijelaskan sebelumnya dengan menggunakan opsi dan plug-in keluaran. `log_group_template` `log_stream_prefix` CloudWatch Untuk informasi tambahan, lihat Parameter konfigurasi di Manual Resmi Bit Lancar.

Langkah 4: Daftarkan definisi tugas Windows Fluent Bit yang merutekan log CloudWatch

Daftarkan definisi tugas Windows Fluent Bit yang merutekan log ke CloudWatch.

> **Note**
>
> Definisi tugas ini mengekspos port kontainer Fluent Bit 24224 ke port host 24224. Pastikan port ini tidak terbuka di grup keamanan EC2 instans Anda untuk mencegah akses dari luar.

Untuk mendaftarkan ketentuan tugas

1. Buat file yang diberi nama `fluent-bit.json` dengan konten berikut ini.

   Ganti variabel berikut:

   - *`task-iam-role`*dengan Nama Sumber Daya Amazon (ARN) dari peran IAM tugas Anda
   - *`region`*dengan Wilayah tempat tugas Anda berjalan

```
{
  "family": "ecs-windows-fluent-bit",
  "taskRoleArn": "task-iam-role",
  "containerDefinitions": [
    {
      "name": "fluent-bit",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-
bit:windowsservercore-latest",
      "cpu": 512,
      "portMappings": [
        {
          "hostPort": 24224,
          "containerPort": 24224,
          "protocol": "tcp"
        }
      ],
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
```

```
      "command": [
        "C:\\entrypoint.ps1 -ConfigFile C:\\ecs_windows_forward_daemon\
\cloudwatch.conf"
      ],
      "environment": [
        {
          "name": "AWS_REGION",
          "value": "region"
        }
      ],
      "memory": 512,
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fluent-bit-logs",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "flb",
          "awslogs-create-group": "true"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. Jalankan perintah berikut untuk mendaftarkan definisi tugas.

```
aws ecs register-task-definition --cli-input-json file://fluent-bit.json --
region region
```

Anda dapat membuat daftar definisi tugas untuk akun Anda dengan menjalankan `list-task-definitions` perintah. Output menampilkan nilai keluarga dan revisi yang dapat Anda gunakan bersama dengan `run-task` atau`start-task`.

Langkah 5: Jalankan definisi **`ecs-windows-fluent-bit`** tugas sebagai layanan Amazon ECS menggunakan strategi penjadwalan daemon

Setelah Anda mendaftarkan definisi tugas untuk akun Anda, Anda dapat menjalankan tugas di klaster. Untuk tutorial ini, Anda menjalankan satu contoh definisi `ecs-windows-fluent-bit:1`

tugas di `FluentBit-cluster` cluster Anda. Jalankan tugas dalam layanan yang menggunakan strategi penjadwalan daemon, yang memastikan bahwa satu instance Fluent Bit selalu berjalan pada setiap instance container Anda.

Untuk menjalankan tugas

1. Jalankan perintah berikut untuk memulai definisi `ecs-windows-fluent-bit:1` tugas (terdaftar pada langkah sebelumnya) sebagai layanan.

   > **Note**
   >
   > Definisi tugas ini menggunakan driver `awslogs` logging, instance container Anda harus memiliki izin yang diperlukan.

   Ganti variabel berikut:

   - *`region`*dengan Wilayah tempat layanan Anda berjalan

   ```
   aws ecs create-service \
       --cluster FluentBit-cluster \
       --service-name FluentBitForwardDaemonService \
       --task-definition ecs-windows-fluent-bit:1 \
       --launch-type EC2 \
       --scheduling-strategy DAEMON \
       --region region
   ```

2. Jalankan perintah berikut untuk membuat daftar tugas Anda.

   Ganti variabel berikut:

   - *`region`*dengan Wilayah tempat tugas layanan Anda berjalan

   ```
   aws ecs list-tasks --cluster FluentBit-cluster --region region
   ```

Langkah 6: Daftarkan definisi tugas Windows yang menghasilkan log

Daftarkan definisi tugas yang menghasilkan log. Definisi tugas ini menyebarkan gambar kontainer Windows yang akan menulis angka tambahan untuk `stdout` setiap detik.

Definisi tugas menggunakan driver logging lancar yang terhubung ke port 24224 yang didengarkan oleh plug-in Fluent Bit. Agen Amazon ECS memberi label pada setiap wadah Amazon ECS dengan tag termasuk nama cluster, ARN tugas, nama keluarga definisi tugas, nomor revisi definisi tugas, dan nama wadah tugas. Label nilai kunci ini diteruskan ke Fluent Bit.

> **Note**
>
> Tugas ini menggunakan mode `default` jaringan. Namun, Anda juga dapat menggunakan mode `awsvpc` jaringan dengan tugas tersebut.

Untuk mendaftarkan ketentuan tugas

1. Buat file yang diberi nama `windows-app-task.json` dengan konten berikut ini.

```
{
  "family": "windows-app-task",
  "containerDefinitions": [
    {
      "name": "sample-container",
      "image": "mcr.microsoft.com/windows/servercore:ltsc2019",
      "cpu": 512,
      "memory": 512,
      "essential": true,
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "$count=1;while(1) { Write-Host $count; sleep 1; $count=$count+1;}"
      ],
      "logConfiguration": {
        "logDriver": "fluentd",
        "options": {
          "fluentd-address": "localhost:24224",
          "tag": "{{ index .ContainerLabels \"com.amazonaws.ecs.task-definition-
family\" }}",
```

```
            "fluentd-async": "true",
            "labels": "com.amazonaws.ecs.cluster,com.amazonaws.ecs.container-
name,com.amazonaws.ecs.task-arn,com.amazonaws.ecs.task-definition-
family,com.amazonaws.ecs.task-definition-version"
          }
        }
      }
    ],
    "memory": "512",
    "cpu": "512"
}
```

2. Jalankan perintah berikut untuk mendaftarkan definisi tugas.

   Ganti variabel berikut:

   - *region*dengan Wilayah tempat tugas Anda berjalan

```
aws ecs register-task-definition --cli-input-json file://windows-app-task.json --
region region
```

   Anda dapat membuat daftar definisi tugas untuk akun Anda dengan menjalankan `list-task-definitions` perintah. Output menampilkan nilai keluarga dan revisi yang dapat Anda gunakan bersama dengan `run-task` atau`start-task`.

Langkah 7: Jalankan definisi **`windows-app-task`** tugas

Setelah Anda mendaftarkan definisi `windows-app-task` tugas, jalankan di `FluentBit-cluster` cluster Anda.

Untuk menjalankan tugas

1. Jalankan definisi `windows-app-task:1` tugas yang Anda daftarkan di langkah sebelumnya.

   Ganti variabel berikut:

   - *region*dengan Wilayah tempat tugas Anda berjalan

```
aws ecs run-task --cluster FluentBit-cluster --task-definition windows-app-task:1
 --count 2 --region region
```

2. Jalankan perintah berikut untuk membuat daftar tugas Anda.

```
aws ecs list-tasks --cluster FluentBit-cluster
```

Langkah 8: Verifikasi log CloudWatch

Untuk memverifikasi penyiapan Fluent Bit Anda, periksa grup log berikut di CloudWatch konsol:

- `/ecs/fluent-bit-logs`- Ini adalah grup log yang sesuai dengan wadah daemon Fluent Bit yang berjalan pada instance container.
- `/aws/ecs/FluentBit-cluster.windows-app-task`- Ini adalah grup log yang sesuai dengan semua tugas yang diluncurkan untuk keluarga definisi `windows-app-task` tugas di dalam `FluentBit-cluster` cluster.

  `task-out.FIRST_TASK_ID.sample-container`- Aliran log ini berisi semua log yang dihasilkan oleh contoh pertama tugas dalam wadah tugas sample-container.

  `task-out.SECOND_TASK_ID.sample-container`- Aliran log ini berisi semua log yang dihasilkan oleh instance kedua tugas dalam wadah tugas sample-container.

Aliran `task-out.TASK_ID.sample-container` log memiliki bidang yang mirip dengan berikut ini:

```
{
    "source": "stdout",
    "ecs_task_arn": "arn:aws:ecs:region:0123456789012:task/FluentBit-
cluster/13EXAMPLE",
    "container_name": "/ecs-windows-app-task-1-sample-container-cEXAMPLE",
    "ecs_cluster": "FluentBit-cluster",
    "ecs_container_name": "sample-container",
    "ecs_task_definition_version": "1",
    "container_id": "61f5e6EXAMPLE",
    "log": "10",
    "ecs_task_definition_family": "windows-app-task"
}
```

Untuk memverifikasi pengaturan Fluent Bit

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, pilih Grup log. Pastikan Anda berada di Wilayah tempat Anda menerapkan Fluent Bit ke kontainer Anda.

   Dalam daftar grup log di Wilayah AWS, Anda akan melihat yang berikut:

   - `/ecs/fluent-bit-logs`
   - `/aws/ecs/FluentBit-cluster.windows-app-task`

   Jika Anda melihat grup log ini, pengaturan Fluent Bit diverifikasi.

Langkah 9: Membersihkan

Setelah Anda menyelesaikan tutorial ini, bersihkan sumber daya yang terkait dengannya untuk menghindari biaya yang tidak Anda gunakan.

Untuk membersihkan sumber daya tutorial

1. Hentikan `windows-simple-task` tugas dan `ecs-fluent-bit` tugas. Untuk informasi selengkapnya, lihat the section called “Menghentikan tugas”.
2. Jalankan perintah berikut untuk menghapus grup `/ecs/fluent-bit-logs` log. Untuk informasi selengkapnya, tentang menghapus grup log lihat delete-log-groupdi AWS Command Line Interface Referensi.

   ```
   aws logs delete-log-group --log-group-name /ecs/fluent-bit-logs
   aws logs delete-log-group --log-group-name /aws/ecs/FluentBit-cluster.windows-app-task
   ```

3. Jalankan perintah berikut untuk mengakhiri instance.

   ```
   aws ec2 terminate-instances --instance-ids instance-id
   ```

4. Jalankan perintah berikut untuk menghapus peran IAM.

   ```
   aws iam delete-role --role-name ecsInstanceRole
   aws iam delete-role --role-name fluentTaskRole
   ```

5. Jalankan perintah berikut untuk menghapus cluster Amazon ECS.

```
aws ecs delete-cluster --cluster FluentBit-cluster
```

Menggunakan gMSA untuk EC2 Linux wadah di Amazon ECS

Amazon ECS mendukung otentikasi Active Directory untuk container Linux EC2 melalui jenis akun layanan khusus yang disebut grup Managed Service Account ()gMSA.

LinuxAplikasi jaringan berbasis, seperti aplikasi .NET Core, dapat menggunakan Active Directory untuk memfasilitasi otentikasi dan manajemen otorisasi antara pengguna dan layanan. Anda dapat menggunakan fitur ini dengan merancang aplikasi yang terintegrasi dengan Active Directory dan berjalan di server yang bergabung dengan domain. Tetapi, karena Linux kontainer tidak dapat digabungkan dengan domain, Anda perlu mengonfigurasi Linux wadah untuk dijalankan. gMSA

LinuxContainer yang berjalan dengan gMSA mengandalkan `credentials-fetcher` daemon yang berjalan pada instance Amazon host container. EC2 Artinya, daemon mengambil gMSA kredensyal dari pengontrol domain Active Directory dan kemudian mentransfer kredensyal ini ke instance container. Untuk informasi selengkapnya tentang akun layanan, lihat Membuat gMSAs kontainer Windows di situs web Microsoft Learn.

Pertimbangan

Pertimbangkan hal berikut sebelum Anda gunakan gMSA untuk Linux wadah:

- Jika kontainer Anda menyala EC2, Anda dapat menggunakannya gMSA untuk Windows wadah dan Linux wadah. Untuk informasi tentang cara menggunakan gMSA untuk wadah Linux di Fargate, lihat. Menggunakan gMSA untuk Linux wadah di Fargate
- Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk menyelesaikan prasyarat. Misalnya, Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk membuat Active Directory denganPowerShell. gMSA PowerShell Alat Direktur RSAT Aktif hanya tersedia untukWindows. Untuk informasi selengkapnya, lihat Menginstal alat administrasi Direktori Aktif.
- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensil untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensional di salah satu opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	kredensial pengguna tanpa domain	kredensial pengguna tanpa domain
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk wadah Linux dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.

- Anda memiliki gMSA akun yang ada di Direktori Aktif. Untuk informasi selengkapnya, lihat Menggunakan gMSA untuk EC2 Linux wadah di Amazon ECS.
- Anda menginstal dan menjalankan `credentials-fetcher` daemon pada instance container Amazon ECS Linux. Anda juga menambahkan kumpulan kredensyal awal ke `credentials-fetcher` daemon untuk mengautentikasi dengan Active Directory.

  > **Note**
  >
  > `credentials-fetcher`Daemon hanya tersedia untuk Amazon Linux 2023 dan Fedora 37 dan yang lebih baru. Daemon tidak tersedia untuk Amazon Linux 2. Untuk informasi selengkapnya, lihat aws/credentials-fetcher di. GitHub

- Anda mengatur kredensyal untuk `credentials-fetcher` daemon untuk mengautentikasi dengan Active Directory. Kredensyal harus menjadi anggota grup keamanan Direktori Aktif yang memiliki akses ke akun. gMSA Ada beberapa opsi diPutuskan apakah Anda ingin menggabungkan instance ke domain, atau menggunakan gMSA domainless..
- Anda menambahkan izin IAM yang diperlukan. Izin yang diperlukan bergantung pada metode yang Anda pilih untuk kredensyal awal dan untuk menyimpan spesifikasi kredensialnya:
  - Jika Anda menggunakan domainless gMSA untuk kredensyal awal, izin IAM untuk AWS Secrets Manager diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensi di SSM Parameter Store, izin IAM untuk Amazon Systems EC2 Manager Parameter Store diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensyal di Amazon S3, izin IAM untuk Amazon Simple Storage Service diperlukan pada peran eksekusi tugas.

Menyiapkan Linux Kontainer gMSA berkemampuan di Amazon ECS

Siapkan infrastruktur

Langkah-langkah berikut adalah pertimbangan dan pengaturan yang dilakukan sekali. Setelah menyelesaikan langkah-langkah ini, Anda dapat mengotomatiskan pembuatan instance kontainer untuk menggunakan kembali konfigurasi ini.

Putuskan bagaimana kredensyal awal disediakan dan konfigurasikan data EC2 pengguna dalam templat EC2 peluncuran yang dapat digunakan kembali untuk menginstal daemon. `credentials-fetcher`

1. Putuskan apakah Anda ingin menggabungkan instance ke domain, atau menggunakan gMSA domainless.

   - Bergabunglah dengan EC2 instance ke domain Active Directory

     - Bergabunglah dengan instance berdasarkan data pengguna

       Tambahkan langkah-langkah untuk bergabung dengan domain Active Directory ke data EC2 pengguna Anda dalam template EC2 peluncuran. Beberapa grup Amazon EC2 Auto Scaling dapat menggunakan template peluncuran yang sama.

       Anda dapat menggunakan langkah-langkah ini Bergabung dengan Active Directory atau FreeIPA domain di Fedora Docs.

   - Jadikan pengguna Active Directory untuk domainless gMSA

     `credentials-fetcher`Daemon memiliki fitur yang disebut domainless. gMSA Fitur ini memerlukan domain, tetapi EC2 instance tidak perlu digabungkan ke domain. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensil untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama. Sebagai gantinya, Anda memberikan nama rahasia di AWS Secrets Manager dalam CredSpec file. Rahasianya harus berisi nama pengguna, kata sandi, dan domain untuk masuk.

     Fitur ini didukung dan dapat digunakan dengan wadah Linux dan Windows.

     Fitur ini mirip dengan gMSA support for non-domain-joined container hostsfitur. Untuk informasi selengkapnya tentang fitur Windows, lihat gMSAarsitektur dan peningkatan di situs web Microsoft Learn.

     a. Buat pengguna di domain Active Directory Anda. Pengguna di Active Directory harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.

     b. Buat rahasia di AWS Secrets Manager, setelah Anda membuat pengguna di Active Directory. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia.

     c. Masukkan nama pengguna, kata sandi, dan domain ke pasangan nilai kunci JSON yang dipanggil`username`, `password` dan`domainName`, masing-masing.

```
{"username":"username","password":"passw0rd", "domainName":"example.com"}
```

d. Tambahkan konfigurasi ke CredSpec file untuk akun layanan. Tambahan `HostAccountConfig` berisi Amazon Resource Name (ARN) dari rahasia di Secrets Manager.

   Pada Windows, `PluginGUID` harus cocok dengan GUID dalam contoh cuplikan berikut. Di Linux, `PluginGUID` itu diabaikan. Ganti `MySecret` dengan contoh dengan Amazon Resource Name (ARN) rahasia Anda.

```
    "ActiveDirectoryConfig": {
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
```

e. gMSAFitur tanpa domain memerlukan izin tambahan dalam peran eksekusi tugas. Ikuti langkahnya(Opsional) rahasia tanpa domain gMSA.

2. Konfigurasikan instance dan instal **`credentials-fetcher`** daemon

   Anda dapat menginstal `credentials-fetcher` daemon dengan skrip data pengguna di Template EC2 Peluncuran Anda. Contoh berikut menunjukkan dua jenis data pengguna, `cloud-config` YAML atau bash script. Contoh-contoh ini untuk Amazon Linux 2023 (AL2023). Ganti `MyCluster` dengan nama cluster Amazon ECS yang Anda inginkan untuk bergabung dengan instans ini.

   - **`cloud-config`** YAML

```
Content-Type: text/cloud-config
package_reboot_if_required: true
packages:
  # prerequisites
  - dotnet
  - realmd
  - oddjob
  - oddjob-mkhomedir
```

```
  - sssd
  - adcli
  - krb5-workstation
  - samba-common-tools
  # https://github.com/aws/credentials-fetcher gMSA credentials management for
 containers
  - credentials-fetcher
write_files:
# configure the ECS Agent to join your cluster.
# replace MyCluster with the name of your cluster.
- path: /etc/ecs/ecs.config
  owner: root:root
  permissions: '0644'
  content: |
    ECS_CLUSTER=MyCluster
    ECS_GMSA_SUPPORTED=true
runcmd:
# start the credentials-fetcher daemon and if it succeeded, make it start after
 every reboot
- "systemctl start credentials-fetcher"
- "systemctl is-active credentials-fetcher && systemctl enable credentials-
fetcher"
```

- bashnaskah

  Jika Anda lebih nyaman dengan bash skrip dan memiliki beberapa variabel untuk ditulis`/etc/ecs/ecs.config`, gunakan `heredoc` format berikut. Format ini menulis segalanya di antara baris dimulai dengan cat dan `EOF` pada file konfigurasi.

```
#!/usr/bin/env bash
set -euxo pipefail

# prerequisites
timeout 30 dnf install -y dotnet realmd oddjob oddjob-mkhomedir sssd adcli
 krb5-workstation samba-common-tools
# install https://github.com/aws/credentials-fetcher gMSA credentials
 management for containers
timeout 30 dnf install -y credentials-fetcher

# start credentials-fetcher
systemctl start credentials-fetcher
systemctl is-active credentials-fetcher && systemctl enable credentials-fetcher
```

```
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_GMSA_SUPPORTED=true
EOF
```

Ada variabel konfigurasi opsional untuk `credentials-fetcher` daemon yang dapat Anda atur. `/etc/ecs/ecs.config` Kami menyarankan Anda mengatur variabel dalam data pengguna di blok YAMM atau `heredoc` mirip dengan contoh sebelumnya. Melakukannya mencegah masalah dengan konfigurasi sebagian yang dapat terjadi dengan mengedit file beberapa kali. Untuk informasi selengkapnya tentang konfigurasi agen ECS, lihat Amazon ECS Container Agent on. GitHub

- Secara opsional, Anda dapat menggunakan variabel `CREDENTIALS_FETCHER_HOST` jika Anda mengubah konfigurasi `credentials-fetcher` daemon untuk memindahkan soket ke lokasi lain.

Menyiapkan izin dan rahasia

Lakukan langkah-langkah berikut sekali untuk setiap aplikasi dan setiap definisi tugas. Kami menyarankan Anda menggunakan praktik terbaik untuk memberikan hak istimewa paling sedikit dan mempersempit izin yang digunakan dalam kebijakan. Dengan cara ini, setiap tugas hanya dapat membaca rahasia yang dibutuhkannya.

1. (Opsional) rahasia tanpa domain gMSA

   Jika Anda menggunakan metode domainless di mana instance tidak bergabung dengan domain, ikuti langkah ini.

   Anda harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas. Melakukan hal itu memberikan akses `credentials-fetcher` daemon ke rahasia Secrets Manager. Ganti `MySecret` contoh dengan Amazon Resource Name (ARN) rahasia Anda dalam daftar. `Resource`

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                   "arn:aws:ssm:arn:aws: 111122223333:secret:MySecret"
            ]
        }
    ]
}
```

> **Note**
>
> Jika Anda menggunakan kunci KMS Anda sendiri untuk mengenkripsi rahasia Anda, Anda harus menambahkan izin yang diperlukan untuk peran ini dan menambahkan peran ini ke kebijakan kunci. AWS KMS

2. Putuskan apakah Anda menggunakan SSM Parameter Store atau S3 untuk menyimpan CredSpec

   Amazon ECS mendukung cara-cara berikut untuk mereferensikan jalur file di `credentialSpecs` bidang definisi tugas.

   Jika Anda menggabungkan instance ke satu domain, gunakan awalan `credentialspec:` di awal ARN dalam string. Jika Anda menggunakan domainlessgMSA, maka gunakan. `credentialspecdomainless:`

   Untuk informasi selengkapnya tentang CredSpec, lihat File spesifikasi kredensi.

