



Panduan Developer

Batas Waktu Cloud



Batas Waktu Cloud: 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 Deadline Cloud?	1
Open Job Description	2
Konsep dan terminologi	2
Apa itu beban kerja Deadline Cloud	6
Bagaimana beban kerja muncul dari produksi	6
Bahan-bahan dari beban kerja	7
Portabilitas beban kerja	8
Memulai	11
Buat peternakan	11
Langkah selanjutnya	15
Jalankan agen pekerja	16
Langkah selanjutnya	18
Kirim lowongan kerja	18
Kirim simple_job sampel	19
Kirim dengan parameter	22
Buat pekerjaan simple_file_job	23
Langkah selanjutnya	26
Kirim pekerjaan dengan lampiran	26
Konfigurasi antrian untuk lampiran pekerjaan	27
Kirim dengan lampiran pekerjaan	30
Bagaimana lampiran pekerjaan disimpan	32
Langkah selanjutnya	36
Menambahkan armada yang dikelola layanan	36
Langkah selanjutnya	39
Bersihkan sumber daya pertanian	39
Konfigurasi pekerjaan menggunakan lingkungan antrian	43
Kontrol lingkungan kerja	44
Tetapkan variabel lingkungan	45
Atur jalurnya	49
Jalankan proses daemon latar belakang	53
Menyediakan aplikasi untuk pekerjaan Anda	59
Mendapatkan aplikasi dari saluran conda	60
Gunakan manajer paket yang berbeda	62
Buat saluran conda menggunakan S3	63

Buat antrian pembuatan paket	64
Konfigurasi izin antrian pembuatan paket	64
Konfigurasi izin antrian produksi untuk paket conda kustom	65
Menambahkan saluran conda ke lingkungan antrian	66
Buat paket conda untuk aplikasi	67
Buat resep conda build untuk Blender	68
Kirim Blender 4.2 paket pekerjaan	70
Uji paket Anda dengan Blender 4.2 pekerjaan render	72
Buat resep conda untuk Maya	72
Buat resep conda untuk MtoA plugin	75
Uji paket Anda dengan Maya pekerjaan render	77
Membangun pekerjaan	78
Job bundel	79
Elemen template Job	82
Nilai parameter elemen	85
Elemen referensi aset	87
Menggunakan file dalam pekerjaan Anda	90
Contoh infrastruktur proyek	91
Profil penyimpanan dan pemetaan jalur	93
Lampiran Job	101
Mengirimkan file dengan pekerjaan	102
Mendapatkan file output dari pekerjaan	113
Menggunakan file dalam langkah dependen	117
Buat batas sumber daya untuk pekerjaan	119
Menghentikan dan menghapus batas	121
Buat batas	122
Kaitkan batas dan antrian	122
Kirim pekerjaan yang membutuhkan batasan	123
Mengirim tugas	124
Dari terminal	125
Dari naskah	126
Dari dalam aplikasi	127
Jadwalkan pekerjaan	129
Tentukan kompatibilitas armada	129
Penskalaan armada	131
Sesi	131

Ketergantungan langkah	134
Ubah pekerjaan	135
Armada yang dikelola pelanggan	141
Buat CMF	141
Pengaturan host pekerja	147
Konfigurasikan lingkungan Python	148
Instal agen pekerja	148
Konfigurasikan agen pekerja	150
Buat pengguna dan grup pekerjaan	151
Kelola akses	154
Berikan akses	155
Mencabut akses	155
Instal perangkat lunak untuk pekerjaan	156
Pasang adaptor DCC	157
Konfigurasikan kredensial	157
Konfigurasikan jaringan	160
Uji host pekerja Anda	161
Buat sebuah AMI	164
Siapkan instance	164
Membangun AMI	166
Buat infrastruktur armada	167
Skalakan armada Anda secara otomatis	172
Pemeriksaan kesehatan armada	177
Armada yang dikelola layanan	178
Jalankan skrip admin	178
Pemecahan Masalah	181
Hubungkan sumber daya VPC ke SMF Anda	184
Cara kerja titik akhir sumber daya VPC	185
Prasyarat	185
Siapkan titik akhir sumber daya VPC	186
Mengakses sumber daya VPC Anda	186
Otentikasi dan keamanan	187
Pertimbangan teknis	187
Pemecahan Masalah	187
Menggunakan lisensi perangkat lunak	189
Connect armada SMF ke server lisensi	189

Langkah 1: Konfigurasi lingkungan antrian	190
Langkah 2: (Opsional) Pengaturan instance proxy lisensi	196
Langkah 3: pengaturan AWS CloudFormation template	198
Connect armada CMF ke titik akhir lisensi	206
Langkah 1: Buat grup keamanan	206
Langkah 2: Siapkan titik akhir lisensi	207
Langkah 3: Hubungkan aplikasi rendering ke titik akhir	208
Langkah 4: Hapus titik akhir lisensi	211
Pemantauan	212
CloudTrail log	213
Deadline Cloud peristiwa data di CloudTrail	215
Deadline Cloud acara manajemen di CloudTrail	217
Deadline Cloud contoh acara	220
Pemantauan CloudWatch dengan	221
CloudWatch metrik	222
Mengelola acara menggunakan EventBridge	224
Acara Batas Waktu Cloud	225
Mengirim acara Deadline Cloud	226
Referensi detail acara	227
Keamanan	242
Perlindungan data	243
Enkripsi diam	244
Enkripsi bergerak	244
Manajemen kunci	245
Privasi lalu lintas antar jaringan	255
Menyisih	255
Identity and Access Management	256
Audiens	257
Mengautentikasi dengan identitas	258
Mengelola akses menggunakan kebijakan	261
Bagaimana Deadline Cloud bekerja dengan IAM	264
Contoh kebijakan berbasis identitas	271
AWS kebijakan terkelola	275
Pemecahan Masalah	280
Validasi kepatuhan	282
Ketahanan	283

Keamanan infrastruktur	283
Konfigurasi dan analisis kerentanan	284
Pencegahan "confused deputy" lintas layanan	285
AWS PrivateLink	286
Pertimbangan	287
Deadline Cloud titik akhir	287
Buat titik akhir	288
Praktik terbaik keamanan	289
Perlindungan data	289
Izin IAM	290
Jalankan pekerjaan sebagai pengguna dan grup	290
Jaringan	291
Data Job	291
Struktur pertanian	291
Antrian lampiran pekerjaan	292
Bucket perangkat lunak khusus	295
Tuan rumah pekerja	295
Skrip konfigurasi host	297
Workstation	297
Verifikasi perangkat lunak yang diunduh	298
Riwayat dokumen	304
.....	cccvii

Apa itu AWS Deadline Cloud?

AWS Deadline Cloud adalah AWS layanan yang dikelola sepenuhnya yang memungkinkan Anda memiliki pertanian pemrosesan yang dapat diskalakan dan berjalan dalam hitungan menit. Ini menyediakan konsol administrasi untuk mengelola pengguna, peternakan, antrian untuk pekerjaan penjadwalan, dan armada pekerja yang melakukan pemrosesan.

Panduan pengembang ini untuk pengembang pipeline, alat, dan aplikasi dalam berbagai kasus penggunaan, termasuk yang berikut ini:

- Pengembang pipa dan direktur teknis dapat mengintegrasikan Deadline Cloud APIs dan fitur ke dalam pipeline produksi khusus mereka.
- Vendor perangkat lunak independen dapat mengintegrasikan Deadline Cloud ke dalam aplikasi mereka yang memungkinkan seniman dan pengguna pembuatan konten digital untuk mengirimkan pekerjaan render Deadline Cloud dengan mulus dari workstation mereka.
- Pengembang layanan berbasis web dan cloud dapat mengintegrasikan rendering Deadline Cloud ke dalam platform mereka, memungkinkan pelanggan menyediakan aset untuk melihat produk secara virtual.

Kami menyediakan alat yang memungkinkan Anda untuk bekerja secara langsung dengan setiap langkah pipa Anda:

- Antarmuka baris perintah yang dapat Anda gunakan secara langsung atau dari skrip.
- AWS SDK untuk 11 bahasa pemrograman populer.
- Antarmuka web berbasis REST yang dapat Anda panggil dari aplikasi Anda.

Anda juga dapat menggunakan yang lain Layanan AWS di aplikasi khusus Anda. Misalnya, Anda dapat menggunakan:

- AWS CloudFormation untuk mengotomatiskan pembuatan dan penghapusan peternakan, antrian, dan armada.
- Amazon CloudWatch mengumpulkan metrik untuk pekerjaan.
- Amazon Simple Storage Service untuk menyimpan dan mengelola aset digital dan output pekerjaan.
- AWS IAM Identity Center untuk mengelola pengguna dan grup untuk peternakan Anda.

Open Job Description

Deadline Cloud menggunakan [spesifikasi Open Job Description \(OpenJD\)](#) untuk menentukan detail pekerjaan. OpenJD dikembangkan untuk mendefinisikan pekerjaan yang portabel antara solusi. Anda menggunakannya untuk menentukan pekerjaan yang merupakan sekumpulan perintah yang berjalan pada host pekerja.

Anda dapat membuat template pekerjaan OpenJD menggunakan pengirim yang disediakan Deadline Cloud, atau Anda dapat menggunakan alat apa pun yang ingin Anda buat templat. Setelah membuat template, Anda mengirimkannya ke Deadline Cloud. Jika Anda menggunakan submitter, itu akan mengurus pengiriman template. Jika Anda membuat template dengan cara lain, Anda memanggil tindakan baris perintah Deadline Cloud, atau Anda dapat menggunakan salah satunya AWS SDKs untuk mengirim pekerjaan. Either way, Deadline Cloud menambahkan pekerjaan ke antrian yang ditentukan dan menjadwalkan pekerjaan.

Konsep dan terminologi untuk Deadline Cloud

Untuk membantu Anda memulai dengan AWS Deadline Cloud, topik ini menjelaskan beberapa konsep dan terminologi utamanya.

Manajer anggaran

Manajer anggaran adalah bagian dari monitor Deadline Cloud. Gunakan manajer anggaran untuk membuat dan mengelola anggaran. Anda juga dapat menggunakannya untuk membatasi aktivitas agar tetap sesuai anggaran.

Pustaka Klien Cloud Batas Waktu

Pustaka Klien menyertakan antarmuka baris perintah dan pustaka untuk mengelola Deadline Cloud. Fungsionalitas termasuk mengirimkan bundel pekerjaan berdasarkan spesifikasi Open Job Description ke Deadline Cloud, mengunduh output lampiran pekerjaan, dan memantau pertanian Anda menggunakan antarmuka baris perintah.

Aplikasi pembuatan konten digital (DCC)

Aplikasi pembuatan konten digital (DCCs) adalah produk pihak ketiga tempat Anda membuat konten digital. Contohnya DCCs adalah Maya, Nuke, dan Houdini. Deadline Cloud menyediakan plugin terintegrasi pengirim pekerjaan untuk spesifik. DCCs

Peternakan

Peternakan adalah tempat sumber daya proyek Anda berada. Ini terdiri dari antrian dan armada.

Armada

Armada adalah sekelompok node pekerja yang melakukan rendering. Node pekerja memproses pekerjaan. Armada dapat dikaitkan dengan beberapa antrian, dan antrian dapat dikaitkan dengan beberapa armada.

Instans

Armada menggunakan instance untuk sumber daya CPU. Instance adalah instance EC2 kinerja Amazon. Deadline Cloud menggunakan instans On-Demand dan Spot.

Contoh Sesuai Permintaan

Instans On-Demand dihargai oleh yang kedua, tidak memiliki komitmen jangka panjang, dan tidak akan terganggu.

Contoh spot

Instans spot adalah kapasitas tanpa syarat yang dapat Anda gunakan dengan harga diskon, tetapi dapat terganggu oleh permintaan Sesuai Permintaan.

Tunggu dan Simpan

Fitur Tunggu dan Simpan menyediakan penjadwalan pekerjaan tertunda untuk biaya lebih rendah dan dapat terganggu oleh permintaan On-Demand dan Spot. Tunggu dan Simpan hanya tersedia dalam armada yang dikelola layanan Deadline Cloud.

Tunggu dan Simpan adalah untuk mengelola eksekusi beban kerja komputasi visual di AWS Deadline Cloud. Lihat [ketentuan AWS layanan](#) untuk detailnya.

Pekerjaan

Pekerjaan adalah permintaan rendering. Pengguna mengirimkan pekerjaan. Pekerjaan berisi properti pekerjaan tertentu yang diuraikan sebagai langkah dan tugas.

Lampiran Job

Lampiran pekerjaan adalah fitur Deadline Cloud yang dapat Anda gunakan untuk mengelola input dan output untuk pekerjaan. File Job diunggah sebagai lampiran pekerjaan selama proses rendering. File-file ini dapat berupa tekstur, model 3D, rig pencahayaan, dan item serupa lainnya.

Prioritas Job

Prioritas Job adalah perkiraan urutan Deadline Cloud memproses pekerjaan dalam antrian. Anda dapat menetapkan prioritas pekerjaan antara 1 dan 100, pekerjaan dengan prioritas angka yang

lebih tinggi umumnya diproses terlebih dahulu. Pekerjaan dengan prioritas yang sama diproses dalam urutan yang diterima.

Properti Job

Properti Job adalah pengaturan yang Anda tentukan saat mengirimkan pekerjaan render. Beberapa contoh termasuk rentang bingkai, jalur keluaran, lampiran pekerjaan, kamera yang dapat dirender, dan banyak lagi. Properti bervariasi berdasarkan DCC tempat render dikirimkan.

Templat Job

Template pekerjaan mendefinisikan lingkungan runtime dan semua proses yang berjalan sebagai bagian dari pekerjaan Deadline Cloud.

Antrian

Antrian adalah tempat pekerjaan yang diajukan berada dan dijadwalkan akan diberikan. Antrian harus dikaitkan dengan armada untuk membuat render yang berhasil. Antrian dapat dikaitkan dengan beberapa armada.

Asosiasi antrian armada

Ketika antrian dikaitkan dengan armada, ada asosiasi antrian-armada. Gunakan asosiasi untuk menjadwalkan pekerja dari armada ke pekerjaan dalam antrian itu. Anda dapat memulai dan menghentikan asosiasi untuk mengontrol penjadwalan kerja.

Sesi

Sesi adalah lingkungan runtime sementara pada host pekerja yang dibuat untuk menjalankan serangkaian tugas dari pekerjaan yang sama. Sesi berakhir ketika host pekerja selesai menjalankan tugas untuk pekerjaan itu.

Sesi ini menyediakan cara untuk mengonfigurasi lingkungan dengan sumber daya yang dibagikan di beberapa tugas yang dijalankan, seperti mendefinisikan variabel lingkungan atau memulai proses latar belakang atau wadah.

Tindakan sesi

Tindakan sesi adalah unit kerja diskrit yang dijalankan oleh pekerja dalam suatu sesi. Ini dapat mencakup operasi tugas inti dari suatu tugas, atau mungkin termasuk langkah-langkah persiapan seperti pengaturan lingkungan dan proses pasca-eksekusi seperti pembongkaran dan pembersihan.

Langkah

Langkah adalah salah satu proses khusus untuk dijalankan dalam pekerjaan.

Batas waktu pengirim Cloud

Submitter Deadline Cloud adalah plugin pembuatan konten digital (DCC). Artis menggunakannya untuk mengirimkan pekerjaan dari antarmuka DCC pihak ketiga yang mereka kenal.

Tanda

Tag adalah label yang dapat Anda tetapkan ke AWS sumber daya. Setiap tag terdiri dari kunci dan nilai opsional yang Anda tentukan.

Dengan tag, Anda dapat mengkategorikan AWS sumber daya Anda dengan berbagai cara. Misalnya, Anda dapat menentukan satu set tag untuk EC2 instans Amazon akun Anda yang membantu Anda melacak setiap pemilik instans dan tingkat tumpukan.

Anda juga dapat mengkategorikan AWS sumber daya Anda berdasarkan tujuan, pemilik, atau lingkungan. Pendekatan ini berguna ketika Anda memiliki banyak sumber daya dari jenis yang sama. Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda tetapkan padanya.

Tugas

Tugas adalah komponen tunggal dari langkah render.

Lisensi berbasis penggunaan (UBL)

Usage-based licensing (UBL) adalah model lisensi on-demand yang tersedia untuk produk pihak ketiga tertentu. Model ini dibayar sesuai keinginan Anda, dan Anda dikenakan biaya untuk jumlah jam dan menit yang Anda gunakan.

Penjelajah penggunaan

Penjelajah penggunaan adalah fitur monitor Deadline Cloud. Ini memberikan perkiraan perkiraan biaya dan penggunaan Anda.

Pekerja

Pekerja termasuk dalam armada dan menjalankan tugas yang diberikan Deadline Cloud untuk menyelesaikan langkah dan pekerjaan. Pekerja menyimpan log dari operasi tugas di Amazon CloudWatch Logs. Pekerja juga dapat menggunakan fitur lampiran pekerjaan untuk menyinkronkan input dan output ke bucket Amazon Simple Storage Service (Amazon S3).

Apa itu beban kerja Deadline Cloud

Dengan AWS Deadline Cloud, Anda dapat mengirimkan pekerjaan untuk menjalankan aplikasi Anda di cloud dan memproses data untuk produksi konten atau wawasan yang penting bagi bisnis Anda. Deadline Cloud menggunakan [Open Job Description](#) (OpenJD) sebagai sintaks untuk template pekerjaan, spesifikasi yang dirancang untuk kebutuhan pipeline komputasi visual tetapi berlaku untuk banyak kasus penggunaan lainnya. Beberapa contoh beban kerja termasuk rendering grafis komputer, simulasi fisika, dan fotogrametri.

Skala beban kerja dari bundel pekerjaan sederhana yang dikirimkan pengguna ke antrian dengan CLI atau GUI yang dibuat secara otomatis, ke plugin pengirim terintegrasi yang secara dinamis menghasilkan bundel pekerjaan untuk beban kerja yang ditentukan aplikasi.

Bagaimana beban kerja muncul dari produksi

Untuk memahami beban kerja dalam konteks produksi dan cara mendukungnya dengan Deadline Cloud, pertimbangkan bagaimana hasilnya. Produksi mungkin melibatkan pembuatan efek visual, animasi, game, citra katalog produk, rekonstruksi 3D untuk pemodelan informasi bangunan (BIM), dan banyak lagi. Konten ini biasanya dibuat oleh tim spesialis artistik atau teknis yang menjalankan berbagai aplikasi perangkat lunak dan skrip khusus. Anggota tim meneruskan data antara satu sama lain menggunakan pipa produksi. Banyak tugas yang dilakukan oleh pipeline melibatkan perhitungan intensif yang akan memakan waktu sehari-hari jika dijalankan di workstation pengguna.

Beberapa contoh tugas dalam jaringan pipa produksi ini meliputi:

- Menggunakan aplikasi fotogrametri untuk memproses foto yang diambil dari set film untuk merekonstruksi mesh digital bertekstur.
- Menjalankan simulasi partikel dalam adegan 3D untuk menambahkan lapisan detail ke efek visual ledakan untuk acara televisi.
- Memasak data untuk level game ke dalam formulir yang diperlukan untuk rilis eksternal dan menerapkan pengaturan optimasi dan kompresi.
- Merender satu set gambar untuk katalog produk termasuk variasi warna, latar belakang, dan pencahayaan.
- Menjalankan skrip yang dikembangkan khusus pada model 3D untuk menerapkan tampilan yang dibuat khusus dan disetujui oleh sutradara film.

Tugas-tugas ini melibatkan banyak parameter untuk disesuaikan untuk mendapatkan hasil artistik atau untuk menyempurnakan kualitas output. Seringkali ada GUI untuk memilih nilai-nilai parameter dengan tombol atau menu untuk menjalankan proses secara lokal dalam aplikasi. Ketika pengguna menjalankan proses, aplikasi dan mungkin komputer host itu sendiri tidak dapat digunakan untuk melakukan operasi lain karena menggunakan status aplikasi dalam memori dan dapat mengkonsumsi semua sumber daya CPU dan memori komputer host.

Dalam banyak kasus prosesnya cepat. Selama produksi, kecepatan proses melambat ketika persyaratan untuk kualitas dan kompleksitas naik. Tes karakter yang memakan waktu 30 detik selama pengembangan dapat dengan mudah berubah menjadi 3 jam ketika diterapkan pada karakter produksi akhir. Melalui perkembangan ini, beban kerja yang memulai kehidupan di dalam GUI dapat tumbuh terlalu besar untuk muat. Mementingnya ke Deadline Cloud dapat meningkatkan produktivitas pengguna yang menjalankan proses ini karena mereka mendapatkan kembali kendali penuh atas workstation mereka dan dapat melacak lebih banyak iterasi dari monitor Deadline Cloud.

Ada dua tingkat dukungan yang harus dituju ketika mengembangkan dukungan untuk beban kerja di Deadline Cloud:

- Bongkar beban kerja dari workstation pengguna ke farm Deadline Cloud tanpa paralelisme atau percepatan. Ini mungkin kurang memanfaatkan sumber daya komputasi yang tersedia di pertanian, tetapi kemampuan untuk mengalihkan operasi panjang ke sistem pemrosesan batch memungkinkan pengguna untuk menyelesaikan lebih banyak pekerjaan dengan workstation mereka sendiri.
- Mengoptimalkan paralelisme beban kerja sehingga memanfaatkan skala horizontal Deadline Cloud farm untuk menyelesaikan dengan cepat.

Ada kalanya jelas bagaimana membuat beban kerja berjalan secara paralel. Misalnya, setiap frame render grafis komputer dapat dilakukan secara independen. Namun, penting untuk tidak terjebak pada paralelisme ini. Alih-alih, pahami bahwa menurunkan beban kerja yang berjalan lama ke Deadline Cloud memberikan manfaat yang signifikan, bahkan ketika tidak ada cara yang jelas untuk membagi beban kerja.

Bahan-bahan dari beban kerja

Untuk menentukan beban kerja Deadline Cloud, terapkan paket pekerjaan yang dikirimkan pengguna ke antrian dengan [Deadline](#) Cloud CLI. Sebagian besar pekerjaan dalam membuat bundel pekerjaan adalah menulis template pekerjaan, tetapi ada lebih banyak faktor seperti bagaimana menyediakan

aplikasi yang dibutuhkan beban kerja. Berikut adalah hal-hal penting yang perlu dipertimbangkan saat menentukan beban kerja untuk Deadline Cloud:

- Aplikasi untuk dijalankan. Pekerja harus dapat meluncurkan proses aplikasi, dan oleh karena itu memerlukan instalasi aplikasi yang tersedia serta lisensi apa pun yang digunakan aplikasi, seperti akses ke server lisensi mengambang. Ini biasanya bagian dari konfigurasi pertanian, dan tidak tertanam dalam bundel pekerjaan itu sendiri.
 - [Konfigurasi pekerjaan menggunakan lingkungan antrian](#)
 - [Connect armada yang dikelola pelanggan ke titik akhir lisensi](#)
- Definisi parameter Job. Pengalaman pengguna dalam mengirimkan pekerjaan sangat dipengaruhi oleh parameter yang disediakan. Contoh parameter termasuk file data, direktori, dan konfigurasi aplikasi.
 - [Nilai parameter elemen untuk bundel pekerjaan](#)
- Aliran data file. Ketika pekerjaan berjalan, ia membaca input dari file yang disediakan oleh pengguna, kemudian menulis outputnya sebagai file baru. Untuk bekerja dengan lampiran pekerjaan dan fitur pemetaan jalur, pekerjaan harus menentukan jalur direktori atau file tertentu untuk input dan output ini.
 - [Menggunakan file dalam pekerjaan Anda](#)
- Skrip langkah. Skrip langkah menjalankan biner aplikasi dengan opsi baris perintah yang tepat untuk menerapkan parameter pekerjaan yang disediakan. Ini juga menangani detail seperti pemetaan jalur jika file data beban kerja menyertakan referensi jalur absolut, bukan referensi jalur relatif.
 - [Elemen template Job untuk bundel pekerjaan](#)

Portabilitas beban kerja

Beban kerja portabel ketika dapat berjalan di beberapa sistem yang berbeda tanpa mengubahnya setiap kali Anda mengirimkan pekerjaan. Misalnya, ini mungkin berjalan di farm render berbeda yang memiliki sistem file bersama yang berbeda yang dipasang, atau pada sistem operasi yang berbeda seperti Linux atau Windows. Saat Anda menerapkan bundel pekerjaan portabel, lebih mudah bagi pengguna untuk menjalankan pekerjaan di pertanian spesifik mereka, atau menyesuaikannya untuk kasus penggunaan lainnya.