   - Bucket Amazon S3

     Tambahkan spesifikasi kredenal ke bucket Amazon S3. Kemudian, rujuk Nama Sumber Daya Amazon (ARN) dari bucket Amazon S3 di bidang `credentialSpecs` definisi tugas.

     ```
     {
         "family": "",
         "executionRoleArn": "",
         "containerDefinitions": [
             {
                 "name": "",
     ```

```
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/
${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

Untuk memberikan akses tugas ke bucket S3, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/{object}"
            ]
        }
    ]
}
```

- Parameter SSM Menyimpan parameter

  Tambahkan spesifikasi kredenal ke parameter Penyimpanan Parameter SSM. Kemudian, rujuk Amazon Resource Name (ARN) dari parameter SSM Parameter Store di `credentialSpecs` bidang definisi tugas.

```
{
```

```
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:ssm:aws-
region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

Untuk memberikan akses tugas Anda ke parameter Penyimpanan Parameter SSM, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
            "arn:aws:ssm:arn:aws::111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

File spesifikasi kredensi

Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif () CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Linux Anda

menghasilkan CredSpec dan mereferensikannya di `credentialSpecs` bidang dalam definisi tugas Anda. CredSpecFile tidak mengandung rahasia apa pun.

Berikut ini adalah contoh CredSpec file.

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

MembuatCredSpec

Anda membuat CredSpec dengan menggunakan CredSpec PowerShell modul pada Windows komputer yang bergabung dengan domain. Ikuti langkah-langkah dalam Membuat spesifikasi kredenal di situs web Microsoft Belajar.

Menggunakan gMSA untuk Linux wadah di Fargate

Amazon ECS mendukung otentikasi Direktori Aktif untuk wadah Linux di Fargate melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (). gMSA

LinuxAplikasi jaringan berbasis, seperti aplikasi .NET Core, dapat menggunakan Active Directory untuk memfasilitasi otentikasi dan manajemen otorisasi antara pengguna dan layanan. Anda dapat menggunakan fitur ini dengan merancang aplikasi yang terintegrasi dengan Active Directory dan berjalan di server yang bergabung dengan domain. Tetapi, karena Linux kontainer tidak dapat digabungkan dengan domain, Anda perlu mengonfigurasi Linux wadah untuk dijalankan. gMSA

Pertimbangan

Pertimbangkan hal berikut sebelum Anda gunakan gMSA untuk Linux wadah di Fargate:

- Anda harus menjalankan Platform Versi 1.4 atau yang lebih baru.
- Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk menyelesaikan prasyarat. Misalnya, Anda mungkin memerlukan Windows komputer yang bergabung dengan domain untuk membuat Active Directory denganPowerShell. gMSA PowerShell Alat Direktur RSAT Aktif hanya tersedia untukWindows. Untuk informasi selengkapnya, lihat Menginstal alat administrasi Direktori Aktif.
- Anda harus menggunakan domainless gMSA.

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda membuat CredSpec file, dan kemudian menyimpannya di ember Amazon S3.
- Sebuah tugas hanya dapat mendukung satu Direktori Aktif.

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk wadah Linux dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.

  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki gMSA akun yang ada di Direktori Aktif dan pengguna yang memiliki izin untuk mengakses akun gMSA layanan. Untuk informasi selengkapnya, lihat Jadikan pengguna Active Directory untuk domainless gMSA.
- Anda memiliki ember Amazon S3. Lihat informasi yang lebih lengkap di Membuat bucket dalam Panduan Pengguna Amazon S3.

Menyiapkan Linux Kontainer gMSA berkemampuan di Amazon ECS

Siapkan infrastruktur

Langkah-langkah berikut adalah pertimbangan dan pengaturan yang dilakukan sekali.

- Jadikan pengguna Active Directory untuk domainless gMSA

  Saat Anda menggunakan domainlessgMSA, penampung tidak bergabung dengan domain. Aplikasi lain yang berjalan di wadah tidak dapat menggunakan kredensil untuk mengakses domain. Tugas yang menggunakan domain berbeda dapat berjalan pada wadah yang sama. Anda memberikan nama rahasia AWS Secrets Manager dalam CredSpec file. Rahasianya harus berisi nama pengguna, kata sandi, dan domain untuk masuk.

  Fitur ini mirip dengan gMSA support for non-domain-joined container hostsfitur. Untuk informasi selengkapnya tentang fitur Windows, lihat gMSAarsitektur dan peningkatan di situs web Microsoft Learn.

  a. Konfigurasikan pengguna di domain Active Directory Anda. Pengguna di Direktori Aktif harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.
  b. Anda memiliki VPC dan subnet yang dapat menyelesaikan nama domain Active Directory. Konfigurasikan VPC dengan opsi DHCP dengan nama domain yang menunjuk ke nama layanan Active Directory. Untuk informasi tentang cara mengonfigurasi opsi DHCP untuk VPC, lihat Set opsi Bekerja dengan DHCP di Panduan Pengguna Amazon Virtual Private Cloud.
  c. Buat rahasia di AWS Secrets Manager.
  d. Buat file spesifikasi kredensi.

Menyiapkan izin dan rahasia

Lakukan langkah-langkah berikut satu kali untuk setiap aplikasi dan setiap definisi tugas. Kami menyarankan Anda menggunakan praktik terbaik untuk memberikan hak istimewa paling sedikit dan mempersempit izin yang digunakan dalam kebijakan. Dengan cara ini, setiap tugas hanya dapat membaca rahasia yang dibutuhkannya.

1. Buat pengguna di domain Active Directory Anda. Pengguna di Active Directory harus memiliki izin untuk mengakses akun gMSA layanan yang Anda gunakan dalam tugas.
2. Setelah Anda membuat pengguna Active Directory, buat rahasia di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Membuat AWS Secrets Manager rahasia.
3. Masukkan nama pengguna, kata sandi, dan domain ke pasangan nilai kunci JSON yang dipanggil`username`, `password` dan`domainName`, masing-masing.

   ```
   {"username":"username","password":"passw0rd", "domainName":"example.com"}
   ```

4. Anda harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas. Melakukan hal itu memberikan akses `credentials-fetcher` daemon ke rahasia Secrets Manager. Ganti `MySecret` contoh dengan Amazon Resource Name (ARN) rahasia Anda dalam daftar. `Resource`

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "secretsmanager:GetSecretValue"
               ],
               "Resource": [
               "arn:aws:secretsmanager:arn:aws::111122223333:secret:MySecret"
               ]
           }
       ]
   }
   ```

> **Note**
>
> Jika Anda menggunakan kunci KMS Anda sendiri untuk mengenkripsi rahasia Anda, Anda harus menambahkan izin yang diperlukan untuk peran ini dan menambahkan peran ini ke kebijakan kunci. AWS KMS

5. Tambahkan spesifikasi kredenal ke bucket Amazon S3. Kemudian, rujuk Nama Sumber Daya Amazon (ARN) dari bucket Amazon S3 di bidang `credentialSpecs` definisi tugas.

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

Untuk memberikan akses tugas ke bucket S3, tambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListObjects"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

File spesifikasi kredensi

Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif () CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Linux Anda menghasilkan CredSpec dan mereferensikannya di `credentialSpecs` bidang dalam definisi tugas Anda. CredSpecFile tidak mengandung rahasia apa pun.

Berikut ini adalah contoh CredSpec file.

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
```

```
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

Membuat CredSpec dan mengunggahnya ke Amazon S3

Anda membuat CredSpec dengan menggunakan CredSpec PowerShell modul pada Windows komputer yang bergabung dengan domain. Ikuti langkah-langkah dalam Membuat spesifikasi kredenal di situs web Microsoft Belajar.

Setelah Anda membuat file spesifikasi kredensi, unggah ke bucket Amazon S3. Salin CredSpec file ke komputer atau lingkungan tempat Anda menjalankan AWS CLI perintah.

Jalankan AWS CLI perintah berikut untuk mengunggah CredSpec ke Amazon S3. Ganti *amzn-s3-demo-bucket* dengan nama bucket Amazon S3 Anda. Anda dapat menyimpan file sebagai objek di bucket dan lokasi mana pun, tetapi Anda harus mengizinkan akses ke bucket dan lokasi tersebut dalam kebijakan yang Anda lampirkan ke peran eksekusi tugas.

Untuk PowerShell, gunakan perintah berikut:

```
$ Write-S3Object -BucketName "amzn-s3-demo-bucket" -Key "ecs-domainless-gmsa-credspec"
 -File "gmsa-cred-spec.json"
```

AWS CLI Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel.

```
$ aws s3 cp gmsa-cred-spec.json \
s3://amzn-s3-demo-bucket/ecs-domainless-gmsa-credspec
```

Menggunakan Amazon ECS Windows container dengan domainless gMSA menggunakan AWS CLI

Tutorial berikut menunjukkan cara membuat tugas Amazon ECS yang menjalankan wadah Windows yang memiliki kredensyal untuk mengakses Active Directory dengan file. AWS CLI Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain

pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

Topik

- Prasyarat
- Langkah 1: Buat dan konfigurasikan gMSA akun di Layanan Domain Direktori Aktif (AD DS)
- Langkah 2: Unggah Kredensial ke Secrets Manager
- Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA
- Langkah 4: Unggah CredSpec ke Amazon S3
- Langkah 5: (Opsional) Buat cluster Amazon ECS
- Langkah 6: Buat peran IAM untuk instance kontainer
- Langkah 7: Buat peran eksekusi tugas khusus
- Langkah 8: Buat peran tugas untuk Amazon ECS Exec
- Langkah 9: Daftarkan definisi tugas yang menggunakan domainless gMSA
- Langkah 10: Daftarkan instance wadah Windows ke cluster
- Langkah 11: Verifikasi contoh kontainer
- Langkah 12: Jalankan tugas Windows
- Langkah 13: Verifikasi wadah memiliki gMSA kredensil
- Langkah 14: Bersihkan
- Debugging Amazon ECS tanpa domain gMSA untuk wadah Windows

Prasyarat

Jika mengikuti tutorial ini, berarti prasyarat berikut telah selesai:

- Langkah-langkah di Siapkan untuk menggunakan Amazon ECS telah selesai.
- Pengguna IAM Anda memiliki izin yang diperlukan yang ditentukan dalam contoh kebijakan Amazonecs_ FullAccess IAM.
- Versi terbaru diinstal dan dikonfigurasi. AWS CLI Untuk informasi selengkapnya tentang menginstal atau memutakhirkan AWS CLI, lihat Menginstal. AWS Command Line Interface

> **Note**
>
> Anda dapat menggunakan titik akhir layanan dual-stack untuk berinteraksi dengan Amazon ECS dari AWS CLI, SDKs dan Amazon ECS API melalui keduanya dan. IPv4 IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:
  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki VPC dan subnet yang dapat menyelesaikan nama domain Active Directory.
- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

  Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensional di salah satu opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec

Lokasi penyimpanan	Linux	Windows
AWS Secrets Manager	kredensial pengguna tanpa domain	kredensial pengguna tanpa domain
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

- (Opsional) AWS CloudShell adalah alat yang memberi pelanggan baris perintah tanpa perlu membuat EC2 instance mereka sendiri. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudShell? dalam AWS CloudShell User Guide.

Langkah 1: Buat dan konfigurasikan gMSA akun di Layanan Domain Direktori Aktif (AD DS)

Buat dan konfigurasikan gMSA akun di domain Active Directory.

> **Note**
>
> Langkah ini membuat dua akun terpisah: Akun Layanan Terkelola Grup (GMSA) yang menyediakan identitas untuk kontainer Anda, dan akun pengguna biasa yang digunakan untuk otentikasi domain. Akun ini melayani tujuan yang berbeda dan harus memiliki nama yang berbeda.

1. Menghasilkan kunci root Layanan Distribusi Kunci

   > **Note**
   >
   > Jika Anda menggunakan AWS Directory Service, maka Anda dapat melewati langkah ini.

   Kunci root dan gMSA izin KDS dikonfigurasi dengan AWS Microsoft AD terkelola Anda.

   Jika Anda belum membuat Akun gMSA Layanan di domain Anda, Anda harus terlebih dahulu membuat kunci root Key Distribution Service (KDS). KDS bertanggung jawab untuk membuat,

memutar, dan melepaskan gMSA kata sandi ke host resmi. Ketika `ccg.exe` kebutuhan untuk mengambil gMSA kredensyal, itu menghubungi KDS untuk mengambil kata sandi saat ini.

Untuk memeriksa apakah kunci root KDS telah dibuat, jalankan PowerShell cmdlet berikut dengan hak admin domain pada pengontrol domain menggunakan modul. `ActiveDirectory` PowerShell Untuk informasi selengkapnya tentang modul, lihat ActiveDirectory Modul di situs web Microsoft Learn.

```
PS C:\> Get-KdsRootKey
```

Jika perintah mengembalikan ID kunci, Anda dapat melewati sisa langkah ini. Jika tidak, buat kunci root KDS dengan menjalankan perintah berikut:

```
PS C:\> Add-KdsRootKey -EffectiveImmediately
```

Meskipun argumen `EffectiveImmediately` untuk perintah menyiratkan kunci efektif segera, Anda harus menunggu 10 jam sebelum kunci root KDS direplikasi dan tersedia untuk digunakan pada semua pengontrol domain.

2. Buat gMSA akun

   Untuk membuat gMSA akun dan memungkinkan `ccg.exe` untuk mengambil gMSA kata sandi, jalankan PowerShell perintah berikut dari Windows Server atau klien dengan akses ke domain. Ganti `ExampleAccount` dengan nama yang Anda inginkan untuk gMSA akun Anda, dan ganti `example-domain` dengan nama domain Active Directory Anda (misalnya, jika domain Anda`contoso.com`, gunakan`contoso`).

   a.
   ```
   PS C:\> Install-WindowsFeature RSAT-AD-PowerShell
   ```

   b.
   ```
   PS C:\> New-ADGroup -Name "ExampleAccount Authorized Hosts" -SamAccountName
    "ExampleAccountHosts" -GroupScope DomainLocal
   ```

   c.
   ```
   PS C:\> New-ADServiceAccount -Name "ExampleAccount" -DnsHostName "example-
   domain" -ServicePrincipalNames "host/ExampleAccount", "host/example-domain" -
   PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
   ```

   d. Buat pengguna dengan kata sandi permanen yang tidak kedaluwarsa. Kredensil ini disimpan AWS Secrets Manager dan digunakan oleh setiap tugas untuk bergabung dengan domain. Ini adalah akun pengguna terpisah dari akun GMSA yang dibuat di atas. Ganti

`ExampleServiceUser` dengan nama yang Anda inginkan untuk akun pengguna layanan ini.

```
PS C:\> New-ADUser -Name "ExampleServiceUser" -AccountPassword (ConvertTo-
SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
```

e.
```
PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members
 "ExampleServiceUser"
```

f. Instal PowerShell modul untuk membuat CredSpec objek di Active Directory dan output CredSpec JSON.

```
PS C:\> Install-PackageProvider -Name NuGet -Force
```

```
PS C:\> Install-Module CredentialSpec
```

g.
```
PS C:\> New-CredentialSpec -AccountName ExampleAccount
```

3. Salin output JSON dari perintah sebelumnya ke dalam file bernama`gmsa-cred-spec.json`. Ini adalah CredSpec file. Ini digunakan pada Langkah 3,Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA.

Langkah 2: Unggah Kredensial ke Secrets Manager

Salin kredensyal Active Directory ke dalam sistem penyimpanan kredensyal yang aman, sehingga setiap tugas mengambilnya. Ini adalah metode domainlessgMSA. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

- Jalankan AWS CLI perintah berikut dan ganti nama pengguna, kata sandi, dan nama domain agar sesuai dengan lingkungan Anda. Gunakan nama akun pengguna layanan (bukan nama akun GMSA) untuk nama pengguna. Simpan ARN rahasia untuk digunakan pada langkah berikutnya, Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA

Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleServiceUser\",\"password\":\"Test123\",
\"domainName\":\"contoso.com\"}"
```

Langkah 3: Ubah CredSpec JSON Anda untuk menyertakan informasi tanpa domain gMSA

Sebelum mengunggah CredSpec ke salah satu opsi penyimpanan, tambahkan informasi ke CredSpec dengan ARN rahasia di Secrets Manager dari langkah sebelumnya. Untuk informasi selengkapnya, lihat Konfigurasi spesifikasi kredensyal tambahan untuk kasus penggunaan host non-domain-joined kontainer di situs web Microsoft Learn.

1. Tambahkan informasi berikut ke CredSpec file di dalam file`ActiveDirectoryConfig`. Ganti ARN dengan rahasia di Secrets Manager dari langkah sebelumnya.

   Perhatikan bahwa `PluginGUID` nilai harus cocok dengan GUID dalam contoh cuplikan berikut dan diperlukan.

   ```
   "HostAccountConfig": {
         "PortableCcgVersion": "1",
         "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
         "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
   region:111122223333:secret:gmsa-plugin-input\"}"
       }
   ```

   Anda juga dapat menggunakan rahasia di SSM Parameter Store dengan menggunakan ARN dalam format ini: `\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`

2. Setelah Anda memodifikasi CredSpec file, itu akan terlihat seperti contoh berikut:

   ```
   {
   ```

```
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "ExampleAccount",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
    "DnsTreeName": "example-domain",
    "DnsName": "example-domain",
    "NetBiosName": "example-domain"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "ExampleAccount",
        "Scope": "example-domain"
      },
      {
        "Name": "ExampleAccount",
        "Scope": "example-domain"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

Langkah 4: Unggah CredSpec ke Amazon S3

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Salin CredSpec file ke komputer atau lingkungan tempat Anda menjalankan AWS CLI perintah.
2. Jalankan AWS CLI perintah berikut untuk mengunggah CredSpec ke Amazon S3. Ganti `MyBucket` dengan nama bucket Amazon S3 Anda. Anda dapat menyimpan file sebagai objek di

bucket dan lokasi mana pun, tetapi Anda harus mengizinkan akses ke bucket dan lokasi tersebut dalam kebijakan yang Anda lampirkan ke peran eksekusi tugas.

Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws s3 cp gmsa-cred-spec.json \
s3://MyBucket/ecs-domainless-gmsa-credspec
```

Langkah 5: (Opsional) Buat cluster Amazon ECS

Secara default, akun Anda memiliki klaster Amazon ECS bernama`default`. Cluster ini digunakan secara default di AWS CLI, SDKs, dan AWS CloudFormation. Anda dapat menggunakan klaster tambahan untuk mengelompokkan dan mengatur tugas dan infrastruktur, dan menetapkan default untuk beberapa konfigurasi.

Anda dapat membuat cluster dari AWS Management Console, AWS CLI, SDKs, atau AWS CloudFormation. Pengaturan dan konfigurasi di cluster tidak mempengaruhigMSA.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

```
$ aws ecs create-cluster --cluster-name windows-domainless-gmsa-cluster
```

> **Important**
>
> Jika Anda memilih untuk membuat cluster Anda sendiri, Anda harus menentukan `--cluster clusterName` untuk setiap perintah yang ingin Anda gunakan dengan cluster itu.

Langkah 6: Buat peran IAM untuk instance kontainer

Instance container adalah komputer host untuk menjalankan container dalam tugas ECS, misalnya EC2 instans Amazon. Setiap instans kontainer mendaftar ke klaster Amazon ECS. Sebelum meluncurkan EC2 instans Amazon dan mendaftarkannya ke klaster, Anda harus membuat peran IAM agar instans penampung dapat digunakan.

Untuk membuat peran instance container, lihatPeran IAM instance wadah Amazon ECS. Default `ecsInstanceRole` memiliki izin yang cukup untuk menyelesaikan tutorial ini.

Langkah 7: Buat peran eksekusi tugas khusus

Amazon ECS dapat menggunakan peran IAM yang berbeda untuk izin yang diperlukan untuk memulai setiap tugas, bukan peran instance container. Peran ini adalah peran eksekusi tugas. Sebaiknya buat peran eksekusi tugas hanya dengan izin yang diperlukan untuk ECS untuk menjalankan tugas, juga dikenal sebagai izin hak istimewa paling sedikit. Untuk informasi lebih lanjut tentang prinsip hak istimewa terkecil, lihat SEC03-BP02 Memberikan akses hak istimewa terkecil dalam Kerangka Kerja Well-Architected.AWS

1. Untuk membuat peran eksekusi tugas, lihatMembuat peran eksekusi tugas . Izin default memungkinkan instance container untuk menarik gambar kontainer dari Amazon Elastic Container Registry `stdout` dan dan `stderr` dari aplikasi Anda untuk dicatat ke Amazon CloudWatch Logs.

   Karena peran membutuhkan izin khusus untuk tutorial ini, Anda dapat memberikan peran nama yang berbeda dari`ecsTaskExecutionRole`. Tutorial ini digunakan `ecsTaskExecutionRole` dalam langkah-langkah selanjutnya.
2. Tambahkan izin berikut dengan membuat kebijakan khusus, baik kebijakan sebaris yang hanya ada untuk peran ini, atau kebijakan yang dapat Anda gunakan kembali. Ganti ARN untuk pernyataan pertama dengan bucket dan lokasi Amazon S3, dan yang kedua `Resource` dengan ARN rahasia di Secrets Manager. `Resource`

   Jika Anda mengenkripsi rahasia di Secrets Manager dengan kunci khusus, Anda juga harus mengizinkan `kms:Decrypt` kunci tersebut.

   Jika Anda menggunakan SSM Parameter Store bukan Secrets Manager, Anda harus mengizinkan `ssm:GetParameter` parameter, bukan. `secretsmanager:GetSecretValue`

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::MyBucket/ecs-domainless-gmsa-credspec/
gmsa-cred-spec.json"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource":
 "arn:aws:secretsmanager:arn:aws:: 111122223333:secret:gmsa-plugin-AbCdEf"
        }
    ]
}
```

Langkah 8: Buat peran tugas untuk Amazon ECS Exec

Tutorial ini menggunakan Amazon ECS Exec untuk memverifikasi fungsionalitas dengan menjalankan perintah di dalam tugas yang sedang berjalan. Untuk menggunakan ECS Exec, layanan atau tugas harus mengaktifkan ECS Exec dan peran tugas (tetapi bukan peran eksekusi tugas) harus memiliki izin. `ssmmessages` Untuk kebijakan IAM yang diperlukan, lihatIzin ECS Exec.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

Untuk membuat peran tugas menggunakan AWS CLI, ikuti langkah-langkah ini.