Berikut adalah beberapa cara Anda dapat membuat bundel pekerjaan Anda portabel.

- Tentukan sepenuhnya file data input yang dibutuhkan oleh beban kerja, menggunakan parameter PATH pekerjaan dan referensi aset dalam bundel pekerjaan. Ini membuat pekerjaan portabel ke

peternakan berdasarkan sistem file bersama dan ke peternakan yang membuat salinan data input, seperti fitur lampiran pekerjaan Deadline Cloud.

- Buat referensi jalur file untuk file input pekerjaan yang dapat dipindahkan dan dapat digunakan pada sistem operasi yang berbeda. Misalnya ketika pengguna mengirimkan pekerjaan dari Windows workstation untuk dijalankan pada Linux armada.
- Gunakan referensi jalur file relatif, jadi jika direktori yang berisi mereka dipindahkan ke lokasi yang berbeda, referensi masih diselesaikan. Beberapa aplikasi, seperti [Blender](#), mendukung pilihan antara jalur relatif dan absolut.
- Jika Anda tidak dapat menggunakan jalur relatif, dukung [metadata pemetaan jalur OpenJD dan terjemahkan jalur absolut sesuai dengan cara Deadline](#) Cloud menyediakan file ke pekerjaan.
- Menerapkan perintah dalam pekerjaan menggunakan skrip portabel. Python dan bash adalah dua contoh bahasa scripting yang dapat digunakan dengan cara ini. Anda harus mempertimbangkan untuk menyediakan keduanya di semua host pekerja armada Anda.
- Gunakan biner interpreter skrip, seperti python atau bash, dengan nama file skrip sebagai argumen. Ini bekerja pada semua sistem operasi termasuk Windows, dibandingkan dengan menggunakan file skrip dengan bit eksekusi disetel Linux.
- Tulis skrip bash portabel dengan menerapkan praktik ini:
 - Perluas parameter jalur template dalam tanda kutip tunggal untuk menangani jalur dengan spasi dan Windows pemisah jalur.
 - Saat berjalan Windows, perhatikan masalah yang terkait dengan terjemahan jalur otomatis MinGW. Misalnya, ia mengubah AWS CLI perintah seperti `aws logs tail /aws/deadline/...` menjadi perintah yang mirip dengan `aws logs tail "C:/Program Files/Git/aws/deadline/..."` dan tidak akan mengekor log dengan benar. Tetapkan variabel `MSYS_NO_PATHCONV=1` untuk menonaktifkan perilaku ini.
 - Dalam kebanyakan kasus, kode yang sama berfungsi pada semua sistem operasi. Ketika kode harus berbeda, gunakan `if/else` konstruksi untuk menangani kasus.

```
if [[ "$(uname)" == MINGW* || "$(uname -s)" == MSYS_NT* ]]; then
    # Code for Windows
elif [[ "$(uname)" == Darwin ]]; then
    # Code for MacOS
else
    # Code for Linux and other operating systems
fi
```

- Anda dapat menulis skrip Python portabel menggunakan `pathlib` untuk menangani perbedaan jalur sistem file dan menghindari fitur khusus pengoperasian. Dokumentasi Python menyertakan

anotasi untuk ini, misalnya dalam dokumentasi pustaka [sinyal](#). Linux-Dukungan fitur khusus ditandai sebagai “Ketersediaan: Linux.”

- Gunakan parameter pekerjaan untuk menentukan persyaratan aplikasi. Gunakan konvensi konsisten yang dapat diterapkan oleh administrator pertanian di lingkungan [antrian](#).
- Misalnya, Anda dapat menggunakan CondaPackages dan/atau RezPackages parameter dalam pekerjaan Anda, dengan nilai parameter default yang mencantumkan nama paket aplikasi dan versi yang dibutuhkan pekerjaan. Kemudian, Anda dapat menggunakan salah satu [contoh lingkungan antrian Conda atau Rez untuk menyediakan lingkungan](#) virtual untuk pekerjaan itu.

Memulai dengan sumber daya Deadline Cloud

Untuk mulai membuat solusi khusus untuk AWS Deadline Cloud, Anda harus menyiapkan sumber daya Anda. Ini termasuk pertanian, setidaknya satu antrian untuk pertanian, dan setidaknya satu armada pekerja untuk melayani antrian. Anda dapat membuat sumber daya menggunakan konsol Deadline Cloud, atau Anda dapat menggunakan AWS Command Line Interface

Dalam tutorial ini, Anda akan menggunakan AWS CloudShell untuk membuat peternakan pengembang sederhana dan menjalankan agen pekerja. Anda kemudian dapat mengirimkan dan menjalankan pekerjaan sederhana dengan parameter dan lampiran, menambahkan armada yang dikelola layanan, dan membersihkan sumber daya pertanian Anda setelah selesai.

Bagian berikut memperkenalkan Anda ke berbagai fitur Deadline Cloud, dan bagaimana mereka berfungsi dan bekerja sama. Mengikuti langkah-langkah ini berguna untuk mengembangkan dan menguji beban kerja dan penyesuaian baru.

Untuk petunjuk cara menyiapkan farm menggunakan konsol, lihat [Memulai](#) di Panduan Pengguna Cloud Tenggat Waktu.

Topik

- [Buat pertanian Cloud Deadline](#)
- [Jalankan agen pekerja Deadline Cloud](#)
- [Kirim dengan Deadline Cloud](#)
- [Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud](#)
- [Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud](#)
- [Bersihkan sumber daya pertanian Anda di Deadline Cloud](#)

Buat pertanian Cloud Deadline

Untuk membuat resource farm dan antrian developer di AWS Deadline Cloud, gunakan AWS Command Line Interface (AWS CLI), seperti yang ditunjukkan pada prosedur berikut. Anda juga akan membuat peran AWS Identity and Access Management (IAM) dan armada yang dikelola pelanggan (CMF) dan mengaitkan armada dengan antrian Anda. Kemudian Anda dapat mengonfigurasi AWS CLI dan mengonfirmasi bahwa peternakan Anda sudah diatur dan berfungsi seperti yang ditentukan.

Anda dapat menggunakan peternakan ini untuk menjelajahi fitur Deadline Cloud, kemudian mengembangkan dan menguji beban kerja, penyesuaian, dan integrasi pipeline baru.

Untuk membuat peternakan

1. [Buka AWS CloudShell sesi](#). Anda akan menggunakan CloudShell jendela untuk memasukkan AWS Command Line Interface (AWS CLI) perintah untuk menjalankan contoh dalam tutorial ini. Biarkan CloudShell jendela tetap terbuka saat Anda melanjutkan.
2. Buat nama untuk peternakan Anda, dan tambahkan nama pertanian itu ke `~/.bashrc`. Ini akan membuatnya tersedia untuk sesi terminal lainnya.

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. Buat sumber daya pertanian, dan tambahkan ID pertaniannya ke `~/.bashrc`.

```
aws deadline create-farm \
  --display-name "$DEV_FARM_NAME"

echo "DEV_FARM_ID=$(aws deadline list-farms \
  --query \"farms[?displayName=='$DEV_FARM_NAME'].farmId \
  | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. Buat sumber daya antrian, dan tambahkan ID antreannya ke `~/.bashrc`.

```
aws deadline create-queue \
  --farm-id $DEV_FARM_ID \
  --display-name "$DEV_FARM_NAME Queue" \
  --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
  "runAs": "QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=$(aws deadline list-queues \
  --farm-id $DEV_FARM_ID \
  --query \"queues[?displayName=='$DEV_FARM_NAME Queue'].queueId \
  | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. Buat peran IAM untuk armada. Peran ini memberi host pekerja di armada Anda kredensi keamanan yang diperlukan untuk menjalankan pekerjaan dari antrian Anda.

```

aws iam create-role \
  --role-name "${DEV_FARM_NAME}FleetRole" \
  --assume-role-policy-document \
    '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.deadline.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
aws iam put-role-policy \
  --role-name "${DEV_FARM_NAME}FleetRole" \
  --policy-name WorkerPermissions \
  --policy-document \
    '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "deadline:AssumeFleetRoleForWorker",
            "deadline:UpdateWorker",
            "deadline>DeleteWorker",
            "deadline:UpdateWorkerSchedule",
            "deadline:BatchGetJobEntity",
            "deadline:AssumeQueueRoleForWorker"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "aws:PrincipalAccount": "${aws:ResourceAccount}"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "logs>CreateLogStream"

```

```

    ],
    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
    "Condition": {
        "StringEquals": {
            "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
    "Condition": {
        "StringEquals": {
            "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
    }
}
]
}'

```

6. Buat armada yang dikelola pelanggan (CMF), dan tambahkan ID armadanya ke. ~/.bashrc

```

FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
    --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME CMF" \
    --role-arn $FLEET_ROLE_ARN \
    --max-worker-count 5 \
    --configuration \
    '{
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {"min": 1},
                "memoryMiB": {"min": 512},
                "osFamily": "linux",
                "cpuArchitectureType": "x86_64"
            }
        }
    }'

```

```

    }'

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
    --farm-id \$DEV_FARM_ID \
    --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
    | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc

```

7. Kaitkan CMF dengan antrian Anda.

```

aws deadline create-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_CMF_ID

```

8. Instal antarmuka baris perintah Deadline Cloud.

```

pip install deadline

```

9. Untuk menyetel farm default ke ID farm dan antrian ke ID antrian yang Anda buat sebelumnya, gunakan perintah berikut.

```

deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID

```

10. (Opsional) Untuk mengonfirmasi bahwa peternakan Anda diatur sesuai dengan spesifikasi Anda, gunakan perintah berikut:

- Daftar semua peternakan — **deadline farm list**
- Buat daftar semua antrian di pertanian default — **deadline queue list**
- Daftar semua armada di peternakan default — **deadline fleet list**
- Dapatkan peternakan default — **deadline farm get**
- Dapatkan antrian default — **deadline queue get**
- Dapatkan semua armada yang terkait dengan antrian default — **deadline fleet get**

Langkah selanjutnya

Setelah membuat peternakan, Anda dapat menjalankan agen pekerja Deadline Cloud di host di armada Anda untuk memproses pekerjaan. Lihat [Jalankan agen pekerja Deadline Cloud](#).

Jalankan agen pekerja Deadline Cloud

Sebelum Anda dapat menjalankan pekerjaan yang Anda kirimkan ke antrian di peternakan pengembang Anda, Anda harus menjalankan agen pekerja AWS Deadline Cloud dalam mode pengembang pada host pekerja.

Sepanjang sisa tutorial ini, Anda akan melakukan AWS CLI operasi di peternakan pengembang Anda menggunakan dua AWS CloudShell tab. Di tab pertama, Anda dapat mengirimkan pekerjaan. Di tab kedua, Anda dapat menjalankan agen pekerja.

Note

Jika Anda membiarkan CloudShell sesi Anda menganggur selama lebih dari 20 menit, itu akan batas waktu dan menghentikan agen pekerja. Untuk memulai kembali agen pekerja, ikuti instruksi dalam prosedur berikut.

Sebelum Anda dapat memulai agen pekerja, Anda harus menyiapkan pertanian Deadline Cloud, antrian, dan armada. Lihat [Buat pertanian Cloud Deadline](#).

Untuk menjalankan agen pekerja dalam mode pengembang

1. Dengan pertanian Anda masih terbuka di CloudShell tab pertama, buka CloudShell tab kedua, lalu buat `demoenv-persist` direktori `demoenv-logs` dan.

```
mkdir ~/demoenv-logs
mkdir ~/demoenv-persist
```

2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

Note

Pada Windows, diperlukan bahwa file agen diinstal ke direktori paket situs global Python. Lingkungan virtual Python saat ini tidak didukung.

```
python -m pip install deadline-cloud-worker-agent
```

- Untuk memungkinkan agen pekerja membuat direktori sementara untuk menjalankan pekerjaan, buat direktori:

```
sudo mkdir /sessions
sudo chmod 750 /sessions
sudo chown cloudshell-user /sessions
```

- Jalankan agen pekerja Deadline Cloud dalam mode pengembang dengan variabel DEV_FARM_ID dan DEV_CMF_ID yang Anda tambahkan ke ~/.bashrc.

```
deadline-worker-agent \
  --farm-id $DEV_FARM_ID \
  --fleet-id $DEV_CMF_ID \
  --run-jobs-as-agent-user \
  --logs-dir ~/demoenv-logs \
  --persistence-dir ~/demoenv-persist
```

Saat agen pekerja menginisialisasi dan kemudian melakukan polling pada operasi UpdateWorkerSchedule API, output berikut ditampilkan:

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
Python Version: 3.9.16 (main, Sep  8 2023, 00:00:00) - [GCC 11.4.1 20230605 (Red
Hat 11.4.1-2)]
Platform: linux
...
[2024-03-27 15:51:02,528][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

- Pilih CloudShell tab pertama Anda, lalu daftarkan pekerja di armada.

```
deadline worker list --fleet-id $DEV_CMF_ID
```

Output seperti berikut ini ditampilkan:

```
Displaying 1 of 1 workers starting at 0  
  
- workerId: worker-8c9af877c8734e89914047111f  
  status: STARTED  
  createdAt: 2023-12-13 20:43:06+00:00
```

Dalam konfigurasi produksi, agen pekerja Deadline Cloud memerlukan pengaturan beberapa pengguna dan direktori konfigurasi sebagai pengguna administratif di mesin host. Anda dapat mengganti pengaturan ini karena Anda menjalankan pekerjaan di peternakan pengembangan Anda sendiri, yang hanya dapat Anda akses.

Langkah selanjutnya

Sekarang agen pekerja berjalan di host pekerja Anda, Anda dapat mengirim pekerjaan ke pekerja Anda. Anda dapat:

- [Kirim dengan Deadline Cloud](#) menggunakan bundel pekerjaan OpenJD sederhana.
- [Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud](#) yang berbagi file antar workstation menggunakan sistem operasi yang berbeda.

Kirim dengan Deadline Cloud

Untuk menjalankan pekerjaan Deadline Cloud di host pekerja Anda, Anda membuat dan menggunakan paket pekerjaan Open Job Description (OpenJD) untuk mengonfigurasi pekerjaan. Bundel mengkonfigurasi pekerjaan, misalnya dengan menentukan file input untuk pekerjaan dan di mana untuk menulis output pekerjaan. Topik ini mencakup contoh cara Anda dapat mengonfigurasi bundel pekerjaan.

Sebelum Anda dapat mengikuti prosedur di bagian ini, Anda harus menyelesaikan yang berikut:

- [Buat pertanian Cloud Deadline](#)
- [Jalankan agen pekerja Deadline Cloud](#)

Untuk menggunakan AWS Deadline Cloud untuk menjalankan pekerjaan, gunakan prosedur berikut. Gunakan AWS CloudShell tab pertama untuk mengirimkan pekerjaan ke peternakan pengembang Anda. Gunakan CloudShell tab kedua untuk melihat output agen pekerja.

Topik

- [Kirim simple_job sampel](#)
- [Kirim simple_job dengan parameter](#)
- [Buat bundel pekerjaan simple_file_job dengan file I/O](#)
- [Langkah selanjutnya](#)

Kirim simple_job sampel

Setelah Anda membuat peternakan dan menjalankan agen pekerja, Anda dapat mengirimkan simple_job contoh ke Deadline Cloud.

Untuk mengirimkan simple_job contoh ke Deadline Cloud

1. Pilih CloudShell tab pertama Anda.
2. Unduh sampel dari GitHub.

```
cd ~  
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. Arahkan ke direktori sampel bundel pekerjaan.

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. Kirim simple_job sampel.

```
deadline bundle submit simple_job
```

5. Pilih CloudShell tab kedua Anda untuk melihat output logging tentang panggilanBatchGetJobEntities, mendapatkan sesi, dan menjalankan tindakan sesi.

```
...  
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting  
# [session-053d77cef82648fe2] Starting new Session.  
[queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
```

```
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
{'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'}},
'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
'*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
[session-053d77cef82648fea9c69827182] Appended new SessionActions.
(ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
[queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
[session-053d77cef82648fea9c69827182] Running as the Worker Agent's
user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
[session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
[session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
[queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
[session-053d77cef82648fea9c69827182] Logs streamed to: local
file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
[queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

Note

Hanya output logging dari agen pekerja yang ditampilkan. Ada log terpisah untuk sesi yang menjalankan pekerjaan.

6. Pilih tab pertama Anda, lalu periksa file log yang ditulis agen pekerja.

- a. Arahkan ke direktori log agen pekerja dan lihat isinya.

```
cd ~/demoenv-logs  
ls
```

- b. Cetak file log pertama yang dibuat oleh agen pekerja.

```
cat worker-agent-bootstrap.log
```

File ini berisi output agen pekerja tentang bagaimana itu disebut Deadline Cloud API untuk membuat sumber daya pekerja di armada Anda, dan kemudian mengambil peran armada.

- c. Cetak output file log saat agen pekerja bergabung dengan armada.

```
cat worker-agent.log
```

Log ini berisi output tentang semua tindakan yang diambil agen pekerja, tetapi tidak berisi output tentang antrian tempat ia menjalankan pekerjaan, kecuali untuk sumber daya tersebut IDs .

- d. Cetak file log untuk setiap sesi dalam direktori yang diberi nama sama dengan id sumber daya antrian.

```
cat $DEV_QUEUE_ID/session-*.log
```

Jika pekerjaan berhasil, output file log akan mirip dengan yang berikut:

```
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

```
2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.  
2024-03-27 16:00:22,404 INFO  
2024-03-27 16:00:22,405 INFO =====  
2024-03-27 16:00:22,405 INFO ----- Running Task  
2024-03-27 16:00:22,405 INFO =====  
2024-03-27 16:00:22,406 INFO -----  
2024-03-27 16:00:22,406 INFO Phase: Setup  
2024-03-27 16:00:22,406 INFO -----  
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
```

```

2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_files_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_files_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO -----
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO -----
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO =====
2024-03-27 16:00:22,572 INFO ----- Session Cleanup
2024-03-27 16:00:22,572 INFO =====
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a

```

7. Cetak informasi tentang pekerjaan itu.

```
deadline job get
```

Saat Anda mengirimkan pekerjaan, sistem menyimpannya sebagai default sehingga Anda tidak perlu memasukkan ID pekerjaan.

Kirim simple_job dengan parameter

Anda dapat mengirimkan pekerjaan dengan parameter. Dalam prosedur berikut, Anda mengedit simple_job template untuk menyertakan pesan kustom, kirimkan simple_job, lalu cetak file log sesi untuk melihat pesan.

Untuk mengirimkan simple_job sampel dengan parameter

1. Pilih CloudShell tab pertama Anda, lalu arahkan ke direktori sampel bundel pekerjaan.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Cetak isi simple_job templat.

```
cat simple_job/template.yaml
```

parameterDefinitionsBagian dengan Message parameter akan terlihat seperti berikut:

```
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
```

3. Kirim simple_job sampel dengan nilai parameter, lalu tunggu pekerjaan selesai berjalan.

```
deadline bundle submit simple_job \
  -p "Message=Greetings from the developer getting started guide."
```

4. Untuk melihat pesan kustom, lihat file log sesi terbaru.

```
cd ~/demoenv-logs
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

Buat bundel pekerjaan simple_file_job dengan file I/O

Pekerjaan render perlu membaca definisi adegan, merender gambar darinya, dan kemudian menyimpan gambar itu ke file output. Anda dapat mensimulasikan tindakan ini dengan membuat pekerjaan menghitung hash input alih-alih merender gambar.

Untuk membuat bundel pekerjaan simple_file_job dengan file I/O

1. Pilih CloudShell tab pertama Anda, lalu arahkan ke direktori sampel bundel pekerjaan.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Buat salinan simple_job dengan nama barusimple_file_job.

```
cp -r simple_job simple_file_job
```

3. Edit template pekerjaan sebagai berikut:

 Note

Kami menyarankan Anda menggunakan nano untuk langkah-langkah ini. Jika Anda lebih suka menggunakan Vim, Anda harus mengatur mode tempel menggunakan `:set paste`.

- a. Buka template di editor teks.

```
nano simple_file_job/template.yaml
```

- b. Tambahkan yang berikut `initType`, `objectType`, dan `dataFlowparameterDefinitions`.

```
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
```

- c. Tambahkan perintah bash script berikut ke akhir file yang membaca dari file input dan menulis ke file output.

```
# hash the input file, and write that to the output
sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

Yang diperbarui `template.yaml` harus sama persis dengan yang berikut:

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
```

```

- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
      - name: Run
        type: TEXT
        runnable: true
        data: |
          #!/usr/bin/env bash
          echo "{{Param.Message}}"

          # hash the input file, and write that to the output
          sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"

```

Note

Jika Anda ingin menyesuaikan spasi di `template.yaml`, pastikan Anda menggunakan spasi alih-alih lekukan.

- d. Simpan file, dan keluar dari editor teks.
4. Berikan nilai parameter untuk file input dan output untuk mengirimkan `simple_file_job`.

```

deadline bundle submit simple_file_job \
  -p "InFile=simple_job/template.yaml" \
  -p "OutFile=hash.txt"

```

5. Cetak informasi tentang pekerjaan itu.

```
deadline job get
```

- Anda akan melihat output seperti berikut:

```

parameters:
  Message:

```

```
string: Welcome to AWS Deadline Cloud!  
InFile:  
  path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/  
template.yaml  
OutFile:  
  path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- Meskipun Anda hanya menyediakan jalur relatif, parameter memiliki jalur lengkap yang disetel. AWS CLI Menggabungkan direktori kerja saat ini ke jalur apa pun yang disediakan sebagai parameter saat jalur memiliki tipePATH.
- Agen pekerja yang berjalan di jendela terminal lain mengambil dan menjalankan pekerjaan. Tindakan ini membuat hash.txt file, yang dapat Anda lihat dengan perintah berikut.

```
cat hash.txt
```

Perintah ini akan mencetak output yang mirip dengan berikut ini.

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/  
cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
```

Langkah selanjutnya

Setelah mempelajari cara mengirimkan pekerjaan sederhana menggunakan Deadline Cloud CLI, Anda dapat menjelajahi:

- [Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud](#) untuk mempelajari cara menjalankan pekerjaan pada host yang menjalankan sistem operasi yang berbeda.
- [Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud](#) untuk menjalankan pekerjaan Anda di host yang dikelola oleh Deadline Cloud.
- [Bersihkan sumber daya pertanian Anda di Deadline Cloud](#) untuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Kirim pekerjaan dengan lampiran pekerjaan di Deadline Cloud

Banyak peternakan menggunakan sistem file bersama untuk berbagi file antara host yang mengirimkan pekerjaan dan yang menjalankan pekerjaan. Misalnya, pada `simple_file_job`

contoh sebelumnya, sistem file lokal dibagi antara jendela AWS CloudShell terminal, yang berjalan di tab satu tempat Anda mengirimkan pekerjaan, dan tab dua tempat Anda menjalankan agen pekerja.