1. Buat file yang disebut `ecs-tasks-trust-policy.json` dengan konten berikut:

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

2. Buat peran IAM. Anda dapat mengganti nama `ecs-exec-demo-task-role` tetapi tetap nama untuk langkah-langkah berikut.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws iam create-role --role-name ecs-exec-demo-task-role \
--assume-role-policy-document file://ecs-tasks-trust-policy.json
```

   Anda dapat menghapus file tersebut`ecs-tasks-trust-policy.json`.

3. Buat file yang disebut `ecs-exec-demo-task-role-policy.json` dengan konten berikut:

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Buat kebijakan IAM dan lampirkan ke peran dari langkah sebelumnya.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws iam put-role-policy \
    --role-name ecs-exec-demo-task-role \
    --policy-name ecs-exec-demo-task-role-policy \
    --policy-document file://ecs-exec-demo-task-role-policy.json
```

Anda dapat menghapus file tersebut`ecs-exec-demo-task-role-policy.json`.

Langkah 9: Daftarkan definisi tugas yang menggunakan domainless gMSA

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Buat file yang disebut `windows-gmsa-domainless-task-def.json` dengan konten berikut:

```
{
  "family": "windows-gmsa-domainless-task",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "memory": 1024,
      "essential": true,
      "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::ecs-domainless-gmsa-
credspec/gmsa-cred-spec.json"
      ],
      "entryPoint": [
        "powershell",
        "-Command"
      ],
      "command": [
        "New-Item -Path C:\\inetpub\\wwwroot\\index.html -ItemType file -Value
 '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>' -Force ; C:\
\ServiceMonitor.exe w3svc"
      ],
      "portMappings": [
```

```
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ]
    }
  ],
  "taskRoleArn": "arn:aws:iam::111122223333:role/ecs-exec-demo-task-role",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

2. Daftarkan definisi tugas dengan menjalankan perintah berikut:

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh sh dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws ecs register-task-definition \
--cli-input-json file://windows-gmsa-domainless-task-def.json
```

Langkah 10: Daftarkan instance wadah Windows ke cluster

Luncurkan instance Amazon EC2 Windows dan jalankan agen kontainer ECS untuk mendaftarkannya sebagai instance container di cluster. ECS menjalankan tugas pada instance kontainer yang terdaftar ke cluster tempat tugas dimulai.

1. Untuk meluncurkan instans Amazon EC2 Windows yang dikonfigurasi untuk Amazon ECS di AWS Management Console, lihatMeluncurkan instans penampung Amazon ECS Windows. Berhenti di langkah untuk data pengguna.
2. UntukgMSA, data pengguna harus mengatur variabel lingkungan `ECS_GMSA_SUPPORTED` sebelum memulai agen kontainer ECS.

   Untuk ECS Exec, agen harus memulai dengan argumen. `-EnableTaskIAMRole`

   Untuk mengamankan peran IAM instance dengan mencegah tugas mencapai layanan web EC2 IMDS untuk mengambil kredensyal peran, tambahkan argumen. `-AwsvpcBlockIMDS` Ini hanya berlaku untuk tugas yang menggunakan mode `awsvpc` jaringan.

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_GMSA_SUPPORTED", $TRUE, "Machine")
Import-Module ECSTools
Initialize-ECSAgent -Cluster windows-domainless-gmsa-cluster -EnableTaskIAMRole -
AwsvpcBlockIMDS
</powershell>
```

3. Tinjau ringkasan konfigurasi instans di panel Ringkasan, dan ketika Anda siap, pilih Luncurkan instans.

Langkah 11: Verifikasi contoh kontainer

Anda dapat memverifikasi bahwa ada instance kontainer di cluster menggunakan file AWS Management Console. Namun, gMSA perlu fitur tambahan yang diindikasikan sebagai atribut. Atribut ini tidak terlihat di AWS Management Console, jadi tutorial ini menggunakan AWS CLI.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Buat daftar instance kontainer di cluster. Instance kontainer memiliki ID yang berbeda dari ID EC2 instance.

   ```
   $ aws ecs list-container-instances
   ```

   Output:

   ```
   {
       "containerInstanceArns": [
           "arn:aws:ecs:aws-region:111122223333:container-
   instance/default/MyContainerInstanceID"
       ]
   }
   ```

   Misalnya, `526bd5d0ced448a788768334e79010fd` adalah ID instance kontainer yang valid.

2. Gunakan ID instance container dari langkah sebelumnya untuk mendapatkan detail untuk instance container. Ganti `MyContainerInstanceID` dengan ID.

Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws ecs describe-container-instances \
     ----container-instances MyContainerInstanceID
```

Perhatikan bahwa outputnya sangat panjang.

3. Verifikasi bahwa `attributes` daftar memiliki objek dengan kunci yang dipanggil `name` dan nilai`ecs.capability.gmsa-domainless`. Berikut ini adalah contoh objek.

   Output:

```
{
    "name": "ecs.capability.gmsa-domainless"
}
```

Langkah 12: Jalankan tugas Windows

Jalankan tugas Amazon ECS. Jika hanya ada 1 instance kontainer di cluster, Anda dapat menggunakan`run-task`. Jika ada banyak instance kontainer yang berbeda, mungkin lebih mudah untuk menggunakan `start-task` dan menentukan ID instance container untuk menjalankan tugas, daripada menambahkan batasan penempatan ke definisi tugas untuk mengontrol jenis instance kontainer untuk menjalankan tugas ini.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
aws ecs run-task --task-definition windows-gmsa-domainless-task \
    --enable-execute-command --cluster windows-domainless-gmsa-cluster
```

   Perhatikan ID tugas yang dikembalikan oleh perintah.

2. Jalankan perintah berikut untuk memverifikasi bahwa tugas telah dimulai. Perintah ini menunggu dan tidak mengembalikan shell prompt sampai tugas dimulai. Ganti `MyTaskID` dengan ID tugas dari langkah sebelumnya.

```
$ aws ecs wait tasks-running --task MyTaskID
```

Langkah 13: Verifikasi wadah memiliki gMSA kredensil

Verifikasi bahwa wadah dalam tugas memiliki Kerberos token. gMSA

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws ecs execute-command \
--task MyTaskID \
--container windows_sample_app \
--interactive \
--command powershell.exe
```

   Outputnya akan menjadi PowerShell prompt.
2. Jalankan perintah berikut di PowerShell terminal di dalam wadah.

```
PS C:\> klist get ExampleAccount$
```

   Dalam output, perhatikan `Principal` adalah salah satu yang Anda buat sebelumnya.

Langkah 14: Bersihkan

Setelah selesai dengan tutorial ini, Anda harus membersihkan sumber daya yang terkait untuk menghindari timbulnya biaya untuk sumber daya yang tidak terpakai.

Langkah ini menggunakan AWS CLI. Anda dapat menjalankan perintah ini AWS CloudShell di shell default, yaitu`bash`.

1. Hentikan tugas. Ganti `MyTaskID` dengan ID tugas dari langkah 12,Langkah 12: Jalankan tugas Windows.

```
$ aws ecs stop-task --task MyTaskID
```

2. Hentikan EC2 instans Amazon. Setelah itu, instance container di cluster akan dihapus secara otomatis setelah satu jam.

   Anda dapat menemukan dan menghentikan instance dengan menggunakan EC2 konsol Amazon. Atau, Anda dapat menjalankan perintah berikut. Untuk menjalankan perintah, cari ID EC2 instance dalam output `aws ecs describe-container-instances` perintah dari langkah 1,Langkah 11: Verifikasi contoh kontainer. i-10a64379 adalah contoh ID instance. EC2

```
$ aws ec2 terminate-instances --instance-ids MyInstanceID
```

3. Hapus CredSpec file di Amazon S3. Ganti `MyBucket` dengan nama bucket Amazon S3 Anda.

```
$ aws s3api delete-object --bucket MyBucket --key ecs-domainless-gmsa-credspec/
gmsa-cred-spec.json
```

4. Hapus rahasia dari Secrets Manager. Jika Anda menggunakan SSM Parameter Store sebagai gantinya, hapus parameternya.

   Perintah berikut menggunakan karakter kelanjutan garis miring terbalik yang digunakan oleh `sh` dan shell yang kompatibel. Perintah ini tidak kompatibel dengan PowerShell. Anda harus memodifikasi perintah untuk menggunakannya PowerShell.

```
$ aws secretsmanager delete-secret --secret-id gmsa-plugin-input \
    --force-delete-without-recovery
```

5. Deregister dan hapus definisi tugas. Dengan membatalkan pendaftaran definisi tugas, Anda menandainya sebagai tidak aktif sehingga tidak dapat digunakan untuk memulai tugas baru. Kemudian, Anda dapat menghapus definisi tugas.

   a. Deregister definisi tugas dengan menentukan versinya. ECS secara otomatis membuat versi definisi tugas, yang diberi nomor mulai dari 1. Anda merujuk ke versi dalam format yang sama dengan label pada gambar kontainer, seperti`:1`.

```
$ aws ecs deregister-task-definition --task-definition windows-gmsa-domainless-
task:1
```

b. Hapus definisi tugas.

```
$ aws ecs delete-task-definitions --task-definition windows-gmsa-domainless-task:1
```

6. (Opsional) Hapus cluster ECS, jika Anda membuat cluster.

```
$ aws ecs delete-cluster --cluster windows-domainless-gmsa-cluster
```

Debugging Amazon ECS tanpa domain gMSA untuk wadah Windows

Status tugas Amazon ECS

ECS mencoba memulai tugas tepat sekali. Setiap tugas yang memiliki masalah dihentikan, dan disetel ke status`STOPPED`. Ada dua jenis masalah umum dengan tugas. Pertama, tugas yang tidak bisa dimulai. Kedua, tugas di mana aplikasi telah berhenti di dalam salah satu wadah. Di AWS Management Console, lihat bidang Alasan berhenti dari tugas karena alasan mengapa tugas itu dihentikan. Dalam AWS CLI, jelaskan tugas dan lihat`stoppedReason`. Untuk langkah-langkah di AWS Management Console dan AWS CLI, lihatMelihat Amazon ECS menghentikan kesalahan tugas.

Peristiwa-peristiwa Windows

Peristiwa Windows untuk gMSA dalam wadah dicatat dalam file `Microsoft-Windows-Containers-CCG` log dan dapat ditemukan di Penampil Acara di bagian Aplikasi dan Layanan di`Logs\Microsoft\Windows\Containers-CCG\Admin`. Untuk tips debugging lainnya, lihat Memecahkan masalah g MSAs untuk kontainer Windows di situs web Microsoft Learn.

Plugin agen gMSA ECS

Logging untuk gMSA plugin untuk agen ECS pada instance kontainer Windows ada di direktori berikut,`C:/ProgramData/Amazon/gmsa-plugin/`. Lihat di log ini untuk melihat apakah kredensil pengguna tanpa domain diunduh dari lokasi penyimpanan, seperti Secrets Manager, dan format kredensialnya telah dibaca dengan benar.

Pelajari cara menggunakan g MSAs untuk wadah EC2 Windows untuk Amazon ECS

Amazon ECS mendukung otentikasi Active Directory untuk wadah Windows melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (GMSA).

Aplikasi jaringan berbasis Windows seperti aplikasi NET sering menggunakan Direktori Aktif untuk memfasilitasi autentikasi dan otorisasi manajemen antara pengguna dan layanan. Developer umumnya merancang aplikasi mereka untuk mengintegrasikan dengan Direktori Aktif dan berjalan pada server yang bergabung dengan domain untuk tujuan ini. Karena kontainer Windows tidak dapat bergabung dengan domain, Anda harus mengonfigurasi kontainer Windows untuk berjalan dengan gMSA.

Wadah Windows yang berjalan dengan GMSA bergantung pada instans EC2 Amazon hostnya untuk mengambil kredenal GMSA dari pengontrol domain Active Directory dan menyediakannya ke instance container. Untuk informasi selengkapnya, lihat Membuat g MSAs untuk kontainer Windows.

> **Note**
>
> Fitur ini tidak didukung pada wadah Windows di Fargate.

Topik

- Pertimbangan
- Prasyarat
- Menyiapkan gMSA untuk Windows Container di Amazon ECS

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan g MSAs untuk wadah Windows:

- Saat menggunakan AMI Penuh Windows Server 2016 yang dioptimalkan Amazon ECS untuk instance container Anda, nama host container harus sama dengan nama akun GMSA yang ditentukan dalam file spesifikasi kredenal. Untuk menentukan hostname untuk kontainer, gunakan parameter ketentuan kontainer `hostname`. Untuk informasi selengkapnya, lihat Pengaturan jaringan.

- Anda memilih antara domainless gMSA dan menggabungkan setiap instance ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

  Kemudian, pilih penyimpanan data untuk CredSpec dan opsional, untuk kredensyal pengguna Active Directory untuk domainless. gMSA

  Amazon ECS menggunakan file spesifikasi kredensi Direktori Aktif ()CredSpec. File ini berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah. Anda menghasilkan CredSpec file dan kemudian menyimpannya di salah satu opsi CredSpec penyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer. Untuk menggunakan metode domainless, bagian opsional dalam CredSpec file dapat menentukan kredensional di salah satu opsi domainless user credentialspenyimpanan dalam tabel berikut, khusus untuk Sistem Operasi instance kontainer.

Lokasi penyimpanan	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	kredensil pengguna tanpa domain	kredensil pengguna tanpa domain
Toko Parameter Amazon EC2 Systems Manager	CredSpec	CredSpec, kredensyal pengguna tanpa domain
Berkas lokal	N/A	CredSpec

Prasyarat

Sebelum Anda menggunakan fitur gMSA untuk Windows container dengan Amazon ECS, pastikan untuk menyelesaikan yang berikut ini:

- Anda menyiapkan domain Direktori Aktif dengan sumber daya yang ingin diakses oleh kontainer Anda. Amazon ECS mendukung pengaturan berikut:

  - Direktori AWS Directory Service Aktif. AWS Directory Service adalah Direktori Aktif AWS terkelola yang di-host di Amazon EC2. Untuk informasi selengkapnya, lihat Memulai dengan Microsoft AD yang AWS Dikelola di Panduan AWS Directory Service Administrasi.
  - Direktori Aktif lokal. Anda harus memastikan bahwa instans penampung Amazon ECS Linux dapat bergabung dengan domain. Untuk informasi selengkapnya, lihat AWS Direct Connect.
- Anda memiliki gMSA akun yang ada di Direktori Aktif. Untuk informasi selengkapnya, lihat Membuat g MSAs untuk kontainer Windows.
- Anda memilih untuk menggunakan domainless gMSA atau instans penampung Amazon ECS Windows yang menghosting tugas Amazon ECS harus domain yang digabungkan ke Direktori Aktif dan menjadi anggota grup keamanan Direktori Aktif yang memiliki akses ke akun GMSA.

  Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.
- Anda menambahkan izin IAM yang diperlukan. Izin yang diperlukan bergantung pada metode yang Anda pilih untuk kredensyal awal dan untuk menyimpan spesifikasi kredensialnya:
  - Jika Anda menggunakan domainless gMSA untuk kredensyal awal, izin IAM untuk AWS Secrets Manager diperlukan pada peran instans Amazon. EC2
  - Jika Anda menyimpan spesifikasi kredensi di SSM Parameter Store, izin IAM untuk Amazon Systems EC2 Manager Parameter Store diperlukan pada peran eksekusi tugas.
  - Jika Anda menyimpan spesifikasi kredensyal di Amazon S3, izin IAM untuk Amazon Simple Storage Service diperlukan pada peran eksekusi tugas.

Menyiapkan gMSA untuk Windows Container di Amazon ECS

Untuk mengatur gMSA untuk Windows Containers di Amazon ECS, Anda dapat mengikuti tutorial lengkap yang mencakup konfigurasi prasyarat. Menggunakan Amazon ECS Windows container dengan domainless gMSA menggunakan AWS CLI

Bagian berikut mencakup CredSpec konfigurasi secara rinci.

Topik

- Contoh CredSpec
- Pengaturan tanpa domain gMSA
- Referensi File Spek Kredensial dalam Definisi Tugas

Contoh CredSpec

Amazon ECS menggunakan file spesifikasi kredensyal yang berisi gMSA metadata yang digunakan untuk menyebarkan konteks gMSA akun ke wadah Windows. Anda dapat menghasilkan file spesifikasi kredensial dan referensi dalam bidang `credentialSpec` pada ketentuan tugas Anda. File spek kredensial tidak berisi rahasia apa pun.

Berikut ini contoh file spek kredensial:

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "contoso.com",
        "DnsName": "contoso.com",
        "NetBiosName": "contoso"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "contoso.com"
            }
        ]
    }
}
```

Pengaturan tanpa domain gMSA

Kami merekomendasikan domainless gMSA daripada menggabungkan instance container ke satu domain. Dengan menggunakan domainlessgMSA, instance container tidak bergabung ke domain, aplikasi lain pada instance tidak dapat menggunakan kredensi untuk mengakses domain, dan tugas yang menggabungkan domain yang berbeda dapat berjalan pada instance yang sama.

1. Sebelum mengunggah CredSpec ke salah satu opsi penyimpanan, tambahkan informasi ke ARN rahasia di Secrets Manager atau SSM Parameter Store. CredSpec Untuk informasi

selengkapnya, lihat Konfigurasi spesifikasi kredensyal tambahan untuk kasus penggunaan host non-domain-joined kontainer di situs web Microsoft Learn.

Format kredensi tanpa domain gMSA

Berikut ini adalah format JSON untuk gMSA kredensyal domainless untuk Active Directory Anda. Simpan kredensialnya di Secrets Manager atau SSM Parameter Store.

```
{
    "username":"WebApp01",
    "password":"Test123!",
    "domainName":"contoso.com"
}
```

2. Tambahkan informasi berikut ke CredSpec file di dalam file`ActiveDirectoryConfig`. Ganti ARN dengan rahasia di Secrets Manager atau SSM Parameter Store.

   Perhatikan bahwa `PluginGUID` nilai harus cocok dengan GUID dalam contoh cuplikan berikut dan diperlukan.

```
    "HostAccountConfig": {
        "PortableCcgVersion": "1",
        "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
        "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
```

   Anda juga dapat menggunakan rahasia di SSM Parameter Store dengan menggunakan ARN dalam format ini: `\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`

3. Setelah Anda memodifikasi CredSpec file, itu akan terlihat seperti contoh berikut:

```
{
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "WebApp01",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
```

```
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      },
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

Referensi File Spek Kredensial dalam Definisi Tugas

Amazon ECS mendukung cara-cara berikut untuk mereferensikan jalur file di `credentialSpecs` bidang definisi tugas. Untuk masing-masing opsi ini, Anda dapat memberikan `credentialspec:` ataudomainless`credentialspec:`, tergantung pada apakah Anda menggabungkan instance penampung ke satu domain, atau menggunakan domainlessgMSA, masing-masing.

Bucket Amazon S3

Tambahkan spesifikasi kredenal ke bucket Amazon S3, lalu rujuk Nama Sumber Daya Amazon (ARN) bucket Amazon S3 di bidang definisi tugas. `credentialSpecs`

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
```

```
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

Anda juga harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS untuk memberikan akses tugas Anda ke bucket Amazon S3.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

Parameter SSM Menyimpan parameter

Tambahkan spesifikasi kredenal ke parameter Penyimpanan Parameter SSM, lalu rujuk Nama Sumber Daya Amazon (ARN) parameter Penyimpanan Parameter SSM di bidang definisi tugas. `credentialSpecs`

```
{
    "family": "",
```

```
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [

 "credentialspecdomainless:arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

Anda juga harus menambahkan izin berikut sebagai kebijakan inline ke peran IAM eksekusi tugas Amazon ECS untuk memberikan akses tugas Anda ke parameter Penyimpanan Parameter SSM.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
            "arn:aws:ssm:arn:aws::111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

File Lokal

Dengan detail spesifikasi kredensial dalam file lokal, referensi jalur file di bidang ketentuan tugas `credentialSpecs`. Jalur file yang direferensikan harus relatif terhadap `C:\ProgramData\Docker\CredentialSpecs` direktori dan menggunakan garis miring terbalik ('\') sebagai pemisah jalur file.

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspec:file://CredentialSpecDir\CredentialSpecFile.json"
            ],
            ...
        }
    ],
    ...
}
```

Menggunakan EC2 Image Builder untuk membuat Amazon ECS yang disesuaikan yang dioptimalkan AMIs

AWS merekomendasikan agar Anda menggunakan Amazon ECS yang dioptimalkan AMIs karena sudah dikonfigurasi sebelumnya dengan persyaratan dan rekomendasi untuk menjalankan beban kerja penampung Anda. Mungkin ada saat-saat ketika Anda perlu menyesuaikan AMI Anda untuk menambahkan perangkat lunak tambahan. Anda dapat menggunakan EC2 Image Builder untuk pembuatan, pengelolaan, dan penyebaran image server Anda. Anda mempertahankan kepemilikan gambar yang disesuaikan yang dibuat di akun Anda. Anda dapat menggunakan pipeline EC2 Image Builder untuk mengotomatiskan pembaruan dan penambalan sistem untuk gambar Anda, atau menggunakan perintah yang berdiri sendiri untuk membuat gambar dengan sumber daya konfigurasi yang ditentukan.