Sistem file bersama menguntungkan ketika workstation submitter dan host pekerja berada di jaringan area lokal yang sama. Jika Anda menyimpan data Anda di lokasi dekat workstation yang mengaksesnya, maka menggunakan farm berbasis cloud berarti Anda harus membagikan sistem file Anda melalui VPN latensi tinggi atau menyinkronkan sistem file Anda di cloud. Tak satu pun dari opsi ini mudah diatur atau dioperasikan.

AWS Deadline Cloud menawarkan solusi sederhana dengan lampiran pekerjaan, yang mirip dengan lampiran email. Dengan lampiran pekerjaan, Anda melampirkan data ke pekerjaan Anda. Kemudian, Deadline Cloud menangani detail transfer dan penyimpanan data pekerjaan Anda di bucket Amazon Simple Storage Service (Amazon S3).

Alur kerja pembuatan konten sering berulang, artinya pengguna mengirimkan pekerjaan dengan subset kecil file yang dimodifikasi. Karena bucket Amazon S3 menyimpan lampiran pekerjaan dalam penyimpanan yang dapat dialamatkan konten, nama setiap objek didasarkan pada hash data objek dan konten pohon direktori disimpan dalam format file manifes yang dilampirkan ke pekerjaan.

Sebelum Anda dapat mengikuti prosedur di bagian ini, Anda harus menyelesaikan yang berikut:

- [Buat pertanian Cloud Deadline](#)
- [Jalankan agen pekerja Deadline Cloud](#)

Untuk menjalankan pekerjaan dengan lampiran pekerjaan, selesaikan langkah-langkah berikut.

Topik

- [Tambahkan konfigurasi lampiran pekerjaan ke antrian Anda](#)
- [Kirim simple_file_job dengan lampiran pekerjaan](#)
- [Memahami bagaimana lampiran pekerjaan disimpan di Amazon S3](#)
- [Langkah selanjutnya](#)

Tambahkan konfigurasi lampiran pekerjaan ke antrian Anda

Untuk mengaktifkan lampiran pekerjaan dalam antrian Anda, tambahkan konfigurasi lampiran pekerjaan ke sumber daya antrian di akun Anda.

Untuk menambahkan konfigurasi lampiran pekerjaan ke antrian Anda

1. Pilih CloudShell tab pertama Anda, lalu masukkan salah satu perintah berikut untuk menggunakan bucket Amazon S3 untuk lampiran pekerjaan.
 - Jika Anda tidak memiliki bucket Amazon S3 pribadi yang sudah ada, Anda dapat membuat dan menggunakan bucket S3 baru.

```
DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
  | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
else LOCATION_CONSTRAINT="--create-bucket-configuration \
  LocationConstraint=${AWS_REGION}"
fi
aws s3api create-bucket \
  $LOCATION_CONSTRAINT \
  --acl private \
  --bucket ${DEV_FARM_BUCKET}
```

- Jika Anda sudah memiliki bucket Amazon S3 pribadi, Anda dapat menggunakannya *MY_BUCKET_NAME* dengan menggantinya dengan nama bucket Anda.

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

2. Setelah membuat atau memilih bucket Amazon S3, tambahkan nama bucket agar bucket tersedia ~/ .bashrc untuk sesi terminal lainnya.

```
echo "DEV_FARM_BUCKET=$DEV_FARM_BUCKET" >> ~/.bashrc
source ~/.bashrc
```

3. Buat peran AWS Identity and Access Management (IAM) untuk antrian.

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
  --assume-role-policy-document \
    '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.deadline.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
```

```

        }
    ]
}'
aws iam put-role-policy \
  --role-name "${DEV_FARM_NAME}QueueRole" \
  --policy-name S3BucketsAccess \
  --policy-document \
    '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": [
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*",
            "s3:DeleteObject*",
            "s3:PutObject",
            "s3:PutObjectLegalHold",
            "s3:PutObjectRetention",
            "s3:PutObjectTagging",
            "s3:PutObjectVersionTagging",
            "s3:Abort*"
          ],
          "Resource": [
            "arn:aws:s3:::'$DEV_FARM_BUCKET'",
            "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
          ],
          "Effect": "Allow"
        }
      ]
    }'
```

4. Perbarui antrian Anda untuk menyertakan pengaturan lampiran pekerjaan dan peran IAM.

```

QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
  --query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --role-arn $QUEUE_ROLE_ARN \
  --job-attachment-settings \
    '{
      "s3BucketName": "'$DEV_FARM_BUCKET'",
      "rootPrefix": "JobAttachments"
    }'
```

```
}'
```

5. Konfirmasikan bahwa Anda memperbarui antrian Anda.

```
deadline queue get
```

Output seperti berikut ini ditampilkan:

```
...
jobAttachmentSettings:
  s3BucketName: DEV_FARM_BUCKET
  rootPrefix: JobAttachments
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole
...
```

Kirim simple_file_job dengan lampiran pekerjaan

Saat Anda menggunakan lampiran pekerjaan, paket pekerjaan harus memberi Deadline Cloud informasi yang cukup untuk menentukan aliran data pekerjaan, seperti menggunakan parameter. PATH Dalam kasus simple_file_job, Anda mengedit template.yaml file untuk memberi tahu Deadline Cloud bahwa aliran data ada di file input dan file output.

Setelah menambahkan konfigurasi lampiran pekerjaan ke antrian, Anda dapat mengirimkan sampel simple_file_job dengan lampiran pekerjaan. Setelah Anda melakukan ini, Anda dapat melihat logging dan output pekerjaan untuk mengonfirmasi bahwa simple_file_job dengan lampiran pekerjaan berfungsi.

Untuk mengirimkan bundel pekerjaan simple_file_job dengan lampiran pekerjaan

1. Pilih CloudShell tab pertama Anda, lalu buka JobBundle-Samples direktori.

2.

```
cd ~/deadline-cloud-samples/job_bundles/
```

3. Kirim simple_file_job ke antrian. Saat diminta untuk mengonfirmasi unggahan, masukkan y.

```
deadline bundle submit simple_file_job \
  -p InFile=simple_job/template.yaml \
  -p OutFile=hash-jobattachments.txt
```

4. Untuk melihat output log sesi transfer data lampiran pekerjaan, jalankan perintah berikut.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --query "sessions[0].sessionId" \
  --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```

5. Buat daftar tindakan sesi yang dijalankan dalam sesi.

```
aws deadline list-session-actions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --session-id $SESSION_ID
```

Output seperti berikut ini ditampilkan:

```
{
  "sessionactions": [
    {
      "sessionActionId": "sessionaction-123-0",
      "status": "SUCCEEDED",
      "startedAt": "<timestamp>",
      "endedAt": "<timestamp>",
      "progressPercent": 100.0,
      "definition": {
        "syncInputJobAttachments": {}
      }
    },
    {
      "sessionActionId": "sessionaction-123-1",
      "status": "SUCCEEDED",
      "startedAt": "<timestamp>",
      "endedAt": "<timestamp>",
      "progressPercent": 100.0,
      "definition": {
        "taskRun": {
          "taskId": "task-abc-0",
          "stepId": "step-def"
        }
      }
    }
  ]
}
```

```
}  
  }  
] }  
}
```

Tindakan sesi pertama mengunduh lampiran pekerjaan input, sedangkan tindakan kedua menjalankan tugas seperti pada langkah sebelumnya dan kemudian mengunggah lampiran pekerjaan keluaran.

6. Daftar direktori output.

```
ls *.txt
```

Output seperti `hash.txt` ada di direktori, tetapi `hash-jobattachments.txt` tidak ada karena file output dari pekerjaan belum diunduh.

7. Unduh output dari pekerjaan terbaru.

```
deadline job download-output
```

8. Lihat output dari file yang diunduh.

```
cat hash-jobattachments.txt
```

Output seperti berikut ini ditampilkan:

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /tmp/openjd/  
session-123/assetroot-abc/simple_job/template.yaml
```

Memahami bagaimana lampiran pekerjaan disimpan di Amazon S3

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengunggah atau mengunduh data untuk lampiran pekerjaan, yang disimpan di bucket Amazon S3. Memahami bagaimana Deadline Cloud menyimpan lampiran pekerjaan di Amazon S3 akan membantu saat Anda mengembangkan beban kerja dan integrasi pipeline.

Untuk memeriksa bagaimana lampiran pekerjaan Deadline Cloud disimpan di Amazon S3

1. Pilih CloudShell tab pertama Anda, lalu buka direktori sampel bundel pekerjaan.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Periksa properti pekerjaan.

```
deadline job get
```

Output seperti berikut ini ditampilkan:

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
template.yaml
  OutFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
jobattachments.txt
attachments:
  manifests:
    - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
      rootPathFormat: posix
      outputRelativeDirectories:
        - .
      inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
      inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
      fileSystem: COPIED
```

Bidang lampiran berisi daftar struktur manifes yang menjelaskan jalur data input dan output yang digunakan pekerjaan saat dijalankan. Lihatlah `rootPath` untuk melihat jalur direktori lokal pada mesin yang mengirimkan pekerjaan. Untuk melihat akhiran objek Amazon S3 yang berisi file manifes, tinjau. `inputManifestFile` File manifes berisi metadata untuk snapshot pohon direktori dari data input pekerjaan.

3. Cetak objek manifes Amazon S3 dengan cantik untuk melihat struktur direktori input untuk pekerjaan tersebut.

```
MANIFEST_SUFFIX=$(aws deadline get-job \
  --farm-id $DEV_FARM_ID \
```

```
--queue-id $DEV_QUEUE_ID \
--job-id $JOB_ID \
--query "attachments.manifests[0].inputManifestPath" \
--output text)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .
```

Output seperti berikut ini ditampilkan:

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "2ec297b04c59c4741ed97ac8fb83080c",
      "mtime": 1698186190000000,
      "path": "simple_job/template.yaml",
      "size": 445
    }
  ],
  "totalSize": 445
}
```

4. Buat awalan Amazon S3 yang menyimpan manifes untuk lampiran pekerjaan keluaran dan daftarkan objek di bawahnya.

```
SESSION_ACTION=$(aws deadline list-session-actions \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --job-id $JOB_ID \
  --session-id $SESSION_ID \
  --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX
```

Lampiran pekerjaan keluaran tidak direferensikan secara langsung dari sumber daya pekerjaan tetapi ditempatkan di bucket Amazon S3 berdasarkan sumber daya pertanian. IDs

5. Dapatkan kunci objek manifes terbaru untuk id tindakan sesi tertentu, lalu cetak objek manifes dengan cantik.

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionId)
MANIFEST_KEY=$(aws s3api list-objects-v2 \
  --bucket $DEV_FARM_BUCKET \
  --prefix $TASK_OUTPUT_PREFIX \
  --query "Contents[*].Key" --output text \
  | grep $SESSION_ACTION_ID \
  | sort | tail -1)
MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
echo $MANIFEST_OBJECT | jq .
```

Anda akan melihat properti file `hash-jobattachments.txt` dalam output seperti berikut ini:

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
      "mtime": 1698785252554950,
      "path": "hash-jobattachments.txt",
      "size": 182
    }
  ],
  "totalSize": 182
}
```

Pekerjaan Anda hanya akan memiliki satu objek manifes per tugas yang dijalankan, tetapi secara umum dimungkinkan untuk memiliki lebih banyak objek per tugas yang dijalankan.

6. Lihat output penyimpanan Amazon S3 yang dapat dialamatkan konten di bawah awalan. Data

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

Output seperti berikut ini ditampilkan:

```
eea2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /tmp/openjd/  
session-123/assetroot-abc/simple_job/template.yaml
```

Langkah selanjutnya

Setelah mempelajari cara mengirimkan pekerjaan dengan lampiran menggunakan Deadline Cloud CLI, Anda dapat menjelajahi:

- [Kirim dengan Deadline Cloud](#) untuk mempelajari cara menjalankan pekerjaan menggunakan bundel OpenJD di host pekerja Anda.
- [Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud](#) untuk menjalankan pekerjaan Anda di host yang dikelola oleh Deadline Cloud.
- [Bersihkan sumber daya pertanian Anda di Deadline Cloud](#) untuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Menambahkan armada yang dikelola layanan ke farm pengembang Anda di Deadline Cloud

AWS CloudShell tidak menyediakan kapasitas komputasi yang cukup untuk menguji beban kerja yang lebih besar. Ini juga tidak dikonfigurasi untuk bekerja dengan pekerjaan yang mendistribusikan tugas di beberapa host pekerja.