Anda membuat resep untuk gambar Anda. Resepnya mencakup gambar induk, dan komponen tambahan apa pun. Anda juga membuat pipeline yang mendistribusikan AMI khusus Anda.

Anda membuat resep untuk gambar Anda. Resep image Builder Image adalah dokumen yang mendefinisikan image dasar dan komponen yang diterapkan pada image dasar untuk menghasilkan konfigurasi yang diinginkan untuk gambar AMI keluaran. Anda juga membuat pipeline yang mendistribusikan AMI khusus Anda. Untuk informasi selengkapnya, lihat Cara kerja EC2 Image Builder di Panduan Pengguna EC2 Image Builder.

Kami menyarankan Anda menggunakan salah satu Amazon ECS berikut yang dioptimalkan AMIs sebagai “Gambar Induk” Anda di EC2 Image Builder:

- Linux
  - Amazon ECS dioptimalkan 023 x86 AL2
  - Amazon ECS yang dioptimalkan Amazon Linux 2023 (arm64) AMI
  - Kernel 5 AMI Amazon Linux 2 yang dioptimalkan oleh Amazon ECS
  - Amazon ECS yang dioptimalkan Amazon Linux 2 x86 AMI
- Windows
  - Amazon ECS-dioptimalkan Windows 2022 Full x86
  - Amazon ECS yang dioptimalkan Windows 2022 Core x86
  - Amazon ECS Windows 2019 Full x86 yang dioptimalkan oleh Amazon
  - Amazon ECS yang dioptimalkan Windows 2019 Core x86
  - Amazon ECS yang dioptimalkan Windows 2016 Full x86

Kami juga menyarankan Anda memilih “Gunakan versi OS terbaru yang tersedia”. Pipeline akan menggunakan versi semantik untuk gambar induk, yang membantu mendeteksi pembaruan ketergantungan dalam pekerjaan yang dijadwalkan secara otomatis. Untuk informasi selengkapnya, lihat Pembuatan versi semantik di Panduan Pengguna Image BuilderEC2 .

AWS secara teratur memperbarui gambar AMI Amazon ECS yang dioptimalkan dengan tambalan keamanan dan versi agen kontainer baru. Saat Anda menggunakan ID AMI sebagai gambar induk Anda dalam resep gambar Anda, Anda perlu secara teratur memeriksa pembaruan pada gambar induk. Jika ada pembaruan, Anda harus membuat versi baru resep Anda dengan AMI yang diperbarui. Ini memastikan bahwa gambar kustom Anda menggabungkan versi terbaru dari gambar induk. Untuk informasi tentang cara membuat alur kerja untuk memperbarui EC2 instans secara otomatis di klaster Amazon ECS dengan yang baru dibuat AMIs, lihat Cara membuat pipeline pengerasan AMI dan mengotomatiskan pembaruan pada armada instans ECS Anda.

Anda juga dapat menentukan Amazon Resource Name (ARN) dari gambar induk yang dipublikasikan melalui pipeline EC2 Image Builder terkelola. Amazon secara rutin menerbitkan gambar AMI Amazon ECS yang dioptimalkan melalui saluran pipa terkelola. Gambar-gambar ini dapat diakses publik. Anda harus memiliki izin yang benar untuk mengakses gambar. Saat Anda menggunakan ARN gambar alih-alih AMI dalam resep Image Builder, pipeline Anda secara otomatis menggunakan versi terbaru

dari gambar induk setiap kali dijalankan. Pendekatan ini menghilangkan kebutuhan untuk membuat versi resep baru secara manual untuk setiap pembaruan.

Menggunakan gambar ARN dengan infrastruktur sebagai kode (IAc)

Anda dapat mengonfigurasi resep menggunakan konsol EC2 Image Builder, atau infrastruktur sebagai kode (misalnya, AWS CloudFormation), atau AWS SDK. Saat menentukan gambar induk dalam resep, Anda dapat menentukan ID EC2 AMI, ARN image Builder Image Builder, ID AWS Marketplace produk, atau gambar kontainer. AWS menerbitkan gambar AMI IDs dan Image Builder ARNs dari Amazon ECS AMIs yang dioptimalkan secara publik. Berikut ini adalah format ARN untuk gambar:

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

Ini `ImageVersion` memiliki format berikut. Ganti*`major`*, *`minor`* dan *`patch`* dengan nilai terbaru.

```
<major>.<minor>.<patch>
```

Anda dapat menggantimajor, minor dan patch ke nilai tertentu atau Anda dapat menggunakan ARN tanpa versi dari sebuah gambar, sehingga pipeline Anda tetap up-to-date dengan versi terbaru dari gambar induk. ARN tanpa versi menggunakan format wildcard 'x.x.x' untuk mewakili versi gambar. Pendekatan ini memungkinkan layanan Image Builder untuk secara otomatis menyelesaikan ke versi terbaru dari gambar. Menggunakan ARN tanpa versi memastikan bahwa referensi Anda selalu mengarah ke gambar terbaru yang tersedia, merampingkan proses mempertahankan gambar terbaru dalam penerapan Anda. Saat Anda membuat resep dengan konsol, EC2 Image Builder secara otomatis mengidentifikasi ARN untuk gambar induk Anda. Saat Anda menggunakan IAc untuk membuat resep, Anda harus mengidentifikasi ARN dan memilih versi gambar yang diinginkan atau menggunakan arn tanpa versi untuk menunjukkan gambar terbaru yang tersedia. Kami menyarankan Anda membuat skrip otomatis untuk memfilter dan hanya menampilkan gambar yang sesuai dengan kriteria Anda. Skrip Python berikut menunjukkan cara mengambil daftar Amazon ECS yang dioptimalkan. AMIs

Skrip menerima dua argumen opsional: owner danplatform, dengan nilai default “Amazon”, dan “Windows” masing-masing. Nilai yang valid untuk argumen pemilik adalah:Self,Shared,Amazon, danThirdParty. Nilai yang valid untuk argumen platform adalah Windows danLinux. Misalnya, jika Anda menjalankan skrip dengan owner argumen yang disetel ke Amazon dan platform disetel keLinux, skrip akan menghasilkan daftar gambar yang diterbitkan oleh Amazon termasuk gambar yang dioptimalkan Amazon ECS.

```
import boto3
import argparse

def list_images(owner, platform):
    # Create a Boto3 session
    session = boto3.Session()

    # Create an EC2 Image Builder client
    client = session.client('imagebuilder')

    # Define the initial request parameters
    request_params = {
        'owner': owner,
        'filters': [
            {
                'name': 'platform',
                'values': [platform]
            }
        ]
    }

    # Initialize the results list
    all_images = []

    # Get the initial response with the first page of results
    response = client.list_images(**request_params)

    # Extract images from the response
    all_images.extend(response['imageVersionList'])

    # While 'nextToken' is present, continue paginating
    while 'nextToken' in response and response['nextToken']:
        # Update the token for the next request
        request_params['nextToken'] = response['nextToken']

        # Get the next page of results
        response = client.list_images(**request_params)

        # Extract images from the response
        all_images.extend(response['imageVersionList'])

    return all_images
```

```
def main():
    # Initialize the parser
    parser = argparse.ArgumentParser(description="List AWS images based on owner and
 platform")

    # Add the parameters/arguments
    parser.add_argument("--owner", default="Amazon", help="The owner of the images.
 Default is 'Amazon'.")
    parser.add_argument("--platform", default="Windows", help="The platform type of the
 images. Default is 'Windows'.")

    # Parse the arguments
    args = parser.parse_args()

    # Retrieve all images based on the provided owner and platform
    images = list_images(args.owner, args.platform)

    # Print the details of the images
    for image in images:
        print(f"Name: {image['name']}, Version: {image['version']}, ARN:
 {image['arn']}")

if __name__ == "__main__":
    main()
```

Menggunakan gambar ARN dengan AWS CloudFormation

Resep gambar Image Builder adalah cetak biru yang menentukan gambar induk dan komponen yang diperlukan untuk mencapai konfigurasi gambar keluaran yang diinginkan. Anda menggunakan `AWS::ImageBuilder::ImageRecipe` sumber daya. Atur `ParentImage` nilai ke gambar ARN. Gunakan ARN tanpa versi dari gambar yang Anda inginkan untuk memastikan pipeline Anda selalu menggunakan versi gambar terbaru. Misalnya, `arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x`. Definisi `AWS::ImageBuilder::ImageRecipe` sumber daya berikut menggunakan ARN image terkelola Amazon:

```
ECSRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: MyRecipe
      Version: '1.0.0'
      Components:
```

```
        - ComponentArn: [<The component arns of the image recipe>]
      ParentImage: "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-
optimized-x86/x.x.x"
```

Untuk informasi selengkapnya tentang `AWS::ImageBuilder::ImageRecipe`sumber daya, lihat di Panduan AWS CloudFormation Pengguna.

Anda dapat mengotomatiskan pembuatan gambar baru di pipeline Anda dengan menyetel `Schedule` properti `AWS::ImageBuilder::ImagePipeline` sumber daya. Jadwal mencakup kondisi awal dan ekspresi cron. Untuk informasi selengkapnya, lihat `AWS::ImageBuilder::ImagePipeline` di AWS CloudFormation Panduan Pengguna.

`AWS::ImageBuilder::ImagePipeline`Contoh berikut ini memiliki pipeline yang menjalankan build pada pukul 10:00 AM Coordinated Universal Time (UTC) setiap hari. Tetapkan `Schedule` nilai-nilai berikut:

- Atur `PipelineExecutionStartCondition` ke `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`. Build dimulai hanya jika sumber daya dependen seperti image induk atau komponen, yang menggunakan wildcard 'x' dalam versi semantiknya, diperbarui. Ini memastikan build menggabungkan pembaruan terbaru dari sumber daya tersebut.
- Setel ScheduleExpression ke ekspresi (`0 10 * * ? *`) cron.

```
ECSPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: my-pipeline
      ImageRecipeArn: <arn of the recipe you created in previous step>
      InfrastructureConfigurationArn: <ARN of the infrastructure configuration
 associated with this image pipeline>
      Schedule:
        PipelineExecutionStartCondition:
 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE
        ScheduleExpression: 'cron(0 10 * * ? *)'
```

Menggunakan gambar ARN dengan Terraform

Pendekatan untuk menentukan gambar dan jadwal induk pipeline Anda di Terraform sejalan dengan yang ada di. AWS CloudFormation Anda menggunakan `aws_imagebuilder_image_recipe` sumber daya. Atur `parent_image` nilai ke gambar ARN. Gunakan ARN tanpa versi dari gambar yang Anda inginkan untuk memastikan pipeline Anda selalu menggunakan versi gambar terbaru. Untuk informasi lebih lanjut, lihat `aws_imagebuilder_image_recipe`di dokumentasi Terraform.

Di blok konfigurasi jadwal`aws_imagebuilder_image_pipeline resource`, atur nilai `schedule_expression` argumen ke ekspresi cron pilihan Anda untuk menentukan seberapa sering pipeline berjalan, dan atur `pipeline_execution_start_condition` ke`EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`. Untuk informasi lebih lanjut, lihat `aws_imagebuilder_image_pipeline`di dokumentasi Terraform.

Menggunakan AWS Deep Learning Containers di Amazon ECS

AWS Deep Learning Containers menyediakan satu set image Docker untuk melatih dan menyajikan model di TensorFlow dan Apache MXNet (Inkubasi) di Amazon ECS. Deep Learning Containers memungkinkan lingkungan yang dioptimalkan dengan TensorFlow library NVIDIA CUDA (untuk instans GPU), dan Intel MKL (untuk instance CPU). Gambar kontainer untuk Deep Learning Containers tersedia di Amazon ECR untuk referensi dalam definisi tugas Amazon ECS. Anda dapat menggunakan Deep Learning Containers bersama dengan Amazon Elastic Inference untuk menurunkan biaya inferensi Anda.

Untuk mulai menggunakan Deep Learning Containers tanpa Elastic Inference di Amazon ECS, lihat penyiapan Amazon ECS di Panduan Pengembang.AWS Deep Learning AMIs

Kuota layanan Amazon ECS

Saat Anda membuat Akun AWS, kami menetapkan kuota default (juga disebut sebagai batas) pada AWS sumber daya Anda per wilayah. Jika Anda mencoba melebihi kuota sumber daya, permintaan gagal.

Konsol Service Quotas adalah lokasi pusat tempat Anda dapat melihat dan mengelola kuota untuk AWS layanan, dan meminta peningkatan kuota untuk banyak sumber daya yang Anda gunakan. Gunakan kuota yang kami sediakan untuk mengelola sumber daya Anda. Rencanakan permintaan peningkatan kuota sebelum Anda membutuhkannya.

Untuk informasi selengkapnya tentang kuota Amazon ECS dan Fargate, lihat titik akhir dan kuota Amazon ECS di. Referensi Umum Amazon Web Untuk informasi tentang pembatasan API di Amazon ECS API, lihat Meminta pembatasan untuk Amazon ECS API.

Mengelola Amazon ECS dan kuota AWS Fargate layanan Anda di AWS Management Console

Amazon ECS telah terintegrasi dengan Service Quotas, AWS sebuah layanan yang memungkinkan Anda untuk melihat dan mengelola kuota Anda dari lokasi pusat. Untuk informasi selengkapnya, lihat Apa itu Service Quotas? dalam Panduan Pengguna Service Quotas.

Service Quotas memudahkan untuk mencari nilai kuota layanan Amazon ECS Anda.

AWS Management Console

Untuk melihat kuota layanan Amazon ECS dan Fargate menggunakan AWS Management Console

1. Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.
2. Di panel navigasi, pilih Layanan AWS .
3. Dari daftar AWS layanan, cari dan pilih Amazon Elastic Container Service (Amazon ECS) atau. AWS Fargate

   Dalam daftar service quotas, Anda dapat melihat nama service quotas, nilai terapan (jika tersedia), kuota default AWS , dan apakah nilai kuota dapat disesuaikan.
4. Untuk melihat informasi tambahan tentang service quotas, seperti deskripsi, pilih nama kuota.

5. (Opsional) Untuk meminta peningkatan kuota, pilih kuota yang ingin Anda tingkatkan, kemudian pilih Meminta peningkatan kuota, masukkan atau pilih informasi yang diperlukan, dan pilih Minta.

Untuk bekerja lebih banyak dengan kuota layanan menggunakan AWS Management Console lihat Panduan Pengguna Service Quotas. Untuk meminta penambahan kuota, lihat Meminta penambahan kuota di Panduan Pengguna Service Quotas.

AWS CLI

Untuk melihat kuota layanan Amazon ECS dan Fargate menggunakan AWS CLI

Jalankan perintah berikut untuk melihat kuota Amazon ECS default.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code ecs \
    --output table
```

Jalankan perintah berikut untuk melihat kuota Fargate default.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

Jalankan perintah berikut untuk melihat kuota Fargate yang Anda terapkan.