Alih-alih menggunakan CloudShell, Anda dapat menambahkan armada terkelola layanan Auto Scaling (SMF) ke peternakan pengembang Anda. SMF menyediakan kapasitas komputasi yang cukup untuk beban kerja yang lebih besar dan dapat menangani pekerjaan yang perlu mendistribusikan tugas pekerjaan di beberapa host pekerja.

Sebelum menambahkan SMF, Anda harus menyiapkan pertanian, antrian, dan armada Deadline Cloud. Lihat [Buat pertanian Cloud Deadline](#).

Untuk menambahkan armada yang dikelola layanan ke peternakan pengembang Anda

1. Pilih AWS CloudShell tab pertama Anda, lalu buat armada yang dikelola layanan dan tambahkan ID armadanya. `.bashrc` Tindakan ini membuatnya tersedia untuk sesi terminal lainnya.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
```

```

--query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
  --farm-id $DEV_FARM_ID \
  --display-name "$DEV_FARM_NAME SMF" \
  --role-arn $FLEET_ROLE_ARN \
  --max-worker-count 5 \
  --configuration \
    '{
      "serviceManagedEc2": {
        "instanceCapabilities": {
          "vCpuCount": {
            "min": 2,
            "max": 4
          },
          "memoryMiB": {
            "min": 512
          },
          "osFamily": "linux",
          "cpuArchitectureType": "x86_64"
        },
        "instanceMarketOptions": {
          "type": "spot"
        }
      }
    }'

echo "DEV_SMF_ID=$(aws deadline list-fleets \
  --farm-id $DEV_FARM_ID \
  --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
  | [0]" --output text)" >> ~/.bashrc
source ~/.bashrc

```

2. Kaitkan SMF dengan antrian Anda.

```

aws deadline create-queue-fleet-association \
  --farm-id $DEV_FARM_ID \
  --queue-id $DEV_QUEUE_ID \
  --fleet-id $DEV_SMF_ID

```

3. Kirim `simple_file_job` ke antrian. Saat diminta untuk mengonfirmasi unggahan, masukkan `y`.

```

deadline bundle submit simple_file_job \
  -p InFile=simple_job/template.yaml \

```

```
-p OutFile=hash-jobattachments.txt
```

4. Konfirmasikan SMF berfungsi dengan benar.

```
deadline fleet get
```

- Pekerja mungkin membutuhkan waktu beberapa menit untuk memulai. Ulangi `deadline fleet get` perintah sampai Anda dapat melihat bahwa armada sedang berjalan.
- Armada yang dikelola `queueFleetAssociationsStatus` untuk layanan akan. `ACTIVE`
- SMF `autoScalingStatus` akan berubah dari `GROWING` ke `STEADY`.

Status Anda akan terlihat mirip dengan yang berikut ini:

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. Lihat log untuk pekerjaan yang Anda kirimkan. Log ini disimpan dalam log di Amazon CloudWatch Logs, bukan sistem CloudShell file.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --query "sessions[0].sessionId" \
    --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
    --log-stream-names $SESSION_ID
```

Langkah selanjutnya

Setelah membuat dan menguji armada yang dikelola layanan, Anda harus menghapus sumber daya yang Anda buat untuk menghindari biaya yang tidak perlu.

- [Bersihkan sumber daya pertanian Anda di Deadline Cloud](#) untuk mematikan sumber daya yang Anda gunakan untuk tutorial ini.

Bersihkan sumber daya pertanian Anda di Deadline Cloud

Untuk mengembangkan dan menguji beban kerja baru dan integrasi pipeline, Anda dapat terus menggunakan Deadline Cloud developer farm yang Anda buat untuk tutorial ini. Jika Anda tidak lagi membutuhkan peternakan pengembang, Anda dapat menghapus sumber dayanya termasuk peran pertanian, armada, antrian, AWS Identity and Access Management (IAM), dan log di Amazon CloudWatch Logs. Setelah Anda menghapus sumber daya ini, Anda harus memulai tutorial lagi untuk menggunakan sumber daya. Untuk informasi selengkapnya, lihat [Memulai dengan sumber daya Deadline Cloud](#).

Untuk membersihkan sumber daya pertanian pengembang

1. Pilih CloudShell tab pertama Anda, lalu hentikan semua asosiasi antrian-armada untuk antrian Anda.

```
FLEETS=$(aws deadline list-queue-fleet-associations \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --query "queueFleetAssociations[].fleetId" \
    --output text)
for FLEET_ID in $FLEETS; do
    aws deadline update-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID \
        --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

2. Buat daftar asosiasi armada antrian.

```
aws deadline list-queue-fleet-associations \
    --farm-id $DEV_FARM_ID \
```

```
--queue-id $DEV_QUEUE_ID
```

Anda mungkin perlu menjalankan kembali perintah sampai laporan output "status": "STOPPED", maka Anda dapat melanjutkan ke langkah berikutnya. Proses ini bisa memakan waktu beberapa menit untuk menyelesaikannya.

```
{
  "queueFleetAssociations": [
    {
      "queueId": "queue-abcdefgh01234567890123456789012id",
      "fleetId": "fleet-abcdefgh01234567890123456789012id",
      "status": "STOPPED",
      "createdAt": "2023-11-21T20:49:19+00:00",
      "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/MySessionName",
      "updatedAt": "2023-11-21T20:49:38+00:00",
      "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/MySessionName"
    },
    {
      "queueId": "queue-abcdefgh01234567890123456789012id",
      "fleetId": "fleet-abcdefgh01234567890123456789012id",
      "status": "STOPPED",
      "createdAt": "2023-11-21T20:32:06+00:00",
      "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/MySessionName",
      "updatedAt": "2023-11-21T20:49:39+00:00",
      "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/MySessionName"
    }
  ]
}
```

3. Hapus semua asosiasi antrian-armada untuk antrian Anda.

```
for FLEET_ID in $FLEETS; do
  aws deadline delete-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $FLEET_ID
done
```

4. Hapus semua armada yang terkait dengan antrian Anda.

```
for FLEET_ID in $FLEETS; do
    aws deadline delete-fleet \
        --farm-id $DEV_FARM_ID \
        --fleet-id $FLEET_ID
done
```

5. Hapus antrian.

```
aws deadline delete-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID
```

6. Hapus peternakan.

```
aws deadline delete-farm \
    --farm-id $DEV_FARM_ID
```

7. Hapus AWS sumber daya lain untuk peternakan Anda.

- a. Hapus peran armada AWS Identity and Access Management (IAM).

```
aws iam delete-role-policy \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --policy-name WorkerPermissions
aws iam delete-role \
    --role-name "${DEV_FARM_NAME}FleetRole"
```

- b. Hapus peran IAM antrian.

```
aws iam delete-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess
aws iam delete-role \
    --role-name "${DEV_FARM_NAME}QueueRole"
```

- c. Hapus grup CloudWatch log Amazon Logs. Setiap antrian dan armada memiliki grup log mereka sendiri.

```
aws logs delete-log-group \
    --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"
```

```
aws logs delete-log-group \  
    --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"  
aws logs delete-log-group \  
    --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
```

Konfigurasi pekerjaan menggunakan lingkungan antrian

AWS Deadline Cloud menggunakan lingkungan antrian untuk mengonfigurasi perangkat lunak pada pekerja Anda. Lingkungan memungkinkan Anda untuk melakukan tugas yang memakan waktu, seperti mengatur dan merobohkan, sekali untuk semua tugas dalam satu sesi. Ini mendefinisikan tindakan untuk dijalankan pada pekerja ketika memulai atau menghentikan sesi. Anda dapat mengonfigurasi lingkungan untuk antrian, pekerjaan yang berjalan dalam antrian, dan langkah-langkah individual untuk suatu pekerjaan.

Anda mendefinisikan lingkungan sebagai lingkungan antrian atau lingkungan kerja. Buat lingkungan antrian dengan konsol Deadline Cloud atau dengan [tenggat waktu: CreateQueueEnvironment](#) operasi dan tentukan lingkungan pekerjaan di templat pekerjaan pekerjaan yang Anda kirimkan. Mereka mengikuti spesifikasi Open Job Description (OpenJD) untuk lingkungan. Untuk detailnya, lihat <https://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment> <Environment> di spesifikasi OpenJD pada GitHub

Selain `name` dan `description`, setiap lingkungan berisi dua bidang yang menentukan lingkungan pada host. File tersebut adalah:

- `script`— Tindakan yang diambil ketika lingkungan ini dijalankan pada seorang pekerja.
- `variables`— Satu set pasangan nama/nilai variabel lingkungan yang ditetapkan saat memasuki lingkungan.

Anda harus menetapkan setidaknya satu dari `script` atau `variables`.

Anda dapat menentukan lebih dari satu lingkungan dalam template pekerjaan Anda. Setiap lingkungan diterapkan dalam urutan yang tercantum dalam template. Anda dapat menggunakan ini untuk membantu mengelola kompleksitas lingkungan Anda.

Lingkungan antrian default untuk Deadline Cloud menggunakan manajer paket conda untuk memuat perangkat lunak ke lingkungan, tetapi Anda dapat menggunakan manajer paket lainnya. Lingkungan default mendefinisikan dua parameter untuk menentukan perangkat lunak yang harus dimuat. Variabel ini diatur oleh pengirim yang disediakan oleh Deadline Cloud, meskipun Anda dapat mengaturnya dalam skrip dan aplikasi Anda sendiri yang menggunakan lingkungan default. File tersebut adalah:

- **CondaPackages**— Daftar spesifikasi paket conda yang dipisahkan ruang yang cocok untuk dipasang untuk pekerjaan itu. Misalnya, pengirim Blender akan menambahkan `blender=3.6` bingkai render di Blender 3.6.
- **CondaChannels**— Daftar saluran conda yang dipisahkan ruang untuk menginstal paket dari. Untuk armada yang dikelola layanan, paket diinstal dari saluran. `deadline-cloud` Anda dapat menambahkan saluran lain.

Topik

- [Kontrol lingkungan kerja dengan lingkungan antrian OpenJD](#)
- [Menyediakan aplikasi untuk pekerjaan Anda](#)

Kontrol lingkungan kerja dengan lingkungan antrian OpenJD

Anda dapat menentukan lingkungan yang disesuaikan untuk pekerjaan rendering Anda menggunakan lingkungan antrian. Lingkungan antrian adalah template yang mengontrol variabel lingkungan, pemetaan file, dan pengaturan lain untuk pekerjaan yang berjalan dalam antrian tertentu. Ini memungkinkan Anda untuk menyesuaikan lingkungan eksekusi untuk pekerjaan yang dikirimkan ke antrian dengan persyaratan beban kerja Anda. AWS Deadline Cloud menyediakan tiga level bersarang di mana Anda dapat menerapkan [lingkungan Open Job Description \(OpenJD\)](#): antrian, pekerjaan, dan langkah. Dengan mendefinisikan lingkungan antrian, Anda dapat memastikan kinerja yang konsisten dan dioptimalkan untuk berbagai jenis pekerjaan, merampingkan alokasi sumber daya, dan menyederhanakan manajemen antrian.

Lingkungan antrian adalah template yang Anda lampirkan ke antrian di AWS akun Anda dari konsol AWS manajemen atau menggunakan AWS CLI. Anda dapat membuat satu lingkungan untuk antrian, atau Anda dapat membuat beberapa lingkungan antrian yang diterapkan untuk membuat lingkungan eksekusi. Ini memungkinkan Anda untuk membuat dan menguji lingkungan dalam langkah-langkah untuk membantu memastikan bahwa itu bekerja dengan benar untuk pekerjaan Anda.

Lingkungan Job dan step didefinisikan dalam template pekerjaan yang Anda gunakan untuk membuat pekerjaan dalam antrian Anda. Sintaks OpenJD sama dalam berbagai bentuk lingkungan ini. Di bagian ini kami akan menunjukkannya di dalam template pekerjaan.

Topik

- [Mengatur variabel lingkungan dalam lingkungan antrian](#)
- [Mengatur jalur di lingkungan antrian](#)

- [Jalankan proses daemon latar belakang dari lingkungan antrian](#)

Mengatur variabel lingkungan dalam lingkungan antrian

[Lingkungan Open Job Description \(OpenJD\)](#) dapat mengatur variabel lingkungan yang digunakan setiap perintah tugas dalam cakupannya. Banyak aplikasi dan kerangka kerja memeriksa variabel lingkungan untuk mengontrol pengaturan fitur, tingkat logging, dan banyak lagi.

Misalnya, [Qt Framework](#) menyediakan fungsionalitas GUI untuk banyak aplikasi desktop. Ketika Anda menjalankan aplikasi ini pada host pekerja tanpa tampilan interaktif, Anda mungkin perlu mengatur variabel `QT_QPA_PLATFORM` lingkungan `offscreen` agar pekerja tidak mencari tampilan.

Dalam contoh ini, Anda akan menggunakan paket pekerjaan sampel dari direktori sampel Deadline Cloud untuk mengatur dan melihat variabel lingkungan untuk suatu pekerjaan.

Prasyarat

Lakukan langkah-langkah berikut untuk menjalankan [paket pekerjaan sampel dengan variabel lingkungan](#) dari repositori github sampel Deadline Cloud.

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di [konsol Deadline Cloud](#) untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI dan Deadline Cloud di workstation Anda, ikuti langkah-langkah [di Mengatur pengirim Deadline Cloud](#) dari panduan pengguna.
3. Gunakan `git` untuk mengkloning repositori [sampel GitHub Deadline Cloud](#).

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
...
cd deadline-cloud-samples/job_bundles
```