```
aws service-quotas list-service-quotas \
    --service-code fargate
```

> **Note**
>
> Amazon ECS tidak mendukung kuota yang diterapkan.

Untuk informasi selengkapnya tentang bekerja dengan kuota layanan menggunakan AWS CLI, lihat Referensi Perintah Service AWS CLI Quotas. Untuk meminta peningkatan kuota, lihat perintah `request-service-quota-increase` dalam Referensi Perintah AWS CLI.

Menangani kuota layanan Amazon ECS dan batas pembatasan API

Amazon ECS terintegrasi dengan beberapa Layanan AWS, termasuk Elastic Load Balancing AWS Cloud Map,, dan Amazon. EC2 Dengan integrasi yang ketat ini, Amazon ECS mencakup beberapa fitur seperti service load balancing, Service Connect, task networking, dan cluster auto scaling. Amazon ECS dan lainnya Layanan AWS yang terintegrasi dengan semua kuota layanan pemeliharaan dan batas tarif API untuk memastikan kinerja dan pemanfaatan yang konsisten. Kuota layanan ini juga mencegah penyediaan sumber daya yang tidak disengaja daripada yang dibutuhkan dan melindungi terhadap tindakan jahat yang dapat meningkatkan tagihan Anda.

Dengan membiasakan diri dengan kuota layanan dan batas tarif AWS API, Anda dapat merencanakan penskalaan beban kerja Anda tanpa khawatir tentang penurunan kinerja yang tidak terduga. Untuk informasi selengkapnya, lihat Meminta pembatasan untuk Amazon ECS API.

Saat menskalakan beban kerja Anda di Amazon ECS, kami sarankan Anda mempertimbangkan kuota layanan berikut.

- AWS Fargate memiliki kuota yang membatasi jumlah tugas yang berjalan bersamaan di masing-masing tugas. Wilayah AWS Ada kuota untuk tugas On-Demand dan Fargate Spot di Amazon ECS. Setiap kuota layanan juga mencakup semua pod Amazon EKS yang Anda jalankan di Fargate.
- Untuk tugas yang berjalan di EC2 instans Amazon, jumlah maksimum EC2 instans Amazon yang dapat Anda daftarkan untuk setiap cluster adalah 5.000. Jika Anda menggunakan penskalaan otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, atau jika Anda mengelola instans EC2 Amazon untuk klaster Anda sendiri, kuota ini mungkin menjadi hambatan penerapan. Jika Anda membutuhkan kapasitas lebih, Anda dapat membuat lebih banyak cluster atau meminta peningkatan kuota layanan.
- Jika Anda menggunakan penskalaan otomatis klaster Amazon ECS dengan penyedia kapasitas grup Auto Scaling, pertimbangkan kuota saat menskalakan layanan Anda. `Tasks in the PROVISIONING state per cluster` Kuota ini adalah jumlah maksimum tugas di `PROVISIONING` negara bagian untuk setiap cluster di mana penyedia kapasitas dapat

meningkatkan kapasitas. Ketika Anda meluncurkan sejumlah besar tugas sekaligus, Anda dapat dengan mudah memenuhi kuota ini. Salah satu contohnya adalah jika Anda secara bersamaan menyebarkan puluhan layanan, masing-masing dengan ratusan tugas. Ketika ini terjadi, penyedia kapasitas perlu meluncurkan instance kontainer baru untuk menempatkan tugas ketika cluster memiliki kapasitas yang tidak mencukupi. Sementara penyedia kapasitas meluncurkan EC2 instans Amazon tambahan, penjadwal layanan Amazon ECS kemungkinan akan terus meluncurkan tugas secara paralel. Namun, aktivitas ini mungkin terhambat karena kapasitas cluster yang tidak mencukupi. Penjadwal layanan Amazon ECS mengimplementasikan strategi back-off dan exponential throttling untuk mencoba kembali penempatan tugas saat instance container baru diluncurkan. Akibatnya, Anda mungkin mengalami waktu penerapan atau penskalaan yang lebih lambat. Untuk menghindari situasi ini, Anda dapat merencanakan penerapan layanan Anda di salah satu yang berikut. Menerapkan sejumlah besar tugas tidak memerlukan peningkatan kapasitas cluster, atau mempertahankan kapasitas cluster cadangan untuk peluncuran tugas baru.

Selain mempertimbangkan kuota layanan Amazon ECS saat menskalakan beban kerja Anda, pertimbangkan juga kuota layanan untuk yang lain Layanan AWS yang terintegrasi dengan Amazon ECS.

Penyeimbang Beban Elastis

Anda dapat mengonfigurasi layanan Amazon ECS agar menggunakan Elastic Load Balancing untuk mendistribusikan lalu lintas secara merata di seluruh tugas. Untuk informasi selengkapnya dan praktik terbaik yang direkomendasikan tentang cara memilih penyeimbang beban, lihatGunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.

Kuota layanan Elastic Load Balancing

Saat Anda menskalakan beban kerja Anda, pertimbangkan kuota layanan Elastic Load Balancing berikut. Sebagian besar kuota layanan Elastic Load Balancing dapat disesuaikan, dan Anda dapat meminta peningkatan konsol Service Quotas.

Penyeimbang Beban Aplikasi

Bila Anda menggunakan Application Load Balancer, tergantung pada kasus penggunaan Anda, Anda mungkin perlu meminta peningkatan kuota untuk:

- `Targets per Application Load Balancer`Kuota yang merupakan jumlah target di belakang Application Load Balancer Anda.

- `Targets per Target Group per Region`Kuota yang merupakan jumlah target di belakang Grup Target Anda.

Untuk informasi selengkapnya, lihat Kuota untuk Penyeimbang Beban Aplikasi Anda di Panduan Pengguna untuk Penyeimbang Beban Aplikasi.

Penyeimbang Beban Jaringan

Ada batasan yang lebih ketat pada jumlah target yang dapat Anda daftarkan dengan Network Load Balancer. Saat menggunakan Network Load Balancer, Anda sering ingin mengaktifkan dukungan lintas zona, yang dilengkapi dengan batasan penskalaan tambahan pada `Targets per Availability Zone Per Network Load Balancer` jumlah maksimum target per Availability Zone untuk setiap Network Load Balancer. Untuk informasi selengkapnya, lihat Kuota untuk Network Load Balancer Anda di Panduan Pengguna untuk Network Load Balancers.

Pelambatan API Elastic Load Balancing

Saat Anda mengonfigurasi layanan Amazon ECS untuk menggunakan penyeimbang beban, pemeriksaan kesehatan grup target harus lulus sebelum layanan dianggap sehat. Untuk melakukan pemeriksaan kesehatan ini, Amazon ECS memanggil operasi Elastic Load Balancing API atas nama Anda. Jika Anda memiliki sejumlah besar layanan yang dikonfigurasi dengan penyeimbang beban di akun Anda, Anda mungkin akan memperlambat penerapan layanan karena potensi pembatasan khusus untuk operasi API Elastic Load Balancing, dan `RegisterTarget` Elastic `DeregisterTarget` Load `DescribeTargetHealth` Balancing. Saat pelambatan terjadi, kesalahan pelambatan terjadi di pesan peristiwa layanan Amazon ECS Anda.

Jika mengalami pelambatan AWS Cloud Map API, Anda dapat menghubungi Dukungan untuk panduan tentang cara meningkatkan batas pelambatan AWS Cloud Map API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat. Menangani masalah pelambatan Amazon ECS

Antarmuka jaringan elastis

Dengan tugas Anda menggunakan mode `awsvpc` jaringan, Amazon ECS menyediakan elastic network interface (ENI) yang unik untuk setiap tugas. Jika layanan Amazon ECS Anda menggunakan penyeimbang beban Elastic Load Balancing, antarmuka jaringan ini juga terdaftar sebagai target ke grup target yang sesuai yang ditentukan dalam layanan.

Kuota layanan antarmuka jaringan elastis

Ketika Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, sebuah elastic network interface yang unik dilampirkan ke setiap tugas. Jika tugas-tugas tersebut harus dicapai melalui internet, tetapkan alamat IP publik ke elastic network interface untuk tugas-tugas tersebut. Saat Anda menskalakan beban kerja Amazon ECS Anda, pertimbangkan dua kuota penting ini:

- `Network interfaces per Region`Kuota yang merupakan jumlah maksimum antarmuka jaringan di akun Wilayah AWS Anda.
- `Elastic IP addresses per Region`Kuota yang merupakan jumlah maksimum alamat IP elastis dalam file Wilayah AWS.

Kedua kuota layanan ini dapat disesuaikan dan Anda dapat meminta peningkatan dari konsol Service Quotas Anda untuk ini. Untuk informasi selengkapnya, lihat kuota layanan Amazon VPC di Panduan pengguna Amazon Virtual Private Cloud.

Untuk beban kerja Amazon ECS yang dihosting di EC2 instans Amazon, saat menjalankan tugas yang menggunakan mode `awsvpc` jaringan, pertimbangkan kuota `Maximum network interfaces` layanan, jumlah maksimum instans jaringan untuk setiap instans Amazon. EC2 Kuota ini membatasi jumlah tugas yang dapat Anda tempatkan pada sebuah instance. Anda tidak dapat menyesuaikan kuota dan tidak tersedia di konsol Service Quotas. Untuk informasi selengkapnya, lihat alamat IP per antarmuka jaringan per jenis instans di Panduan EC2 Pengguna Amazon.

Meskipun Anda tidak dapat mengubah jumlah antarmuka jaringan yang dapat dilampirkan ke EC2 instans Amazon, Anda dapat menggunakan fitur trunking elastic network interface untuk meningkatkan jumlah antarmuka jaringan yang tersedia. Misalnya, secara default sebuah `c5.large` instance dapat memiliki hingga tiga antarmuka jaringan. Antarmuka jaringan utama untuk instance dihitung sebagai satu. Jadi, Anda dapat melampirkan dua antarmuka jaringan tambahan ke instance. Karena setiap tugas yang menggunakan mode `awsvpc` jaringan memerlukan antarmuka jaringan, Anda biasanya hanya dapat menjalankan dua tugas tersebut pada jenis instance ini. Hal ini dapat menyebabkan kurangnya pemanfaatan kapasitas cluster Anda. Jika Anda mengaktifkan trunking elastic network interface, Anda dapat meningkatkan kepadatan antarmuka jaringan untuk menempatkan lebih banyak tugas pada setiap instance. Dengan trunking dihidupkan, sebuah `c5.large` instance dapat memiliki hingga 12 antarmuka jaringan. Instans memiliki antarmuka jaringan utama dan Amazon ECS membuat dan melampirkan antarmuka jaringan “trunk” ke instance. Akibatnya, dengan konfigurasi ini Anda dapat menjalankan 10 tugas pada instance alih-alih dua

tugas default. Untuk informasi selengkapnya, lihat Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux.

Pelambatan API antarmuka jaringan elastis

Saat Anda menjalankan tugas yang menggunakan mode `awsvpc` jaringan, Amazon ECS bergantung pada Amazon berikut. EC2 APIs Masing-masing APIs memiliki throttle API yang berbeda. Untuk informasi selengkapnya, lihat Meminta pembatasan untuk Amazon EC2 API di Referensi Amazon EC2 API.

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

Jika panggilan Amazon EC2 API dibatasi selama alur kerja penyediaan elastic network interface, penjadwal layanan Amazon ECS secara otomatis mencoba ulang dengan back-off eksponensial. Pensiun otomatis ini terkadang dapat menyebabkan penundaan dalam meluncurkan tugas, yang menghasilkan kecepatan penerapan yang lebih lambat. Ketika API throttling terjadi, Anda akan melihat pesan `Operations are being throttled. Will try again later.` pada pesan acara layanan Anda. Jika Anda secara konsisten memenuhi throttle Amazon EC2 API, Anda dapat menghubungi Dukungan untuk panduan tentang cara meningkatkan batas pembatasan API Anda. Untuk informasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan, lihat Menangani masalah pelambatan.

AWS Cloud Map

Penemuan layanan Amazon ECS dan Service Connect digunakan AWS Cloud Map APIs untuk mengelola ruang nama untuk layanan Amazon ECS Anda. Jika layanan Anda memiliki banyak tugas, pertimbangkan rekomendasi berikut.

AWS Cloud Map kuota layanan

Ketika layanan Amazon ECS dikonfigurasi untuk menggunakan service discovery atau Service Connect, `Tasks per service` kuota yang merupakan jumlah maksimum tugas untuk layanan, dipengaruhi oleh kuota AWS Cloud Map `Instances per service` layanan yang merupakan jumlah maksimum instans untuk layanan tersebut. Secara khusus, kuota AWS Cloud Map layanan mengurangi jumlah instans yang dapat Anda jalankan paling banyak 1.000 instans per layanan. Anda tidak dapat mengubah AWS Cloud Map kuota. Untuk informasi selengkapnya, lihat AWS Cloud Map service quotas.

AWS Cloud Map Pelambatan API

Amazon ECS memanggil`ListInstances`,, `GetInstancesHealthStatus``RegisterInstance`, dan `DeregisterInstance` AWS Cloud Map APIs atas nama Anda. Mereka membantu dengan penemuan layanan dan melakukan pemeriksaan kesehatan ketika Anda meluncurkan tugas. Ketika beberapa layanan yang menggunakan penemuan layanan dengan sejumlah besar tugas diterapkan pada saat yang sama, hal ini dapat mengakibatkan melebihi batas pembatasan AWS Cloud Map API. Ketika ini terjadi, Anda mungkin akan melihat pesan berikut: `0perations are being throttled. Will try again later` di pesan acara layanan Amazon ECS Anda dan penerapan yang lebih lambat dan kecepatan peluncuran tugas. AWS Cloud Map tidak mendokumentasikan batas pelambatan untuk ini. APIs Jika Anda mengalami pembatasan dari ini, Anda dapat menghubungi Dukungan untuk panduan tentang peningkatan batas pelambatan API Anda. Untuk rekomendasi selengkapnya tentang pemantauan dan pemecahan masalah kesalahan pelambatan tersebut, lihat. Menangani masalah pelambatan Amazon ECS

Referensi API Amazon ECS

Selain AWS Management Console dan AWS Command Line Interface (AWS CLI), Amazon ECS juga menyediakan API. Anda dapat menggunakan API untuk mengotomatiskan tugas untuk mengelola sumber daya Amazon ECS.

- Untuk daftar operasi API menurut resource Amazon ECS, lihat Actions by Amazon ECS resource.
- Untuk daftar abjad operasi API, lihat Tindakan.
- Untuk daftar abjad jenis data, lihat Jenis data.
- Untuk daftar parameter kueri umum, lihat Parameter umum.
- Untuk deskripsi kode kesalahan, lihat Kesalahan umum.

Untuk informasi selengkapnya tentang AWS CLI, lihat AWS Command Line Interface referensi untuk Amazon ECS.

Riwayat dokumen

Tabel berikut menjelaskan pembaruan utama dan fitur baru untuk Amazon Elastic Container Service Developer Guide. Kami juga secara rutin memperbarui dokumentasi untuk menjawab umpan balik yang Anda kirimkan kepada kami.

Perubahan	Deskripsi	Tanggal
Tambahkan izin ke kebijakan terkelola EC2ContainerServiceforEC2Peran Amazon.	Kebijakan IAM `AmazonEC2ContainerServiceforEC2Role` terkelola telah diperbarui untuk menyertakan `ecs:ListTagsForResource` izin. Izin ini memungkinkan agen Amazon ECS untuk mengambil tag instance tugas dan container melalui titik akhir metadata tugas (). `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags` Untuk informasi selengkapnya, lihat EC2ContainerServiceforEC2Peran Amazon.	Agustus 4, 2025
Tambahkan izin ke Amazon ECSInfrastructure RolePolicyForLoadBalancers.	Kebijakan IAM `AmazonECSInfrastructureRolePolicyForLoadBalancers` terkelola diperbarui dengan izin baru untuk mendeskripsikan, membatalkan pendaftaran, dan mendaftarkan grup sasaran.	Juli 25, 2025
Support untuk memperbarui pengontrol penyebaran layanan	Amazon ECS mendukung pembaruan pengontrol penyebaran layanan, memungkinkan Anda mengubah di antara berbagai jenis penerapan untuk layanan yang ada. Untuk informasi selengkapnya, lihat Memperbarui parameter layanan Amazon ECS.	Juli 15, 2025
Penerapan asli blue/green Amazon ECS	Amazon ECS mendukung blue/green penerapan asli, memungkinkan Anda memvalidasi revisi layanan baru sebelum mengarahkan lalu lintas produksi ke sana. Untuk informasi selengkapnya, lihat blue/green penerapan Amazon ECS.	Juli 15, 2025
Tambahkan kebijakan ECSInfrastructure	Kebijakan ini menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola	Juli 15, 2025

Perubahan	Deskripsi	Tanggal
RolePolicyForLoadBalancers terkelola Amazon baru.	penyeimbang beban yang terkait dengan beban kerja Amazon ECS atas nama Anda. Untuk informasi selengkapnya, lihat Amazon ECSInfrastructureRolePolicyForLoadBalancers.	
Dukungan manifes indeks SOCI v2	Pelanggan Fargate yang ingin menggunakan SOCI hanya akan dapat menggunakan manifes indeks SOCI v2. Pelanggan lama yang sebelumnya telah menggunakan SOCI di Fargate dapat terus menggunakan manifes indeks SOCI v1, namun kami sangat menyarankan pelanggan tersebut untuk bermigrasi ke manifes indeks SOCI v2. Untuk informasi selengkapnya, lihat Lazy loading gambar kontainer menggunakan Seekable OCI (SOCI).	Juli 2, 2025
Menambahkan ID tugas dalam peristiwa tindakan layanan yang dihasilkan karena kegagalan kesehatan.	Untuk informasi selengkapnya, lihat pesan acara tidak sehat layanan Amazon ECS.	Juni 30, 2025
Perbarui ke mode driver log default.	Mode logging default untuk Amazon ECS telah diperbarui dari `blocking` kenon-blocking . Anda dapat mengganti mode default ini dengan menentukan definisi kontainer `logConfiguration` atau mengonfigurasi default Anda sendiri dengan menyetel setelan `defaultLogDriverMode` akun menggunakan konsol Amazon ECS atau. `mode` AWS CLI Untuk informasi selengkapnya, lihat Mode driver log default.	Juni 25, 2025
Tambahkan izin ke Amazon ECSServiceRolePolicy.	Kebijakan IAM `AmazonECSServiceRolePolicy` terkelola telah diperbarui dengan AWS Cloud Map izin baru yang Amazon ECS dapat memperbarui atribut AWS Cloud Map layanan untuk layanan yang dikelola Amazon ECS.	24 Juni 2025

Perubahan	Deskripsi	Tanggal
Support untuk Windows Server 2025 Full AMI yang dioptimalkan Amazon ECS dan Windows Server 2025 Core AMI yang dioptimal kan Amazon ECS dan Amazon ECS.	Untuk informasi selengkapnya, lihat Windows Amazon ECS yang dioptimalkan. AMIs	Juni 20, 2025
Support untuk memperbarui strategi penyedia kapasitas layanan.	Anda sekarang dapat memperbarui strategi penyedia kapasitas layanan. Untuk informasi selengkapnya, lihat Memperbarui parameter layanan.	12 Juni 2025
Tambahkan izin ke Amazon ECSInfras tructure RolePolic yForVolumes	`AmazonECSInfrastructureRolePolicyFor Volumes` Kebijakan telah diperbarui untuk menambahkan `ec2:DescribeInstances` izin. Izin tersebut membantu mencegah tabrakan nama perangkat untuk volume Amazon EBS yang dilampirk an ke tugas Amazon ECS yang berjalan pada instance container yang sama.	Juni 2, 2025
Support untuk tingkat inisialisasi volume untuk volume Amazon EBS yang dilampirkan ke tugas Amazon ECS.	Anda sekarang dapat mengatur `volumeInt iializationRate` untuk menentukan laju, di MIB/s, di mana data diambil dari snapshot volume Amazon EBS yang ada untuk membuat volume baru untuk lampiran ke tugas Amazon ECS. Untuk informasi selengkapnya, lihat volumeInitializationRate.	13 Mei 2025
Support untuk menghenti kan penyebaran layanan.	Anda sekarang dapat menghentikan penyebara n layanan. Untuk informasi selengkapnya, lihat Menghentikan penerapan layanan Amazon ECS.	2 Mei 2025

Perubahan	Deskripsi	Tanggal
Support untuk titik akhir dual-stack Amazon ECS.	Anda sekarang dapat menggunakan titik akhir dual-stack untuk berinteraksi dengan Amazon ECS. Titik akhir dual-stack mendukung permintaan ke Amazon ECS melalui Internet Protocol versi 4 (IPv4) dan Internet Protocol versi 6 (). IPv6 Untuk informasi selengkapnya, lihat Menggunakan titik akhir tumpukan ganda Amazon ECS.	April 30, 2025
Support untuk mengonfigurasi mode driver log default di tingkat akun Wilayah AWS di file.	Sekarang Anda dapat mengonfigurasi mode pengiriman default pesan log dari wadah ke driver log yang Anda tentukan dalam `logConfiguration` menggunakan pengaturan `defaultLogDriverMode` akun. Untuk informasi selengkapnya, lihat Mode driver log default.	17 April 2025
Support untuk memigrasikan layanan dengan format ARN pendek ke format ARN yang panjang.	Anda sekarang dapat memigrasikan layanan dengan format ARN pendek ke paket format ARN yang panjang. Untuk informasi selengkapnya, lihat Memigrasikan ARN layanan singkat Amazon ECS ke ARN yang panjang.	Februari 13, 2025
Support untuk tiga jenis injeksi kesalahan di Amazon ECS dan Fargate.	Anda sekarang dapat menggunakan port blackhole jaringan, latensi jaringan, dan injeksi kesalahan kehilangan paket jaringan dengan Amazon ECS dan Fargate. Untuk informasi selengkapnya, lihat Menggunakan AWS Fault Injection Service untuk menguji beban kerja Amazon ECS Anda.	Desember 19, 2024
Support untuk CloudWatch Container Insights dengan observabilitas yang ditingkatkan.	Anda dapat menggunakan CloudWatch Container Insights dengan observabilitas yang ditingkatkan. Untuk informasi selengkapnya, lihat Memantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.	Desember 2, 2024

Perubahan	Deskripsi	Tanggal
Support untuk penskalaan prediktif.	Anda dapat menggunakan penskalaan prediktif untuk mengantisipasi kebutuhan masa depan dan secara proaktif meningkatkan tugas dalam layanan Anda sesuai kebutuhan. Untuk informasi selengkapnya, lihat Menggunakan pola historis untuk menskalakan CloudWatch layanan dengan penskalaan prediktif.	November 21, 2024