Jalankan sampel variabel lingkungan

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_vars` sampel.

```
deadline bundle submit job_env_vars
Submitting to Queue: MySampleQueue
```

...

2. Di monitor Deadline Cloud, Anda dapat melihat pekerjaan baru dan memantau kemajuannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia untuk menjalankan tugas pekerjaan, pekerjaan selesai dalam beberapa detik. Pilih tugas, lalu pilih opsi Lihat log di menu kanan atas panel tugas.

Di sebelah kanan adalah tiga tindakan sesi, Launch JobEnv, Launch StepEnv, dan Task run. Tampilan log di tengah jendela sesuai dengan tindakan sesi yang dipilih di sebelah kanan.

Bandingkan tindakan sesi dengan definisinya

Di bagian ini Anda menggunakan monitor Deadline Cloud untuk membandingkan tindakan sesi dengan di mana mereka didefinisikan dalam template pekerjaan. Ini berlanjut dari bagian sebelumnya.

Buka file [job_env_vars/template.yaml](#) di editor teks. Ini adalah template pekerjaan yang mendefinisikan tindakan sesi.

1. Pilih tindakan Peluncuran JobEnv sesi di monitor Deadline Cloud. Anda akan melihat output log berikut.

```
024/07/16 16:18:27-07:00
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 ----- Entering Environment: JobEnv
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 Setting: JOB_VERBOSITY=MEDIUM
2024/07/16 16:18:27-07:00 Setting: JOB_EXAMPLE_PARAM=An example parameter value
2024/07/16 16:18:27-07:00 Setting: JOB_PROJECT_ID=project-12
2024/07/16 16:18:27-07:00 Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/
path
2024/07/16 16:18:27-07:00 Setting: QT_QPA_PLATFORM=offscreen
```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```
jobEnvironments:
- name: JobEnv
  description: Job environments apply to everything in the job.
  variables:
    # When applications have options as environment variables, you can set them
    here.
```

```
JOB_VERBOSITY: MEDIUM
# You can use the value of job parameters when setting environment variables.
JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
# Some more ideas.
JOB_PROJECT_ID: project-12
JOB_ENDPOINT_URL: https://internal-host-name/some/path
# This variable lets applications using the Qt Framework run without a display
QT_QPA_PLATFORM: offscreen
```

2. Pilih tindakan Peluncuran StepEnv sesi di monitor Deadline Cloud. Anda akan melihat output log berikut.

```
2024/07/16 16:18:27-07:00
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 ----- Entering Environment: StepEnv
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 Setting: STEP_VERBOSITY=HIGH
2024/07/16 16:18:27-07:00 Setting: JOB_PROJECT_ID=step-project-12
```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```
stepEnvironments:
- name: StepEnv
  description: Step environments apply to all the tasks in the step.
  variables:
    # These environment variables are only set within this step, not other steps.
    STEP_VERBOSITY: HIGH
    # Replace a variable value defined at the job level.
    JOB_PROJECT_ID: step-project-12
```

3. Pilih tindakan sesi Task run di monitor Deadline Cloud. Anda akan melihat output berikut.

```
2024/07/16 16:18:27-07:00
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 ----- Running Task
2024/07/16 16:18:27-07:00 =====
2024/07/16 16:18:27-07:00 -----
2024/07/16 16:18:27-07:00 Phase: Setup
2024/07/16 16:18:27-07:00 -----
2024/07/16 16:18:27-07:00 Writing embedded files for Task to disk.
2024/07/16 16:18:27-07:00 Mapping: Task.File.Run -> /sessions/session-
b4bd451784674c0987be82c5f7d5642deupf6tk9/embedded_files08cdnuyt/tmpmdiajwvh
```

```

2024/07/16 16:18:27-07:00 Wrote: Run -> /sessions/session-
b4bd451784674c0987be82c5f7d5642deupf6tk9/embedded_files08cdnuyt/tmpmdiajwvh
2024/07/16 16:18:27-07:00 -----
2024/07/16 16:18:27-07:00 Phase: Running action
2024/07/16 16:18:27-07:00 -----
2024/07/16 16:18:27-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-b4bd451784674c0987be82c5f7d5642deupf6tk9/tmpiqbrsby4.sh
2024/07/16 16:18:27-07:00 Command started as pid: 2176
2024/07/16 16:18:27-07:00 Output:
2024/07/16 16:18:28-07:00 Running the task
2024/07/16 16:18:28-07:00
2024/07/16 16:18:28-07:00 Environment variables starting with JOB_*:
2024/07/16 16:18:28-07:00 JOB_ENDPOINT_URL=https://internal-host-name/some/path
2024/07/16 16:18:28-07:00 JOB_EXAMPLE_PARAM='An example parameter value'
2024/07/16 16:18:28-07:00 JOB_PROJECT_ID=step-project-12
2024/07/16 16:18:28-07:00 JOB_VERBOSITY=MEDIUM
2024/07/16 16:18:28-07:00
2024/07/16 16:18:28-07:00 Environment variables starting with STEP_*:
2024/07/16 16:18:28-07:00 STEP_VERBOSITY=HIGH
2024/07/16 16:18:28-07:00
2024/07/16 16:18:28-07:00 Done running the task
2024/07/16 16:18:28-07:00 -----
2024/07/16 16:18:28-07:00 Uploading output files to Job Attachments
2024/07/16 16:18:28-07:00 -----

```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```

script:
  actions:
    onRun:
      command: bash
      args:
        - '{{Task.File.Run}}'
  embeddedFiles:
  - name: Run
    type: TEXT
    data: |
      echo Running the task
      echo ""

      echo Environment variables starting with JOB_*:
      set | grep ^JOB_
      echo ""

```

```
echo Environment variables starting with STEP_*:  
set | grep ^STEP_  
echo ""  
  
echo Done running the task
```

Mengatur jalur di lingkungan antrian

Gunakan lingkungan OpenJD untuk memberikan perintah baru di lingkungan. Pertama Anda membuat direktori yang berisi file skrip, dan kemudian menambahkan direktori itu ke variabel PATH lingkungan sehingga executable dalam skrip Anda dapat menjalankannya tanpa perlu menentukan jalur direktori setiap kali. Daftar variabel dalam definisi lingkungan tidak menyediakan cara untuk memodifikasi variabel, jadi Anda melakukan ini dengan menjalankan skrip sebagai gantinya. Setelah skrip mengatur segalanya dan memodifikasi PATH, ia mengeksport variabel ke runtime OpenJD dengan perintah. `echo "openjd_env: PATH=$PATH"`

Prasyarat

Lakukan langkah-langkah berikut untuk menjalankan [paket pekerjaan sampel dengan variabel lingkungan](#) dari repositori github sampel Deadline Cloud.

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di [konsol Deadline Cloud](#) untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI dan Deadline Cloud di workstation Anda, ikuti langkah-langkah [di Mengatur pengirim Deadline Cloud](#) dari panduan pengguna.
3. Gunakan `git` untuk mengkloning repositori [sampel GitHub Deadline Cloud](#).

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git  
Cloning into 'deadline-cloud-samples'...  
...  
cd deadline-cloud-samples/job_bundles
```

Jalankan sampel jalur

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_with_new_command` sampel.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. Di monitor Deadline Cloud, Anda akan melihat pekerjaan baru dan dapat memantau kemajuannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia untuk menjalankan tugas pekerjaan, pekerjaan selesai dalam beberapa detik. Pilih tugas, lalu pilih opsi Lihat log di menu kanan atas panel tugas.

Di sebelah kanan adalah dua tindakan sesi, Launch RandomSleepCommand dan Task run. Penampil log di tengah jendela sesuai dengan tindakan sesi yang dipilih di sebelah kanan.

Bandingkan tindakan sesi dengan definisinya

Di bagian ini Anda menggunakan monitor Deadline Cloud untuk membandingkan tindakan sesi dengan di mana mereka didefinisikan dalam template pekerjaan. Ini berlanjut dari bagian sebelumnya.

Buka file [job_env_with_new_command/template.yaml di editor](#) teks. Bandingkan tindakan sesi dengan tempat mereka didefinisikan dalam template pekerjaan.

1. Pilih tindakan Peluncuran RandomSleepCommand sesi di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```
2024/07/16 17:25:32-07:00
2024/07/16 17:25:32-07:00 =====
2024/07/16 17:25:32-07:00 ----- Entering Environment: RandomSleepCommand
2024/07/16 17:25:32-07:00 =====
2024/07/16 17:25:32-07:00 -----
2024/07/16 17:25:32-07:00 Phase: Setup
2024/07/16 17:25:32-07:00 -----
2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
```

```

2024/07/16 17:25:32-07:00 -----
2024/07/16 17:25:32-07:00 Phase: Running action
2024/07/16 17:25:32-07:00 -----
2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
2024/07/16 17:25:32-07:00 Command started as pid: 2205
2024/07/16 17:25:32-07:00 Output:
2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
No newer logs at this moment.

```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```

jobEnvironments:
- name: RandomSleepCommand
  description: Adds a command 'random-sleep' to the environment.
  script:
    actions:
      onEnter:
        command: bash
        args:
          - "{{Env.File.Enter}}"
    embeddedFiles:
      - name: Enter
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail

          # Make a bin directory inside the session's working directory for providing
new commands
          mkdir -p '{{Session.WorkingDirectory}}/bin'

          # If this bin directory is not already in the PATH, then add it
          if ! [[ ":$PATH:" == *'{{Session.WorkingDirectory}}/bin:*' ]]; then
            export "PATH={{Session.WorkingDirectory}}/bin:$PATH"

            # This message to Open Job Description exports the new PATH value to the
environment
            echo "openjd_env: PATH=$PATH"
          fi

```

```

        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...