Support untuk penyeimbangan kembali layanan.	Anda sekarang dapat meminta Amazon ECS secara otomatis menyeimbangkan tugas layanan Anda di seluruh Availability Zone. Untuk informasi selengkapnya, lihat Menyeimbangkan layanan Amazon ECS di seluruh Availability Zone.	November 19, 2024
Support untuk mengonfigurasi resolusi tag gambar kontainer ke intisari gambar.	Anda sekarang dapat mengonfigurasi resolusi tag gambar kontainer untuk digest untuk setiap kontainer dalam tugas yang merupakan bagian dari layanan. Untuk informasi selengkapnya, lihat Resolusi gambar kontainer dan `versionConsistency`.	November 19, 2024
Tambahkan kebijakan `AmazonECSInfrastructureRolePolicyForVpcLattice` IAM baru	`AmazonECSInfrastructureRolePolicyForVpcLattice` Kebijakan ini menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk mengelola fitur VPC Lattice di beban kerja Amazon ECS atas nama Anda.	November 18, 2024
VPC Lattice sekarang tersedia untuk layanan Amazon ECS.	Anda sekarang dapat menggunakan VPC Lattice untuk menstandarisasi service-to-service konektivitas, keamanan, dan observabilitas Anda. Untuk informasi selengkapnya, lihat Menggunakan Amazon VPC Lattice untuk menghubungkan, mengamati, dan mengamankan layanan Amazon ECS Anda.	November 18, 2024

Perubahan	Deskripsi	Tanggal
Support untuk riwayat layanan menggunakan penerapan layanan dan revisi layanan.	Anda dapat menggunakan penerapan layanan dan revisi layanan untuk melihat riwayat penerapan. Untuk informasi selengkapnya, lihat Melihat riwayat layanan menggunakan penerapan layanan Amazon ECS dan revisi layanan Amazon ECS.	Oktober 30, 2024
Support untuk volume Amazon EBS untuk jenis EC2 peluncuran menggunakan sistem operasi Windows.	Amazon ECS mendukung volume Amazon EBS yang dilampirkan ke tugas yang berjalan pada EC2 instance dengan sistem operasi Windows. Untuk informasi selengkapnya, lihat Menggunakan volume Amazon EBS dengan Amazon ECS.	Oktober 24, 2024
Tambahkan izin ke Amazon ECSInfrastructureRolePolicyForVolumes.	`AmazonECSInfrastructureRolePolicyForVolumes` Kebijakan ini telah diperbarui untuk memungkinkan pelanggan membuat volume Amazon EBS dari snapshot. Untuk informasi selengkapnya, lihat Amazon ECSInfrastructure RolePolicyForVolumes.	Oktober 10, 2024
Fargate Spot tersedia untuk Linux ARM.	Support untuk Fargate Spot di Linux ARM telah ditambahkan ke Amazon ECS.	September 6, 2024
Kebijakan restart kontainer untuk setiap kontainer dalam tugas Amazon ECS.	Anda dapat mengaktifkan kebijakan restart untuk wadah penting dan non-esensial yang ditentukan dalam definisi tugas Anda untuk mengatasi kegagalan sementara lebih cepat dan mempertahankan ketersediaan tugas. Untuk informasi selengkapnya, lihat Mulai ulang kontainer individual dalam tugas Amazon ECS dengan kebijakan restart kontainer.	Agustus 15, 2024

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke the section called “Amazonecs_ FullAccess”	`AmazonECS_FullAccess` Kebijakan telah diperbarui untuk menambahkan `iam:PassRole` izin untuk peran IAM untuk peran bernama. `ecsInfrastructureRole` Ini adalah peran IAM default yang dibuat oleh AWS Management Console yang dimaksudkan untuk digunakan sebagai peran infrastruktur ECS yang memungkinkan Amazon ECS mengelola volume Amazon EBS yang melekat pada tugas ECS.	Agustus 13, 2024
Amazon Linux 2023, CentOS Stream 9, Fedora 40, dan dukungan Ubuntu 24 ditambahkan untuk ECS Anywhere	Dukungan untuk sistem operasi Amazon Linux 2023, CentOS Stream 9, Fedora 40, dan Ubuntu 24 ditambahkan ke ECS Anywhere. Untuk informasi selengkapnya, lihat Sistem pengoperasian dan arsitektur sistem yang didukung.	Juli 23, 2024
Resolusi gambar kontainer untuk layanan yang menggunakan tipe penerapan rolling update (ECS).	Untuk memastikan bahwa semua tugas dalam layanan yang menggunakan pengontrol penerapan pembaruan bergulir menggunakan image kontainer yang sama, Amazon ECS menyelesaikan nama gambar kontainer dan tag gambar apa pun yang ditentukan dalam definisi tugas ke intisari gambar kontainer. Untuk informasi selengkapnya, lihat Resolusi gambar kontainer.	Juli 10, 2024
Dukungan Debian 11 dan Debian 12 ditambahkan untuk ECS Anywhere	Support untuk sistem operasi Debian 11 dan Debian 12 ditambahkan ke ECS Anywhere. Untuk informasi selengkapnya, lihat Sistem pengoperasian dan arsitektur sistem yang didukung.	Maret 28, 2024
GMSA untuk Kontainer Linux pada dukungan Fargate	Amazon ECS mendukung otentikasi Direktori Aktif untuk wadah Linux di Fargate melalui jenis akun layanan khusus yang disebut grup Akun Layanan Terkelola (GMSA). Untuk informasi lebih lanjut, lihat Menggunakan gMSA untuk Linux kontainer di Fargate.	Maret 5, 2024

Perubahan	Deskripsi	Tanggal
CloudWatch metrik ditambahkan untuk volume Amazon EBS yang dilampirkan ke tugas	Amazon ECS sekarang menerbitkan CloudWatch metrik untuk pemanfaatan penyimpanan Amazon EBS untuk tugas-tugas yang memiliki volume Amazon EBS terpasang. Untuk informasi selengkapnya, lihat CloudWatch metrik Amazon ECS.	Februari 8, 2024
Layanan Connect TLS	Anda sekarang dapat menggunakan TLS dengan Service Connect.	Januari 22, 2024
Kebijakan terkelola Service Connect TLS	Menambahkan ECSInfrastructure RolePolicyForServiceConnectTransportLayerSecurity kebijakan Amazon baru.	Januari 22, 2024
Pembaruan konfigurasi batas waktu tunggu Service Connect	Konfigurasi batas waktu Service Connect sekarang dapat diperbarui dan mencakup dua parameter opsional - `idleTimeout` dan `perRequestTimeout` .	Januari 22, 2024
Pengurasan instans terkelola Amazon ECS	Anda dapat menggunakan pengurasan instans terkelola Amazon ECS untuk memfasilitasi penghentian instans Amazon ECS secara anggun.	Januari 19, 2024
Dukungan Ubuntu 22 ditambahkan untuk ECS Anywhere	Support untuk sistem operasi Ubuntu 22 ditambahkan ke ECS Anywhere. Untuk informasi selengkapnya, lihat Sistem pengoperasian dan arsitektur sistem yang didukung.	Januari 16, 2024
Tambahkan `AmazonECSInfrastructureRolePolicyForVolumes` kebijakan IAM	`AmazonECSInfrastructureolePolicyForVolumes` telah ditambahkan. Kebijakan ini memberikan izin yang diperlukan oleh Amazon ECS untuk melakukan panggilan AWS API guna mengelola volume Amazon EBS yang terkait dengan beban kerja Amazon ECS.	Januari 11, 2024

Perubahan	Deskripsi	Tanggal
Volume data Amazon EBS untuk tugas Amazon ECS	Anda dapat mengonfigurasi 1 volume data Amazon EBS per tugas selama penerapan lampiran ke tugas atau tugas Amazon ECS mandiri yang dikelola oleh layanan ECS. Mengonfigurasi volume saat penerapan memungkinkan Anda membuat definisi tugas yang dapat dilanjutkan yang tidak dibatasi untuk jenis atau pengaturan volume tertentu. Volume Amazon EBS menyediakan penyimpanan blok berkinerja tinggi yang sangat tersedia, hemat biaya, tahan lama, dan berkinerja tinggi untuk beban kerja kontainer intensif data.	Januari 11, 2024
Konsol klasik Amazon ECS mencapai akhir masa pakai	Konsol Amazon ECS telah mencapai akhir masa pakai.	Desember 4, 2023
Kebijakan yang diperbarui	Kebijakan IAM ECSService RolePolicy terkelola Amazon diperbarui dengan `events` izin baru `autoscaling` dan `autoscaling-plans` tambahan serta izin.	Desember 4, 2023
Dukungan Runtime Monitoring	Anda dapat menggunakan Runtime Monitoring untuk memantau beban kerja Amazon ECS untuk mengident ifikasi perilaku berbahaya atau tidak sah. Untuk informasi selengkapnya, lihat Runtime Monitoring.	26 November 2023
Kebijakan yang diperbarui	Kebijakan IAM `AmazonECSServiceRo lePolicy` terkelola telah diperbarui untuk mengizinkan akses ke AWS Cloud Map `DiscoverI nstancesRevision` API.	4 Oktober 2023
AWS Fargate konfigurasi tugas pensiun	Anda dapat mengonfigurasi periode tunggu sebelum tugas Fargate dihentikan Untuk informasi selengkap nya, lihat AWS Fargate pemeliharaan tugas.	5 September 2023

Perubahan	Deskripsi	Tanggal
Parameter definisi tugas tambahan di AWS Fargate	AWS Fargate menambahkan dukungan untuk `pidMode` dan `systemControls` dalam versi platform Linux`1.4.0`. Untuk informasi selengkapnya, lihat Definisi tugas.	9 Agustus 2023
Desain ulang halaman definisi tugas konsol Amazon ECS	Halaman definisi tugas di konsol Amazon ECS telah didesain ulang dan berisi opsi tambahan. Untuk informasi selengkapnya, lihat Membuat definisi tugas menggunakan konsol.	26 Juli 2023
Fargate mendukung pemuatan lambat dengan indeks OCI Seekable	AWS Fargate memperkenalkan indeks Seekable OCI (SOCI). Dengan SOCI, kontainer hanya menghabis kan beberapa detik pada penarikan gambar sebelum mereka dapat memulai, menyediakan waktu untuk pengaturan lingkungan dan instantiasi aplikasi saat gambar diunduh di latar belakang. Untuk informasi selengkapnya, lihat Lazy loading gambar kontainer menggunakan Seekable OCI (SOCI).	Juli 17, 2023
Peningkatan dukungan untuk GMSA di Linux dan Windows	Definisi tugas memiliki `credentialSpecs` bidang baru untuk GMSA untuk Linux dan Windows. Tutorial lengkap baru untuk GMSA tanpa domain di Windows telah ditambahkan, lihat Tutorial: Menggunakan Wadah Windows dengan GMSA Tanpa Domain menggunakan file. AWS CLI Untuk informasi selengkapnya, lihat Menggunakan g MSAs untuk Kontainer Linux dan Menggunakan g MSAs untuk Kontainer Windows.	14 Juli 2023

Perubahan	Deskripsi	Tanggal
Peningkatan versi ECS Agent dokumentasi	Dokumentasi untuk versi Amazon ECS Agent telah diperbarui. Kami menyarankan Anda menggunakan `v20.10.13` versi atau yang lebih baru dari Docker dengan versi terbaru dari agen penampung Amazon ECS. Versi yang dirilis dan perubahan pada agen tersedia di GitHub. Untuk informasi selengkapnya, lihat versi agen penampung Amazon ECS Linux.	20 Juni 2023
Ketersediaan Wilayah yang diperbarui untuk dukungan Fargate ARM64	Ketersediaan Wilayah untuk ARM64 dukungan Fargate telah diperbarui. Untuk informasi selengkap nya, lihat Pertimbangan.	19 Juni 2023
Tingkatkan dokumenta si penskalaan otomatis cluster	Dokumentasi untuk penskalaan Amazon ECS dari Amazon Auto EC2 Scaling memiliki peningkatan yang signifikan dalam akurasi dan keterbacaan. Untuk informasi selengkapnya, lihat penskalaan otomatis kluster Amazon ECS.	4 Mei 2023
Menandai otorisasi untuk pembuatan sumber daya.	Pengguna harus memiliki izin untuk tindakan yang membuat sumber daya, seperti`ecsCreateCluster` . Saat Anda membuat sumber daya dan menentuka n tag untuk sumber daya tersebut, AWS lakukan otorisasi tambahan untuk memverifikasi bahwa ada izin untuk membuat tag. Untuk informasi selengkap nya, lihat Otorisasi penandaan dan Memberikan izin untuk menandai sumber daya pada pembuatan.	18 April 2023
Support untuk GMSA untuk kontainer Linux di EC2	Anda dapat menggunakan GMSA untuk mengauten tikasi ke Active Directory untuk wadah Linux. EC2 Untuk informasi selengkapnya, lihat Menggunakan g MSAs untuk Kontainer Linux.	April 14, 2023

Perubahan	Deskripsi	Tanggal
Support untuk penyimpanan sementara untuk wadah Windows aktif AWS Fargate	Anda dapat menggunakan penyimpanan sementara untuk wadah Windows aktif. AWS Fargate Untuk informasi selengkapnya, lihat Penyimpanan tugas Fargate.	April 14, 2023
AWS Cost Management dukungan untuk data CUR tingkat tugas	Anda dapat mengaktifkan biaya tingkat tugas dan penggunaan sumber daya di Laporan Biaya dan Penggunaan. Ini menambahkan Data Alokasi Biaya Split untuk tugas-tugas yang berjalan di AWS EC2 Fargate dan. Untuk informasi selengkapnya, lihat Laporan Biaya dan Penggunaan Tingkat Tugas.	12 April 2023
Amazon Linux 2023 Amazon ECS yang dioptimalkan AMI	Anda dapat menerapkan beban kerja di AMI Amazon Linux 2023 Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	10 April 2023
AWS Fargate Standar Pemrosesan Informasi Federal (FIPS) 140	Anda dapat menerapkan beban kerja di AWS Fargate Amazon ECS dengan cara yang sesuai dengan Federal Information Processing Standard (FIPS) 140. Untuk informasi selengkapnya, lihat AWS Fargate Standar Pemrosesan Informasi Federal (FIPS-140).	10 April 2023
Penghapusan definisi tugas	Anda dapat menghapus definisi tugas menggunakan konsol Amazon ECS, SDK, dan. AWS CLI Untuk informasi selengkapnya, lihat Menghapus revisi definisi tugas menggunakan konsol dan definisi Tugas.	Februari 24, 2023
AWS Fargate rekomendasi layanan di Compute Optimizer	AWS Compute Optimizer menghasilkan rekomendasi ukuran tugas dan wadah berdasarkan pemanfaatan tugas yang sedang berjalan di layanan Amazon ECS di Fargate. AWS Untuk informasi selengkapnya, lihat Melihat rekomendasi untuk layanan Amazon ECS di Fargate.	27 Januari 2023

Perubahan	Deskripsi	Tanggal
Konsol Amazon ECS	Konsol Amazon ECS yang baru sekarang menjadi konsol default. Untuk informasi selengkapnya, lihat Konsol Amazon ECS baru.	19 Januari 2023
Kebijakan `AmazonECS_FullAccess` IAM yang diperbarui	Kebijakan `AmazonECS_FullAccess` IAM diperbarui untuk menyertakan izin untuk menambahkan tag ke penyeimbang beban selama pembuatan. Untuk informasi selengkapnya, lihat Amazonecs_ FullAccess.	4 Januari 2023
Gunakan CloudWatch alarm untuk mendeteksi kegagalan penerapan layanan Amazon ECS	Anda dapat mengonfigurasi Amazon ECS untuk menyetel penerapan menjadi gagal saat mendeteksi bahwa CloudWatch alarm tertentu telah masuk ke status ALARM. Untuk informasi selengkapnya, lihat the section called “Penerapan pembaruan bergulir”.	19 Desember 2022
Support untuk pemetaan port kontainer	Anda dapat mengatur rentang nomor port pada wadah yang terikat ke rentang port host yang dipetakan secara dinamis. Untuk informasi selengkapnya, lihat the section called “Pemetaan pelabuhan”.	15 Desember 2022
Ketersediaan umum Amazon ECS Service Connect	Fitur ini menambahkan penemuan layanan dan mesh layanan yang dikendalikan oleh penyebaran layanan Amazon ECS. Untuk informasi selengkapnya, lihat the section called “ Layanan Connect”.	27 November 2022
Pengalaman konsol Amazon ECS baru untuk definisi tugas diperbarui	Pengalaman konsol Amazon ECS yang baru sekarang berisi editor JSON untuk definisi tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.	27 Oktober 2022
Pengalaman konsol Amazon ECS baru untuk definisi tugas diperbarui	Pengalaman konsol Amazon ECS yang baru sekarang berisi editor JSON untuk definisi tugas. Untuk informasi selengkapnya, lihat the section called “Membuat definisi tugas menggunakan konsol”.	27 Oktober 2022

Perubahan	Deskripsi	Tanggal
Pengalaman konsol Amazon ECS yang baru diperbarui	Pengalaman konsol Amazon ECS yang baru telah diperbarui dengan parameter layanan dan tugas tambahan. Untuk informasi selengkapnya, lihat the section called “Membuat penerapan pembaruan bergulir” dan the section called “Menjalankan aplikasi sebagai tugas”.	Oktober 7, 2022
Informasi baru di titik akhir metadata tugas versi 4	Titik akhir metadata tugas versi 4 sekarang menyertakan ID VPC dan nama layanan. Untuk informasi selengkapnya, lihat the section called “Titik akhir metadata tugas versi 4”.	Oktober 7, 2022
Ukuran definisi tugas baru	Amazon ECS di Fargate sekarang mendukung ukuran tugas 8 vCPU dan 16 vCPU. Untuk informasi selengkapnya, lihat the section called “Ukuran tugas”.	September 16, 2022
Halaman ECS CLI diarsipkan	Dokumentasi CLI ECS telah diarsipkan. Kami merekomendasikan menggunakan AWS Copilot untuk kebutuhan alat baris perintah Anda. Untuk informasi selengkapnya, lihat Membuat sumber daya Amazon ECS menggunakan antarmuka baris AWS perintah Copilot.	15 September 2022
Kuota Fargate baru	Fargate sedang beralih dari kuota berbasis hitungan tugas ke kuota berbasis VCPU. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon ECS.	8 September 2022
Dukungan untuk kolam hangat Amazon EC2 Auto Scaling.	Anda sekarang dapat menggunakan kolam hangat Amazon EC2 Auto Scaling untuk meningkatkan skala aplikasi Anda lebih cepat dan menghemat biaya. Untuk informasi selengkapnya, lihat Mengonfigurasi instans pra-inisialisasi untuk grup Auto Scaling Amazon ECS Anda.	Maret 23, 2022

Perubahan	Deskripsi	Tanggal
Support untuk instance Windows di ECS Anywhere.	ECS Anywhere sekarang mendukung instance Windows. Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk jenis peluncuran eksternal.	3 Maret 2022
Menambahkan dukungan ECS Exec untuk instans eksternal	ECS Exec sekarang didukung untuk instans eksternal . Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	24 Januari 2022
Pengalaman konsol Amazon ECS baru diperbarui	Pengalaman konsol Amazon ECS yang baru mendukung pembuatan dan penghapusan klaster, memperbarui definisi tugas, dan membatalkan pendaftaran definisi tugas. Untuk informasi lebih lanjut, lihat Membuat cluster Amazon ECS untuk jenis peluncuran Fargate, Menghapus kluster Amazon ECS, Memperbarui definisi tugas Amazon ECS menggunakan konsol, dan Membatalkan pendaftaran revisi definisi tugas Amazon ECS menggunakan konsol .	8 Desember 2021
Pengalaman konsol Amazon ECS baru diperbarui	Pengalaman konsol Amazon ECS yang baru mendukung pembuatan definisi tugas. Untuk informasi selengkapnya, lihat Membuat definisi tugas Amazon ECS menggunakan konsol.	23 November 2021
Amazon ECS mendukung arsitektur ARM 64-bit untuk Linux.	Amazon ECS mendukung arsitektur CPU ARM 64-bit untuk sistem operasi Linux. Untuk informasi selengkapnya, lihat the section called “Definisi tugas untuk beban kerja ARM 64-bit”.	23 November 2021
Dukungan Amazon ECS untuk opsi lancar log-driver-buffer-limit	Amazon ECS mendukung opsi lancar`log-driver-buffer-limit` . Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.	22 November 2021

Perubahan	Deskripsi	Tanggal
Skrip pembuatan AMI Linux AMI yang dioptimal kan Amazon ECS	Amazon ECS telah membuka skrip build yang digunakan untuk membangun varian Linux dari AMI Amazon ECS yang dioptimalkan. Untuk informasi selengkapnya, lihat Skrip pembuatan AMI Linux AMI yang dioptimalkan Amazon ECS.	November 19, 2021
Kesehatan contoh kontainer	Amazon ECS menambahkan dukungan untuk pemantauan kesehatan instans kontainer. Untuk informasi selengkapnya, lihat Pantau kesehatan instans kontainer Amazon ECS.	November 10, 2021
Dukungan untuk Windows Amazon ECS Exec	Amazon ECS Exec mendukung Windows. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	November 1, 2021
Support untuk wadah Windows di Fargate.	Amazon ECS mendukung wadah Windows di Fargate. Untuk informasi selengkapnya, lihat Log perubahan versi platform Windows Fargate.	28 Oktober 2021
Dukungan GPU untuk instans eksternal di Amazon ECS Anywhere	Amazon ECS mendukung penetapan persyaratan GPU dalam definisi tugas untuk tugas yang dijalankan pada instans eksternal. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU dan Mendaftarkan instans eksternal ke kluster Amazon ECS.	Oktober 8, 2021
Dukungan dari mode jaringan `awsvpc` di Windows	Amazon ECS mendukung mode `awsvpc` jaringan pada Windows. Untuk informasi selengkapnya, lihat Alokasikan antarmuka jaringan untuk tugas Amazon ECS.	15 Juli 2021
Bottlerocket yang umumnya tersedia	Amazon ECS mendukung varian AMI Amazon ECS yang dioptimalkan dari sistem operasi Bottlerocket disediakan sebagai AMI. Untuk informasi selengkap nya, lihat Amazon ECS dioptimalkan Bottlerocket AMIs.	30 Juni 2021

Perubahan	Deskripsi	Tanggal
Pembaruan tugas terjadwal Amazon ECS	Amazon EventBridge menambahkan dukungan untuk parameter tambahan saat membuat aturan yang memicu tugas terjadwal Amazon ECS.	25 Juni 2021
AWS kebijakan terkelola untuk Amazon ECS	Amazon ECS menambahkan dokumentasi kebijakan AWS terkelola untuk peran terkait layanan. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Service.	8 Juni 2021
Memulai dengan AWS CDK	Menambahkan panduan memulai untuk menggunak an AWS CDK dengan Amazon ECS. Untuk informasi selengkapnya, lihat Membuat sumber daya Amazon ECS menggunakan AWS CDK.	27 Mei 2021
Amazon ECS Anywhere	Amazon ECS telah menambahkan dukungan untuk mendaftarkan server on-premise atau mesin virtual (VM) dengan cluster Anda. Untuk informasi selengkap nya, lihat Cluster Amazon ECS untuk jenis peluncuran eksternal.	25 Mei 2021
Amazon ECS yang dioptimalkan Windows Server 20H2 Core AMI	Amazon ECS telah menambahkan dukungan untuk varian AMI Windows Amazon ECS baru yang dioptimal kan untuk Windows Server 20H2 Core. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	19 April 2021
Amazon ECS Exec	Amazon ECS telah merilis alat debugging baru yang disebut ECS Exec. Untuk informasi selengkapnya, lihat Pantau wadah Amazon ECS dengan ECS Exec.	15 Maret 2021
Dukungan kebijakan VPC endpoint	Amazon ECS sekarang mendukung kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat Membuat kebijakan titik akhir VPC untuk Amazon ECS.	11 Januari 2021

Perubahan	Deskripsi	Tanggal
Pengalaman konsol baru	Amazon ECS telah merilis pengalaman konsol baru yang mendukung pembuatan atau pembaruan layanan atau menjalankan tugas mandiri. Untuk informasi selengkapnya, lihat Membuat penyebaran pembaruan bergulir Amazon ECS dan Menjalankan aplikasi sebagai tugas Amazon ECS.	28 Desember 2020
Pembaruan penyedia kapasitas	Amazon ECS menambahkan dukungan untuk memperbarui penyedia kapasitas grup Auto Scaling yang ada.	23 November 2020
ECS sekarang mendukung Amazon FSx untuk Windows File Server untuk tugas Windows	Amazon ECS menambahkan dukungan untuk menentukan Amazon FSx untuk volume Windows File Server dalam definisi tugas Windows. Untuk informasi selengkapnya, lihat Gunakan FSx untuk volume Server File Windows dengan Amazon ECS.	11 November 2020
Dukungan mode tumpukan dobel VPC ditambahkan	Amazon ECS menambahkan dukungan untuk menggunakan VPC dalam mode dual-stack dengan tugas menggunakan mode jaringan, `awsvpc` yang menyediakan dukungan untuk alamat. IPv6 Untuk informasi selengkapnya, lihat Menggunakan VPC dalam mode tumpukan dobel.	5 November 2020
Pembaruan v4 titik akhir metadata tugas	Amazon ECS menambahkan metadata tambahan ke output v4 titik akhir metadata tugas. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	5 November 2020
Dukungan untuk Local Zones dan Zona Wavelength	Amazon ECS menambahkan dukungan untuk beban kerja di Local Zones dan Wavelength Zones. Untuk informasi selengkapnya, lihat Aplikasi Amazon ECS di subnet bersama, Local Zones, dan Wavelength Zones.	Selasa, 04 September 2020

Perubahan	Deskripsi	Tanggal
Varian Amazon ECS dari Bottlerocket AMI	Bottlerocket adalah sistem operasi open source berbasis Linux yang dibuat khusus untuk menjalank an kontainer. AWS Varian AMI yang dioptimal kan Amazon ECS dari sistem operasi Bottlerocket disediakan sebagai AMI yang dapat Anda gunakan saat meluncurkan instans kontainer Amazon ECS. Untuk informasi selengkapnya, lihat Amazon ECS dioptimalkan Bottlerocket AMIs.	31 Agustus 2020
Titik akhir metadata tugas versi 4 diperbarui untuk statistik tingkat jaringan	Titik akhir metadata tugas versi 4 telah diperbarui untuk menyediakan statistik laju jaringan untuk tugas Amazon ECS yang menggunakan `awsvpc` mode `bridge` atau jaringan yang dihosting di instans EC2 Amazon yang menjalankan setidaknya `1.43.0` versi agen penampung. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	10 Agustus 2020
Metrik penggunaan Fargate	AWS Fargate menyediakan metrik CloudWatch penggunaan yang memberikan visibilitas ke akun Anda, penggunaan sumber daya Fargate On-Demand dan Fargate Spot. Untuk informasi selengkapnya, lihat Metrik penggunaan.	3 Agustus 2020
AWS Copilot versi 0.1.0	AWS Copilot CLI baru diluncurkan, menyediakan perintah tingkat tinggi untuk menyederhanakan pemodelan, pembuatan, rilis, dan pengelolaan aplikasi kontainer di Amazon ECS dari lingkungan pengembangan lokal. Untuk informasi selengkapnya, lihat Membuat sumber daya Amazon ECS menggunak an antarmuka baris AWS perintah Copilot.	9 Juli 2020
AWS Fargate jadwal penghentian versi platform	Jadwal penghentian versi platform Fargate telah ditambahkan. Untuk informasi selengkapnya, lihat AWS Penghentian versi platform Fargate Linux.	8 Juli 2020

Perubahan	Deskripsi	Tanggal
AWS Perluasan Wilayah Fargate	Amazon ECS di AWS Fargate telah diperluas ke Wilayah Eropa (Milan).	25 Juni 2020
Amazon ECS dioptimal kan Amazon Linux 2 (Neuron) AMI dirilis	Amazon ECS merilis Amazon ECS yang dioptimal kan Amazon Linux 2 (Neuron) AMI untuk beban kerja inferensial. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	24 Juni 2020
Menambahkan dukungan untuk menghapus penyedia kapasitas	Amazon ECS menambahkan dukungan untuk menghapus penyedia kapasitas grup Auto Scaling.	11 Juni 2020
AWS Pembaruan platform Fargate versi 1.4.0	Mulai 28 Mei 2020, setiap tugas Fargate baru yang diluncurkan menggunakan platform versi 1.4.0 akan memiliki penyimpanan singkat 20 GB yang dienkrips i dengan algoritme enkripsi AES-256 menggunak an kunci enkripsi yang dikelola. AWS Fargate Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate untuk Amazon ECS.	28 Mei 2020
Dukungan file variabel lingkungan	Menambahkan dukungan untuk menentukan file variabel lingkungan dalam ketentuan tugas, yang memungkinkan Anda untuk menambahkan banyak variabel lingkungan ke kontainer Anda. Untuk informasi selengkapnya, lihat Meneruskan variabel lingkungan individual ke wadah Amazon ECS.	18 Mei 2020
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah diperluas ke Wilayah Afrika (Cape Town).	11 Mei 2020

Perubahan	Deskripsi	Tanggal
Service Quotas diperbarui	Service Quotas berikut telah diperbarui: • Klaster per akun ditingkatkan dari `2,000` ke `10,000`. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon ECS.	17 April 2020

Perubahan	Deskripsi	Tanggal
AWS Platform Fargate versi 1.4.0	AWS Platform Fargate versi 1.4.0 dirilis, yang berisi fitur-fitur berikut: • Menambahkan dukungan untuk menggunakan volume sistem file Amazon EFS untuk penyimpan an tugas persisten. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS. • Penyimpanan tugas sementara telah ditingkatkan menjadi 20 GB. Untuk informasi selengkapnya, lihat Penyimpanan sementara tugas Fargate untuk Amazon ECS. • Perilaku lalu lintas jaringan ke dan dari tugas telah diperbarui. Dimulai dengan platform versi 1.4, semua tugas Fargate menerima satu elastic network interface (disebut sebagai tugas ENI) dan semua lalu lintas jaringan mengalir melalui ENI tersebut dalam VPC Anda dan akan terlihat oleh Anda melalui log aliran VPC Anda. Untuk informasi selengkapnya, lihat opsi jaringan tugas Amazon ECS untuk jenis peluncuran Fargate. • Tugas ENIs menambahkan dukungan untuk bingkai jumbo. Antarmuka jaringan dikonfigurasi dengan unit transmisi maksimum (MTU), yang merupakan ukuran muatan terbesar yang muat dalam satu frame. Semakin besar MTU, semakin banyak muatan aplikasi yang termuat dalam satu frame, yang mengurangi overhead per frame dan meningkatkan efisiensi. Dengan mendukung jumbo frame akan dapat mengurangi overhead ketika jalur jaringan antara tugas dengan tujuan Anda	8 April 2020

Perubahan	Deskripsi	Tanggal
	mendukung jumbo frame, seperti semua lalu lintas yang tetap berada dalam VPC Anda. • CloudWatch Container Insights akan menyertakan metrik kinerja jaringan untuk tugas Fargate. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan. • Menambahkan dukungan untuk titik akhir metadata tugas v4 yang menyediakan informasi tambahan untuk tugas Fargate Anda, termasuk statistik jaringan untuk tugas dan Zona Ketersediaan mana tugas sedang berjalan. Untuk informasi selengkap nya, lihat Titik akhir metadata tugas Amazon ECS versi 4. • Menambahkan dukungan untuk parameter Linux `SYS_PTRACE` dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Parameter Linux. • Agen kontainer Fargate menggantikan penggunaa n agen kontainer Amazon ECS untuk semua tugas Fargate. Perubahan ini seharusnya tidak berpengar uh pada cara tugas Anda berjalan. • Waktu aktif kontainer kini menggunakan Container d bukan Docker. Perubahan ini seharusnya tidak berpengaruh pada cara tugas Anda berjalan. Anda akan melihat bahwa beberapa pesan kesalahan yang berasal dari waktu aktif kontainer akan berubah dari menyebutkan Docker menjadi kesalahan yang lebih umum.	

Perubahan	Deskripsi	Tanggal
	Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.	
Dukungan sistem file Amazon EFS untuk volume tugas	Sistem file Amazon EFS dapat digunakan sebagai volume data untuk tugas Amazon ECS dan Fargate Anda. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.	8 April 2020
Titik Akhir Metadata Tugas Amazon ECS versi 4	Dimulai dengan agen kontainer Amazon ECS versi 1.39.0 dan platform Fargate versi 1.4.0, variabel lingkungan `ECS_CONTAINER_METADATA_URI_V4` bernama disuntikkan ke setiap wadah dalam suatu tugas. Saat Anda melakukan kueri terhadap titik akhir metadata tugas versi 4, berbagai metadata tugas dan statistik Docker tersedia untuk tugas. Untuk informasi selengkapnya, lihat Titik akhir metadata tugas Amazon ECS versi 4.	8 April 2020
Support untuk versi spesifik rahasia Secrets Manager untuk disuntikkan sebagai variabel lingkungan	Menambahkan dukungan untuk menentukan data sensitif menggunakan versi rahasia Secrets Manager tertentu. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.	24 Februari 2020
Menambahkan opsi konfigurasi CodeDeploy penerapan tambahan untuk penerapan biru/hijau	CodeDeploy Layanan ini menambahkan konfigurasi penerapan canary dan linier baru untuk jenis penyebaran Amazon ECS. Kemampuan untuk menentukan konfigurasi deployment kustom juga tersedia. Untuk informasi selengkapnya, lihat CodeDeploy penerapan biru/hijau untuk Amazon ECS.	6 Februari 2020

Perubahan	Deskripsi	Tanggal
Ditambahkan parameter definisi efsVolume Configuration tugas	Parameter definisi `efsVolumeConfiguration` tugas ada di pratinjau publik, yang memudahkan penggunaan sistem file Amazon EFS dengan tugas Amazon ECS Anda. Untuk informasi selengkapnya, lihat Gunakan volume Amazon EFS dengan Amazon ECS.	17 Januari 2020
Perilaku pencatatan agen penampung Amazon ECS diperbarui	Lokasi pencatatan agen kontainer Amazon ECS dan perilaku rotasi telah diperbarui. Untuk informasi selengkapnya, lihat Parameter konfigurasi log agen kontainer Amazon ECS.	13 Januari 2020
Spot Fargate	Amazon ECS menambahkan dukungan untuk menjalankan tugas menggunakan Fargate Spot. Untuk informasi selengkapnya, lihat Cluster Amazon ECS untuk Fargate.	3 Desember 2019
Auto Scaling Klaster	Penskalaan otomatis klaster Amazon ECS memungkinkan Anda memiliki kontrol lebih besar atas cara Anda menskalakan tugas dalam klaster. Untuk informasi selengkapnya, lihat Secara otomatis mengelola kapasitas Amazon ECS dengan penskalaan otomatis cluster.	3 Desember 2019
Penyedia Kapasitas Klaster	Penyedia kapasitas cluster Amazon ECS menentukan infrastruktur yang akan digunakan untuk tugas Anda. Untuk informasi selengkapnya, lihat Klaster-klaster Amazon ECS.	3 Desember 2019
Membuat cluster pada AWS Outposts	Amazon ECS sekarang mendukung pembuatan cluster di file. AWS Outposts Untuk informasi selengkapnya, lihat the section called “Layanan Kontainer Elastis Amazon aktif AWS Outposts”.	3 Desember 2019

Perubahan	Deskripsi	Tanggal
Kejadian Tindakan Layanan	Amazon ECS sekarang mengirimkan peristiwa ke Amazon EventBridge ketika tindakan layanan tertentu terjadi. Untuk informasi selengkapnya, lihat Acara tindakan layanan Amazon ECS.	25 November 2019
Amazon ECS GPU AMI yang Dioptimalkan Mendukung Instans G4	Amazon ECS menambahkan dukungan untuk keluarga tipe instans g4 saat menggunakan AMI yang dioptimal kan untuk GPU Amazon ECS. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.	8 Oktober 2019
FireLens untuk Amazon ECS	FireLens untuk Amazon ECS dalam ketersediaan umum. FireLens Amazon ECS memungkinkan Anda menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan mitra untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.	30 September 2019
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkemban g ke wilayah Eropa (Paris), Eropa (Stockholm), dan Timur Tengah (Bahrain).	30 September 2019
Deep Learning Container s dengan Elastic Inference di Amazon ECS	Amazon ECS mendukung pemasangan akselerat or Amazon Elastic Inference ke container Anda untuk membuat menjalankan beban kerja inferensi pembelajaran mendalam menjadi lebih efisien.	3 September 2019
FireLens untuk Amazon ECS	FireLens untuk Amazon ECS ada di pratinjau publik. FireLens Amazon ECS memungkinkan Anda menggunakan parameter definisi tugas untuk merutekan log ke AWS layanan atau tujuan mitra untuk penyimpanan log dan analitik. Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke AWS layanan atau AWS Partner.	30 Agustus 2019

Perubahan	Deskripsi	Tanggal
CloudWatch Wawasan Kontainer	CloudWatch Wawasan Kontainer sekarang tersedia secara umum. Aplikasi ini memungkinkan Anda untuk mengumpulkan, menggabungkan, serta merangkum metrik dan log dari aplikasi dan layanan mikro terkontai nerisasi Anda. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.	30 Agustus 2019
Konfigurasi Pertukaran Tingkat Kontainer	Amazon ECS menambahkan dukungan untuk mengontrol penggunaan ruang memori swap pada instans penampung Linux Anda di tingkat penampung . Dengan menggunakan konfigurasi pertukaran per kontainer, setiap kontainer dalam ketentuan tugas dapat mengaktifkan atau menonaktifkan pertukaran, dan untuk yang diaktifkan, jumlah maksimum ruang tukar yang digunakan dapat dibatasi. Untuk informasi selengkapnya, lihat Mengelola ruang memori swap kontainer di Amazon ECS.	16 Agustus 2019
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Asia Pasifik (Hong Kong).	6 Agustus 2019
Trunking Antarmuka Jaringan Elastis	Menambahkan jenis EC2 instans Amazon tambahan yang didukung untuk fitur trunking ENI. Untuk informasi selengkapnya, lihat Instans yang didukung untuk peningkatan antarmuka jaringan kontainer Amazon ECS.	1 Agustus 2019
Mendaftarkan Beberapa Grup Target dengan sebuah Layanan	Menambahkan dukungan untuk menentukan beberapa grup target dalam sebuah definisi layanan. Untuk informasi selengkapnya, lihat Mendaftarkan beberapa grup sasaran dengan layanan Amazon ECS.	30 Juli 2019

Perubahan	Deskripsi	Tanggal
Menentukan Data Sensitif Menggunakan Rahasia Secrets Manager	Ditambahkan tutorial untuk menentukan data sensitif menggunakan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Menentukan data sensitif menggunakan rahasia Secrets Manager di Amazon ECS.	Juli 20, 2019
CloudWatch Wawasan Kontainer	Amazon ECS telah menambahkan dukungan untuk CloudWatch Container Insights. Untuk informasi selengkapnya, lihat Pantau kontainer Amazon ECS menggunakan Container Insights dengan kemampuan observasi yang ditingkatkan.	9 Juli 2019
Izin tingkat sumber daya untuk layanan dan set tugas Amazon ECS	Amazon ECS telah memperluas dukungan izin tingkat sumber daya untuk layanan dan tugas Amazon ECS. Untuk informasi selengkapnya, lihat Bagaimana Amazon Elastic Container Service bekerja dengan IAM.	27 Juni 2019
AMI Amazon ECS baru yang dioptimalkan ditambal untuk -2019-005 AWS	Amazon ECS telah memperbarui AMI yang dioptimal kan Amazon ECS untuk mengatasi kerentanan yang dijelaskan dalam -2019-005.AWS	17 Juni 2019
Trunking Antarmuka Jaringan Elastis	Amazon ECS memperkenalkan dukungan untuk meluncurkan instans kontainer menggunakan jenis EC2 instans Amazon yang didukung yang telah meningkatkan kepadatan elastic network interface (ENI). Dengan menggunakan tipe instans ini dan ikut serta dalam pengaturan akun `awsvpcTrunking` akan meningkatkan kepadatan ENI pada instans kontainer yang baru diluncurkan yang mengizinkan Anda untuk menempatkan lebih banyak tugas pada setiap instans kontainer. Untuk informasi selengkap nya, lihat Meningkatkan antarmuka jaringan instans kontainer Amazon ECS Linux.	6 Juni 2019

Perubahan	Deskripsi	Tanggal
AWS Pembaruan platform Fargate versi 1.3.0	Mulai 1 Mei 2019, setiap tugas Fargate baru yang diluncurkan mendukung driver `splunk` log selain driver `awslogs` log. Untuk informasi selengkapnya, lihat Penyimpanan dan pencatatan.	1 Mei 2019
AWS Pembaruan platform Fargate versi 1.3.0	Mulai 1 Mei 2019, tugas Fargate baru apa pun yang diluncurkan mendukung referensi data sensitif dalam konfigurasi log wadah menggunakan parameter definisi `secretOptions` wadah. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.	1 Mei 2019
AWS Pembaruan platform Fargate versi 1.3.0	Dimulai pada 2 April 2019, tugas Fargate baru apa pun yang diluncurkan mendukung penyuntikan data sensitif ke dalam wadah Anda dengan menyimpan data sensitif Anda baik dalam AWS Secrets Manager rahasia atau AWS Systems Manager parameter Parameter Store dan kemudian merujuknya dalam definisi penampung Anda. Untuk informasi selengkap nya, lihat Meneruskan data sensitif ke wadah Amazon ECS.	April 2, 2019
AWS Pembaruan platform Fargate versi 1.3.0	Mulai tanggal 27 Maret 2019, setiap tugas Fargate baru yang diluncurkan dapat menggunakan parameter definisi tugas tambahan yang memungkinkan Anda menentukan konfigurasi proxy, dependensi untuk startup dan shutdown kontainer serta nilai batas waktu mulai dan berhenti per kontainer. Lihat informasi selengkapnya di Konfigurasi proxy, Dependensi kontainer, dan Waktu habis kontainer.	27 Maret 2019

Perubahan	Deskripsi	Tanggal
Amazon ECS memperkenalkan tipe penerapan eksternal	Jenis penerapan eksternal memungkinkan Anda menggunakan pengontrol penyebaran pihak ketiga untuk kontrol penuh atas proses penyebaran untuk layanan Amazon ECS. Untuk informasi selengkapnya, lihat Menyebarkan layanan Amazon ECS menggunakan pengontrol pihak ketiga.	27 Maret 2019
AWS Deep Learning Containers di Amazon ECS	AWS Deep Learning Containers adalah sekumpulan gambar Docker untuk melatih dan menyajikan model TensorFlow di Amazon Elastic Container Service (Amazon ECS). Deep Learning Containers menyediakan lingkungan yang dioptimalkan dengan TensorFlow library Nvidia CUDA (untuk instans GPU), dan Intel MKL (untuk instans CPU) dan tersedia di Amazon ECR. Untuk informasi selengkapnya, lihat Menggunakan AWS Deep Learning Containers di Amazon ECS.	27 Maret 2019
Amazon ECS memperkenalkan manajemen ketergantungan kontainer yang disempurnakan	Amazon ECS memperkenalkan parameter definisi tugas tambahan yang memungkinkan Anda menentukan dependensi untuk startup dan shutdown container serta nilai batas waktu mulai dan berhenti per kontainer. Untuk informasi selengkapnya, lihat Dependensi kontainer.	7 Maret 2019
Amazon ECS memperkenalkan API `PutAccountSettingDefault`	Amazon ECS memperkenalkan `PutAccountSettingDefault` API yang memungkinkan pengguna untuk mengatur status pilih ARN/ID format default untuk semua pengguna dan peran di akun. Sebelumnya, pengaturan status keikutsertaan default akun mengharuskan penggunaan pemilik akun. Untuk informasi selengkapnya, lihat Nama Sumber Daya Amazon (ARNs) dan IDs.	8 Februari 2019

Perubahan	Deskripsi	Tanggal
Amazon ECS mendukung beban kerja GPU	Amazon ECS memperkenalkan dukungan untuk beban kerja GPU dengan memungkinkan Anda membuat cluster dengan instans kontainer berkemampuan GPU. Dalam definisi tugas Anda dapat menentukan jumlah yang diperlukan GPUs dan agen ECS akan menyematkan fisik GPUs ke wadah. Untuk informasi selengkapnya, lihat Definisi tugas Amazon ECS untuk beban kerja GPU.	4 Februari 2019
Amazon ECS memperluas dukungan rahasia	Amazon ECS memperluas dukungan untuk menggunakan AWS Secrets Manager rahasia secara langsung dalam definisi tugas Anda untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.	21 Januari 2019
VPC endpoint Antarmuka (AWS PrivateLink)	Menambahkan dukungan untuk mengonfigurasi VPC endpoint antarmuka yang didukung oleh AWS PrivateLink. Ini memungkinkan Anda untuk membuat koneksi pribadi antara VPC Anda dan Amazon ECS tanpa memerlukan akses melalui Internet, melalui instans NAT, koneksi VPN, atau. AWS Direct Connect Untuk informasi selengkapnya, lihat VPC Endpoint Antarmuka (AWS PrivateLink).	Desember 26, 2018

Perubahan	Deskripsi	Tanggal
AWS Platform Fargate versi 1.3.0	Versi platform AWS Fargate baru dirilis, yang berisi: • Menambahkan dukungan untuk menggunakan AWS Systems Manager parameter Parameter Store untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS. • Menambahkan daur ulang tugas untuk tugas Fargate, yang merupakan proses menyegarkan tugas yang merupakan bagian dari layanan Amazon ECS. Untuk informasi selengkapnya, lihat Pensiun tugas dan pemeliharaan AWS Fargate di Amazon ECS. Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.	17 Desember 2018
Kuota layanan diperbarui	Kuota layanan berikut diperbarui: • Jumlah klaster per Wilayah, per akun ditingkatkan dari `1000` ke `2000`. • Jumlah instans kontainer per klaster ditingkatkan dari `1000` ke `2000`. • Jumlah layanan per klaster ditingkatkan dari `500` ke `1000`. Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECS.	14 Desember 2018

Perubahan	Deskripsi	Tanggal
AWS Perluasan wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Asia Pasifik (Mumbai) dan Kanada (Tengah). Untuk informasi selengkapnya, lihat Wilayah yang Didukung untuk Amazon ECS di Fargate AWS.	7 Desember 2018
Penerapan Amazon ECS blue/green	Amazon ECS menambahkan dukungan untuk blue/green penerapan yang digunakan. CodeDeploy Tipe deployment ini mengizinkan Anda untuk memverifikasi deployment layanan baru sebelum mengirim lalu lintas produksi ke dalamnya. Untuk informasi selengkapnya, lihat CodeDeploy penerapan biru/hijau untuk Amazon ECS.	27 November 2018
Amazon ECS yang dioptimalkan Amazon Linux 2 (arm64) AMI dirilis	Amazon ECS merilis Amazon Linux AMIs 2 yang dioptimalkan Amazon ECS untuk arsitektur arm64. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	26 November 2018
Menambahkan dukungan untuk flag Docker tambahan dalam ketentuan tugas	Amazon ECS memperkenalkan dukungan untuk flag Docker berikut dalam definisi tugas: • Mode IPC • Mode PID	16 Novbucket 2018
Dukungan rahasia Amazon ECS	Amazon ECS menambahkan dukungan untuk menggunakan AWS Systems Manager parameter Parameter Store untuk menyuntikkan data sensitif ke dalam wadah Anda. Untuk informasi selengkapnya, lihat Meneruskan data sensitif ke wadah Amazon ECS.	15 November 2018

Perubahan	Deskripsi	Tanggal
Penandaan sumber daya	Amazon ECS menambahkan dukungan untuk menambahkan tag metadata ke layanan, definisi tugas, tugas, kluster, dan instans penampung Anda. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon ECS.	15 November 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke wilayah AS Barat (California N.) dan Asia Pasifik (Seoul). Untuk informasi selengkapnya, lihat AWS Fargateuntuk Amazon ECS.	7 November 2018
Kuota layanan diperbarui	Kuota layanan berikut diperbarui: • Jumlah tugas yang menggunakan jenis peluncuran Fargate, per Wilayah, per akun dinaikkan dari ke20. 50 • Jumlah alamat IP publik untuk tugas yang menggunakan tipe peluncuran Fargate dinaikkan dari ke20. 50 Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECS.	31 Oktober 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke Wilayah Eropa (London). Untuk informasi selengkapnya, lihat AWS Fargateuntuk Amazon ECS.	26 Oktober 2018

Perubahan	Deskripsi	Tanggal
Amazon ECS yang Dioptimalkan Amazon Linux 2 AMI Dirilis	Amazon ECS menjual Linux AMIs yang dioptimalkan untuk layanan dalam dua varian. Versi terbaru dan yang direkomendasikan didasarkan pada x;. Amazon ECS juga vendor AMIs yang didasarkan pada, tetapi kami menyarankan Anda memigrasikan beban kerja Anda ke varian Amazon Linux 2, karena dukungan untuk Amazon Linux AMI akan berakhir selambat-lambatnya 30 Juni 2020. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	18 Oktober 2018
Titik Akhir Metadata Tugas Amazon ECS versi 3	Dimulai dengan versi 1.21.0 dari agen kontainer Amazon ECS, agen menyuntikkan variabel lingkungan yang dipanggil `ECS_CONTAINER_METADATA_URI` ke setiap kontainer dalam tugas. Saat Anda menanyakan titik akhir metadata tugas versi 3, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Pantau beban kerja menggunakan metadata Amazon ECS.	18 Oktober 2018
Penemuan layanan Amazon ECS Perluasan wilayah	Penemuan layanan Amazon ECS telah memperluas dukungan ke Kanada (Tengah), Amerika Selatan (São Paulo), Asia Pasifik (Seoul), Asia Pasifik (Mumbai), dan Wilayah Eropa (Paris). Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.	27 September 2018

Perubahan	Deskripsi	Tanggal
Menambahkan dukungan untuk flag Docker tambahan dalam ketentuan kontainer	Amazon ECS memperkenalkan dukungan untuk flag Docker berikut dalam definisi container: • Kontrol sistem • Interaktif • Terminal semu	17 September 2018
Dukungan otentikasi registri pribadi untuk Amazon ECS menggunakan tugas Fargate AWS	Amazon ECS memperkenalkan dukungan untuk tugas Fargate menggunakan otentikasi registri pribadi menggunakan. AWS Secrets Manager Fitur ini mengizinkan Anda untuk menyimpan kredensial Anda secara aman dan kemudian mereferensikan mereka dalam ketentuan kontainer Anda, yang mengizinkan tugas Anda untuk menggunakan citra privat. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.	10 September 2018
Penemuan layanan Amazon ECS Perluasan wilayah	Penemuan layanan Amazon ECS telah memperluas dukungan ke wilayah Asia Pasifik (Singapura), Asia Pasifik (Sydney), Asia Pasifik (Tokyo), UE (Frankfurt), dan Eropa (London). Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.	30 Agustus 2018
Tugas terjadwal dengan dukungan tugas Fargate	Amazon ECS memperkenalkan dukungan untuk tugas terjadwal untuk jenis peluncuran Fargate.	28 Agustus 2018

Perubahan	Deskripsi	Tanggal
Otentikasi registri pribadi menggunakan dukungan AWS Secrets Manager	Amazon ECS memperkenalkan dukungan untuk otentikasi registri pribadi menggunakan. AWS Secrets Manager Fitur ini mengizinkan Anda untuk menyimpan kredensial Anda secara aman dan kemudian mereferensikan mereka dalam ketentuan kontainer Anda, yang mengizinkan tugas Anda untuk menggunakan citra privat. Untuk informasi selengkapnya, lihat Menggunakan gambar AWS non-kontainer di Amazon ECS.	16 Agustus 2018
Dukungan volume docker ditambahkan	Amazon ECS memperkenalkan dukungan untuk volume Docker. Untuk informasi selengkapnya, lihat Opsi penyimpanan untuk tugas Amazon ECS.	9 Agustus 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah berkembang ke wilayah Eropa (Frankfurt), Asia Pasifik (Singapura), dan Asia Pasifik (Sydney). Untuk informasi selengkapnya, lihat AWS Fargateun tuk Amazon ECS.	19 Juli 2018

Perubahan	Deskripsi	Tanggal
Strategi penjadwal layanan Amazon ECS ditambahkan	Amazon ECS memperkenalkan konsep strategi penjadwal layanan. Ada dua strategi penjadwal layanan yang tersedia: • `REPLICA`Strategi penjadwalan replika menempatkan dan mempertahankan jumlah tugas yang diinginkan di seluruh klaster Anda. Secara default tugas tersebar di seluruh Availability Zone. Anda dapat menggunakan strategi penempatan tugas dan kendala untuk menyesuaikan keputusan penempatan tugas. Untuk informasi selengkapnya, lihat Strategi penjadwalan replika. • `DAEMON`—Strategi penjadwalan daemon menerapkan tepat satu tugas pada setiap instance kontainer aktif yang memenuhi semua batasan penempatan tugas yang Anda tentukan di cluster Anda. Saat menggunakan strategi ini, tidak perlu menentukan jumlah tugas yang diinginkan, strategi penempatan tugas, atau menggunakan kebijakan Auto Scaling Layanan. Untuk informasi selengkapnya, lihat Strategi penjadwalan daemon. **Note** Tugas Fargate tidak mendukung strategi `DAEMON` penjadwalan. .	12 Juni 2018

Perubahan	Deskripsi	Tanggal
Agen kontainer Amazon ECS v1.18.0	Versi baru dari agen penampung Amazon ECS dirilis, yang menambahkan fungsionalitas berikut: • Menambahkan dukungan untuk menyesuaikan perilaku penarikan citra agen kontainer menggunak an parameter `ECS_IMAGE_PULL_BEHAVIOR` . Untuk informasi selengkapnya, lihat Konfigurasi agen kontainer Amazon ECS. Untuk informasi lebih lanjut, lihat amazon-ecs-agent github.	24 Mei 2018
Menambahkan Dukungan untuk `bridge` dan Mode `host` Jaringan Saat Mengkonfigurasi Discovery Layanan	Menambahkan dukungan untuk mengonfigurasi penemuan layanan untuk layanan Amazon ECS menggunakan definisi tugas yang menentukan mode `bridge` atau `host` jaringan. Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.	22 Mei 2018
Menambahkan dukungan untuk parameter metadata AMI tambahan Amazon ECS yang dioptimalkan	Menambahkan subparameter yang memungkin kan Anda mengambil ID AMI yang dioptimalkan Amazon ECS, nama gambar, sistem operasi, versi agen kontainer, dan versi runtime yang dioptimalkan Amazon ECS. Kueri metadata menggunakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan.	9 Mei 2018
AWS Perluasan Wilayah Fargate	AWS Fargate dengan Amazon ECS telah diperluas ke Wilayah AS Timur (Ohio), AS Barat (Oregon), dan Uni Eropa Barat (Irlandia). Untuk informasi selengkapnya, lihat AWS Fargateun tuk Amazon ECS.	26 April 2018

Perubahan	Deskripsi	Tanggal
Pengambilan Metadata AMI yang dioptimalkan Amazon ECS	Menambahkan kemampuan untuk mengambil metadata AMI Amazon ECS yang dioptimalkan secara terprogram menggunakan Systems Manager Parameter Store API. Untuk informasi selengkapnya, lihat Mengambil metadata AMI Linux Amazon ECS yang dioptimalkan.	10 April 2018
AWS Versi platform Fargate	Versi platform AWS Fargate baru dirilis, yang berisi: • Menambahkan dukungan untuk Pantau beban kerja menggunakan metadata Amazon ECS. • Menambahkan dukungan untuk Pemeriksaan kondisi. • Menambahkan dukungan untuk Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS Untuk informasi selengkapnya, lihat Versi platform Fargate untuk Amazon ECS.	Maret 26, 2018
Penemuan Layanan Amazon ECS	Menambahkan integrasi dengan Route 53 untuk mendukung penemuan layanan Amazon ECS. Untuk informasi selengkapnya, lihat Gunakan penemuan layanan untuk menghubungkan layanan Amazon ECS dengan nama DNS.	22 Maret 2018
Dukungan shm-size dan tmpfs Docker	Menambahkan dukungan untuk parameter shm-size dan tmpfs Docker dalam definisi tugas Amazon ECS. Untuk informasi selengkapnya tentang sintaksis CLI ECS yang diperbarui, lihat Parameter Linux.	20 Maret 2018

Perubahan	Deskripsi	Tanggal
Pemeriksaan Kondisi Kontainer	Menambahkan dukungan untuk pemeriksaan kondisi Docker dalam ketentuan kontainer. Untuk informasi selengkapnya, lihat Pemeriksaan kondisi.	8 Maret 2018
AWS Fargate	Ditambahkan ikhtisar untuk Amazon ECS dengan AWS Fargate. Untuk informasi selengkapnya, lihat AWS Fargateuntuk Amazon ECS.	22 Februari 2018
Titik Akhir Metadata Tugas Amazon ECS	Dimulai dengan versi 1.17.0 dari agen penampung Amazon ECS, berbagai metadata tugas dan statistik Docker tersedia untuk tugas yang menggunakan mode `awsvpc` jaringan pada titik akhir HTTP yang disediakan oleh agen penampung Amazon ECS. Untuk informasi selengkapnya, lihat Pantau beban kerja menggunakan metadata Amazon ECS.	8 Februari 2018
Auto Scaling Amazon ECS Service menggunakan kebijakan pelacakan target	Menambahkan dukungan untuk Auto Scaling Layanan ECS menggunakan kebijakan pelacakan target di konsol Amazon ECS. Untuk informasi selengkapnya, lihat Gunakan metrik target untuk menskalakan layanan Amazon ECS. Menghapus tutorial sebelumnya untuk penskalaan langkah di dalam ECS yang pertama kali menjalankan wizard. Tutorial ini diganti dengan tutorial baru untuk pelacakan target.	8 Februari 2018
Dukungan Docker versi 17.09	Menambahkan dukungan untuk Docker 17.09. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	18 Januari 2018
Perilaku penjadwal layanan baru	Informasi yang diperbarui tentang perilaku untuk tugas layanan yang gagal diluncurkan. Mendokumentasikan pesan acara layanan baru yang dipicu ketika tugas layanan mengalami kegagalan berturut-turut.	11 Januari 2018

Perubahan	Deskripsi	Tanggal
Periode tunggu inisialisasi pemeriksaan kesehatan Elastic Load Balancing	Menambahkan kemampuan untuk menentukan masa tunggu pemeriksaan kondisi.	Desember 27, 2017
CPU dan memori tingkat tugas	Menambahkan dukungan untuk menentukan CPU dan memori pada tingkat tugas dalam ketentuan tugas. Untuk informasi selengkapnya, lihat TaskDefinition.	12 Desember 2017
Peran pelaksanaan tugas	Agen penampung Amazon ECS melakukan panggilan ke tindakan Amazon ECS API atas nama Anda, sehingga memerlukan kebijakan dan peran IAM agar layanan mengetahui bahwa agen tersebut milik Anda. Tindakan berikut termasuk peran eksekusi tugas: • Panggilan ke Amazon ECR untuk menarik gambar kontainer • Panggilan CloudWatch untuk menyimpan log aplikasi kontainer Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas Amazon ECS.	7 Desember 2017
Kontainer Windows mendukung umumnya tersedia (GA)	Menambahkan dukungan untuk wadah Windows Server 2016. Untuk informasi selengkapnya, lihat Varian AMI yang dioptimalkan Amazon ECS.	5 Desember 2017
AWS Fargate GA	Menambahkan dukungan untuk meluncurkan layanan Amazon ECS menggunakan jenis peluncuran Fargate. Untuk informasi selengkapnya, lihat Jenis peluncuran Amazon ECS.	29 November 2017
Perubahan nama Amazon ECS	Amazon Elastic Container Service diganti namanya (sebelumnya Amazon EC2 Container Service).	21 November 2017

Perubahan	Deskripsi	Tanggal
Jaringan tugas	Fitur jaringan tugas yang disediakan oleh mode jaringan awsvpc memberikan tugas Amazon ECS properti jaringan yang sama dengan instans Amazon. EC2 Saat Anda menggunakan mode jaringan awsvpc dalam ketentuan tugas, setiap tugas yang diluncurkan dari ketentuan tugas tersebut mendapatkan antarmuka jaringan elastis, alamat IP privat primer, dan nama host DNS internalnya sendiri. Fitur jaringan tugas menyederhanakan jaringan kontainer dan memberi Anda lebih banyak kontrol atas bagaimana aplikasi kontainer berkomunikasi satu sama lain dan layanan lain di dalam Anda. VPCs Untuk informasi selengkap nya, lihat Opsi jaringan tugas Amazon ECS untuk jenis EC2 peluncuran.	14 November 2017
Metadata wadah Amazon ECS	Container Amazon ECS sekarang dapat mengakses metadata seperti wadah Docker atau ID gambar, konfigurasi jaringan, atau Amazon. ARNs Untuk informasi selengkapnya, lihat File metadata wadah Amazon ECS.	2 November 2017
Dukungan Docker versi 17.06	Menambahkan dukungan untuk Docker versi 17.06. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	2 November 2017
Dukungan untuk flag Docker: perangkat dan init	Menambahkan dukungan untuk perangkat Docker dan fitur init dalam ketentuan tugas menggunak an parameter `LinuxParameters` (`devices` dan `initProcessEnabled`). Untuk informasi selengkapnya, lihat LinuxParameters.	2 November 2017
Dukungan untuk flag Docker: cap-add dan cap-drop	Menambahkan dukungan untuk fitur cap-add dan cap-drop Docker dalam ketentuan tugas menggunakan parameter`LinuxParameters` (`capabilities`). Untuk informasi selengkapnya, lihat LinuxParameters.	22 September 2017

Perubahan	Deskripsi	Tanggal
Dukungan Penyeimbang Beban Jaringan	Amazon ECS menambahkan dukungan untuk Network Load Balancers di konsol Amazon ECS.	7 September 2017
RunTask mengesamp ingkan	Menambahkan dukungan untuk penggantian ketentuan tugas saat menjalankan tugas. Hal ini mengizinkan Anda untuk menjalankan suatu tugas bersamaan dengan mengubah ketentuan tugas tanpa perlu membuat revisi ketentuan tugas yang baru. Untuk informasi selengkapnya, lihat Menjalankan aplikasi sebagai tugas Amazon ECS.	27 Juni 2017
Tugas terjadwal Amazon ECS	Menambahkan dukungan untuk tugas penjadwalan menggunakan cron.	7 Juni 2017
Spot Instans di konsol Amazon ECS	Menambahkan dukungan untuk membuat instans kontainer Spot Fleet dalam konsol Amazon ECS. Untuk informasi selengkapnya, lihat Meluncurkan instans penampung Amazon ECS Linux.	6 Juni 2017
Pemberitahuan Amazon SNS untuk rilis AMI baru yang dioptimalkan Amazon ECS	Menambahkan kemampuan untuk berlangganan pemberitahuan SNS tentang rilis AMI baru yang dioptimalkan Amazon ECS.	Kamis, 23 Maret 2017
Tugas layanan mikro dan batch	Menambahkan dokumentasi untuk dua kasus penggunaan umum untuk Amazon ECS: layanan mikro dan pekerjaan batch.	Februari 2017
Pengosongan instans kontainer	Menambahkan dukungan untuk pengosongan instans kontainer, yang menyediakan metode untuk menghapus instans kontainer dari klaster. Untuk informasi selengkapnya, lihat Menguras instans kontainer Amazon ECS.	24 Januari 2017

Perubahan	Deskripsi	Tanggal
Dukungan Docker 1.12	Menambahkan dukungan untuk Docker 1.12. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	24 Januari 2017
Strategi penempatan tugas baru	Menambahkan dukungan untuk strategi penempata n tugas: penempatan berbasis atribut, paket bin, penyebaran Availability Zone, dan satu per host. Untuk informasi selengkapnya, lihat Gunakan strategi untuk menentukan penempatan tugas Amazon ECS.	Desember 29, 2016
Dukungan kontainer Windows dalam versi beta	Menambahkan dukungan untuk wadah Windows Server 2016 (beta). Untuk informasi selengkapnya, lihat Varian AMI yang dioptimalkan Amazon ECS.	20 Desember 2016
Dukungan Blox OSS	Menambahkan dukungan untuk Blox OSS, yang mengizinkan penjadwal tugas kustom. Untuk informasi selengkapnya, lihat Jadwalkan wadah Anda di Amazon ECS.	1 Desember 2016
Aliran acara Amazon ECS untuk CloudWatch Acara	Amazon ECS sekarang mengirimkan instance kontainer dan perubahan status tugas ke CloudWatc h Peristiwa. Untuk informasi selengkapnya, lihat Otomatiskan tanggapan terhadap kesalahan Amazon ECS menggunakan EventBridge.	21 November 2016
Pencatatan kontainer Amazon ECS ke CloudWatch Log	Menambahkan dukungan untuk driver awslogs untuk mengirim aliran log kontainer ke Log. CloudWatch Untuk informasi selengkapnya, lihat Kirim log Amazon ECS ke CloudWatch .	September 12, 2016

Perubahan	Deskripsi	Tanggal
Layanan Amazon ECS dengan dukungan Elastic Load Balancing untuk port dinamis	Menambahkan dukungan untuk penyeimbang beban agar dapat mendukung beberapa instans, yaitu: kombinasi port per listener, yang dapat meningkat kan fleksibilitas untuk kontainer. Kini Anda dapat secara dinamis membiarkan Docker menentukan port host kontainer dan penjadwal ECS yang mendaftar kan instans:port dengan penyeimbang beban. Untuk informasi selengkapnya, lihat Gunakan load balancing untuk mendistribusikan lalu lintas layanan Amazon ECS.	11 Agustus, 2016
Peran IAM untuk tugas Amazon ECS	Menambahkan dukungan untuk mengaitkan peran IAM dengan tugas. Ini memberikan izin yang detail ke kontainer kebalikan dari peran tunggal untuk seluruh instans kontainer. Untuk informasi selengkapnya, lihat Peran IAM tugas Amazon ECS.	13 Juli 2016
Dukungan Docker 1.11	Menambahkan dukungan untuk Docker 1.11. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	31 Mei 2016
Penskalaan otomatis tugas	Amazon ECS menambahkan dukungan untuk secara otomatis menskalakan tugas Anda yang dijalanka n oleh layanan. Untuk informasi selengkapnya, lihat Secara otomatis menskalakan layanan Amazon ECS Anda.	18 Mei 2016
Pemfilteran ketentuan tugas pada famili tugas	Menambahkan dukungan untuk memfilter daftar ketentuan tugas berdasarkan famili ketentuan tugas. Untuk informasi selengkapnya, lihat ListTaskDefinition s.	17 Mei 2016
Kontainer Docker dan pencatatan agen Amazon ECS	Amazon ECS menambahkan kemampuan untuk mengirim agen ECS dan log kontainer Docker dari instance kontainer ke CloudWatch Log untuk menyederhanakan masalah pemecahan masalah.	5 Mei 2016

Perubahan	Deskripsi	Tanggal
AMI yang dioptimal kan ECS sekarang mendukung Amazon Linux 2016.03.	AMI yang dioptimalkan ECS menambahkan dukungan untuk Amazon Linux 2016.03. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimal kan AMIs.	5 April 2016
Dukungan Docker 1.9	Menambahkan dukungan untuk Docker versi 1.9. Untuk informasi selengkapnya, lihat Linux Amazon ECS yang dioptimalkan AMIs.	Desember 22, 2015
CloudWatch metrik untuk CPU cluster dan reservasi memori	Amazon ECS menambahkan CloudWatch metrik khusus untuk reservasi CPU dan memori.	Desember 22, 2015
Pengalaman Amazon ECS baru yang dijalankan pertama	Pengalaman pertama kali dijalankan konsol Amazon ECS menambahkan pembuatan peran nol-klik.	November 23, 2015
Penempatan tugas di seluruh Availability Zone	Penjadwal layanan Amazon ECS menambahkan dukungan untuk penempatan tugas di seluruh Availabil ity Zone.	8 Oktober 2015
CloudWatch metrik untuk kluster dan layanan Amazon ECS	Amazon ECS menambahkan CloudWatch metrik khusus untuk pemanfaatan CPU dan memori untuk setiap instance kontainer, layanan, dan keluarga definisi tugas dalam sebuah cluster. Metrik baru ini dapat digunakan untuk menskalakan instance kontainer di klaster menggunakan grup Auto Scaling atau untuk membuat alarm khusus. CloudWatch	17 Agustus 2015
Dukungan port UDP	Menambahkan dukungan untuk port UDP dalam ketentuan tugas.	Juli 7, 2015
Penggantian variabel lingkungan	Menambahkan dukungan untuk deregisterTaskDefi nition dan penggantian variabel lingkungan untuk RunTask.	18 Juni 2015

Perubahan	Deskripsi	Tanggal
Pembaruan agen Amazon ECS otomatis	Menambahkan kemampuan untuk melihat versi agen ECS yang berjalan pada instans kontainer. Juga dapat memperbarui agen ECS dari AWS Management Console, AWS CLI, dan SDK.	11 Juni 2015
Penjadwal layanan Amazon ECS dan integrasi Elastic Load Balancing	Menambahkan kemampuan untuk menentukan layanan dan mengaitkan layanan tersebut dengan penyeimbang beban Elastic Load Balancing.	9 April 2015
Amazon ECS GA	Ketersediaan umum Amazon ECS di Wilayah AS Timur (Virginia N.), AS Barat (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia).	9 April 2015

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.