```

2. Pilih tindakan Peluncuran StepEnv sesi di monitor Deadline Cloud. Anda melihat output log sebagai berikut.

```

2024/07/16 17:25:33-07:00
2024/07/16 17:25:33-07:00 =====
2024/07/16 17:25:33-07:00 ----- Running Task
2024/07/16 17:25:33-07:00 =====
2024/07/16 17:25:33-07:00 -----
2024/07/16 17:25:33-07:00 Phase: Setup
2024/07/16 17:25:33-07:00 -----
2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
2024/07/16 17:25:33-07:00 -----
2024/07/16 17:25:33-07:00 Phase: Running action
2024/07/16 17:25:33-07:00 -----
2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
2024/07/16 17:25:33-07:00 Command started as pid: 2256
2024/07/16 17:25:33-07:00 Output:
2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
2024/07/16 17:26:00-07:00 -----
2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
2024/07/16 17:26:00-07:00 -----

```

3. Baris berikut dari template pekerjaan menentukan tindakan ini.

```

steps:
- name: EnvWithCommand

```

```
script:
  actions:
    onRun:
      command: bash
      args:
        - '{{Task.File.Run}}'
  embeddedFiles:
    - name: Run
      type: TEXT
      data: |
        set -xeuo pipefail

        # Run the script installed into PATH by the job environment
        random-sleep 12.5 27.5
  hostRequirements:
    attributes:
      - name: attr.worker.os.family
        anyOf:
          - linux
```

Jalankan proses daemon latar belakang dari lingkungan antrian

Dalam banyak kasus penggunaan rendering, memuat aplikasi dan data adegan dapat memakan waktu yang cukup lama. Jika pekerjaan memuat ulang mereka untuk setiap frame, itu akan menghabiskan sebagian besar waktunya untuk overhead. Seringkali mungkin untuk memuat aplikasi sekali sebagai proses daemon latar belakang, memintanya memuat data adegan, dan kemudian mengirimkannya perintah melalui komunikasi antar-proses (IPC) untuk melakukan render.

Banyak integrasi Deadline Cloud open source menggunakan pola ini. Proyek Open Job Description menyediakan [pustaka runtime adaptor](#) dengan pola IPC yang kuat pada semua sistem operasi yang didukung.

Untuk mendemonstrasikan pola ini, ada [bundel pekerjaan sampel mandiri](#) yang menggunakan Python dan kode bash untuk mengimplementasikan daemon latar belakang dan IPC untuk tugas-tugas untuk berkomunikasi dengannya. Daemon diimplementasikan dengan Python, dan mendengarkan SIGUSR1 sinyal POSIX kapan harus memproses tugas. Detail tugas diteruskan ke daemon dalam file JSON tertentu, dan hasil menjalankan tugas dikembalikan sebagai file JSON lain.

Prasyarat

Lakukan langkah-langkah berikut untuk menjalankan [sample job bundle dengan proses daemon](#) dari repositori github sampel Deadline Cloud.

1. Jika Anda tidak memiliki peternakan Deadline Cloud dengan antrian dan armada Linux terkait, ikuti pengalaman orientasi terpandu di [konsol Deadline Cloud](#) untuk membuatnya dengan pengaturan default.
2. Jika Anda tidak memiliki monitor Deadline Cloud CLI dan Deadline Cloud di workstation Anda, ikuti langkah-langkah [di Mengatur pengirim Deadline Cloud](#) dari panduan pengguna.
3. Gunakan git untuk mengkloning repositori [sampel GitHub Deadline Cloud](#).

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
...
cd deadline-cloud-samples/job_bundles
```

Jalankan sampel daemon

1. Gunakan Deadline Cloud CLI untuk mengirimkan `job_env_daemon_process` sampel.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
...
cd deadline-cloud-samples/job_bundles
```

2. Dalam aplikasi monitor Deadline Cloud, Anda akan melihat pekerjaan baru dan dapat memantau kemajuannya. Setelah Linux armada yang terkait dengan antrian memiliki pekerja yang tersedia untuk menjalankan tugas pekerjaan, selesai dalam waktu sekitar satu menit. Dengan salah satu tugas yang dipilih, pilih opsi Lihat log di menu kanan atas panel tugas.

Di sebelah kanan ada dua tindakan sesi, Launch DaemonProcess dan Task run. Penampil log di tengah jendela sesuai dengan tindakan sesi yang dipilih di sebelah kanan.

Pilih opsi Lihat log untuk semua tugas. Garis waktu menunjukkan sisa tugas yang berjalan sebagai bagian dari sesi, dan Shut down DaemonProcess tindakan yang keluar dari lingkungan.

Lihat log daemon

1. Di bagian ini Anda menggunakan monitor Deadline Cloud untuk membandingkan tindakan sesi dengan di mana mereka didefinisikan dalam template pekerjaan. Ini berlanjut dari bagian sebelumnya.

Buka file [job_env_daemon_process/template.yaml di editor](#) teks. Bandingkan tindakan sesi dengan tempat mereka didefinisikan dalam template pekerjaan.

2. Pilih tindakan Launch DaemonProcess sesi di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```
2024/07/17 16:27:20-07:00
2024/07/17 16:27:20-07:00 =====
2024/07/17 16:27:20-07:00 ----- Entering Environment: DaemonProcess
2024/07/17 16:27:20-07:00 =====
2024/07/17 16:27:20-07:00 -----
2024/07/17 16:27:20-07:00 Phase: Setup
2024/07/17 16:27:20-07:00 -----
2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
```

```

2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
2024/07/17 16:27:20-07:00 -----
2024/07/17 16:27:20-07:00 Phase: Running action
2024/07/17 16:27:20-07:00 -----
2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
2024/07/17 16:27:20-07:00 Command started as pid: 2187
2024/07/17 16:27:20-07:00 Output:
2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh

```

Baris berikut dari template pekerjaan menentukan tindakan ini.

```

stepEnvironments:
- name: DaemonProcess
  description: Runs a daemon process for the step's tasks to share.
  script:
    actions:
      onEnter:
        command: bash
        args:
          - "{{Env.File.Enter}}"
      onExit:
        command: bash
        args:
          - "{{Env.File.Exit}}"
    embeddedFiles:
      - name: Enter
        filename: enter-daemon-process-env.sh
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail

          DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
          echo "openjd_env: DAEMON_LOG=${DAEMON_LOG}"
          nohup python {{Env.File.DaemonScript}} > ${DAEMON_LOG} 2>&1 &

```

```

        echo "openjd_env: DAEMON_PID=$!"
        echo "openjd_env:
DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

        echo 0 > 'daemon_log_cursor.txt'
        ...

```

3. Pilih salah satu tindakan Task run: N session di monitor Deadline Cloud. Anda akan melihat output log sebagai berikut.

```

2024/07/17 16:27:22-07:00
2024/07/17 16:27:22-07:00 =====
2024/07/17 16:27:22-07:00 ----- Running Task
2024/07/17 16:27:22-07:00 =====
2024/07/17 16:27:22-07:00 Parameter values:
2024/07/17 16:27:22-07:00 Frame(INT) = 2
2024/07/17 16:27:22-07:00 -----
2024/07/17 16:27:22-07:00 Phase: Setup
2024/07/17 16:27:22-07:00 -----
2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
2024/07/17 16:27:22-07:00 -----
2024/07/17 16:27:22-07:00 Phase: Running action
2024/07/17 16:27:22-07:00 -----
2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
2024/07/17 16:27:22-07:00 Command started as pid: 2301
2024/07/17 16:27:22-07:00 Output:
2024/07/17 16:27:23-07:00 Daemon PID is 2223
2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 === Previous output from daemon
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 Sending command to daemon
2024/07/17 16:27:23-07:00 Received task result:
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00     "result": "SUCCESS",
2024/07/17 16:27:23-07:00     "processedTaskCount": 1,

```

```

2024/07/17 16:27:23-07:00 "randomValue": 0.2578537967668988,
2024/07/17 16:27:23-07:00 "failureRate": 0.1
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 === Daemon log from running the task
2024/07/17 16:27:23-07:00 Loading the task details file
2024/07/17 16:27:23-07:00 Received task details:
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00 "pid": 2329,
2024/07/17 16:27:23-07:00 "frame": 2
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00 Processing frame number 2
2024/07/17 16:27:23-07:00 Writing result
2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 -----
2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
2024/07/17 16:27:23-07:00 -----

```

Baris berikut dari template pekerjaan adalah apa yang menentukan tindakan ini. ``langkah:

```

steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
      - name: Frame
        type: INT
        range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
          - '{{Task.File.Run}}'
    embeddedFiles:
      - name: Run
        filename: run-task.sh

```

```
type: TEXT
data: |
  # This bash script sends a task to the background daemon process,
  # then waits for it to respond with the output result.

  set -euo pipefail

  source "$DAEMON_BASH_HELPER_SCRIPT"

  echo "Daemon PID is $DAEMON_PID"
  echo "Daemon log file is $DAEMON_LOG"

  print_daemon_log "Previous output from daemon"

  send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
  wait_for_daemon_task_result

  echo Received task result:
  echo "$TASK_RESULT" | jq .

  print_daemon_log "Daemon log from running the task"

hostRequirements:
  attributes:
    - name: attr.worker.os.family
      anyOf:
        - linux
```

Menyediakan aplikasi untuk pekerjaan Anda

Anda dapat menggunakan lingkungan antrian untuk memuat aplikasi untuk memproses pekerjaan Anda. Saat membuat armada yang dikelola layanan menggunakan konsol Deadline Cloud, Anda memiliki opsi untuk membuat lingkungan antrian yang menggunakan manajer paket conda untuk memuat aplikasi.

Jika Anda ingin menggunakan manajer paket yang berbeda, Anda dapat membuat lingkungan antrian untuk manajer itu. Untuk contoh menggunakan Rez, lihat [Gunakan manajer paket yang berbeda](#).

Deadline Cloud menyediakan saluran conda untuk memuat pilihan aplikasi rendering ke lingkungan Anda. Mereka mendukung pengirim yang disediakan Deadline Cloud untuk aplikasi pembuatan konten digital.

Anda juga dapat memuat perangkat lunak untuk conda-forge untuk digunakan dalam pekerjaan Anda. Contoh berikut menunjukkan templat pekerjaan menggunakan lingkungan antrian yang disediakan oleh Deadline Cloud untuk memuat aplikasi sebelum menjalankan pekerjaan.

Topik

- [Mendapatkan aplikasi dari saluran conda](#)
- [Gunakan manajer paket yang berbeda](#)

Mendapatkan aplikasi dari saluran conda

Anda dapat membuat lingkungan antrian khusus untuk pekerja Deadline Cloud Anda yang menginstal perangkat lunak pilihan Anda. Contoh lingkungan antrian ini memiliki perilaku yang sama dengan lingkungan yang digunakan oleh konsol untuk armada yang dikelola layanan. Ini menjalankan conda langsung untuk menciptakan lingkungan.

Lingkungan menciptakan lingkungan virtual conda baru untuk setiap sesi Deadline Cloud yang berjalan pada pekerja, dan kemudian menghapus lingkungan ketika selesai.

Conda menyimpan paket yang diunduh sehingga tidak perlu diunduh lagi, tetapi setiap sesi harus menautkan semua paket ke lingkungan.

Lingkungan mendefinisikan tiga skrip yang berjalan saat Deadline Cloud memulai sesi pada pekerja. Skrip pertama berjalan ketika onEnter tindakan dipanggil. Ini memanggil dua lainnya untuk mengatur variabel lingkungan. Ketika skrip selesai berjalan, lingkungan conda tersedia dengan semua variabel lingkungan yang ditentukan ditetapkan.

Untuk versi terbaru dari contoh, lihat [conda_queue_env_console_equivalent.yaml](#) di repositori pada [deadline-cloud-samples](#) GitHub

Jika Anda ingin menggunakan aplikasi yang tidak tersedia di saluran conda, Anda dapat membuat saluran conda di Amazon S3 dan kemudian membuat paket Anda sendiri untuk aplikasi itu. Lihat [Buat saluran conda menggunakan S3](#) untuk mempelajari selengkapnya.

Dapatkan pustaka open source dari conda-forge

Bagian ini menjelaskan cara menggunakan pustaka sumber terbuka dari conda-forge saluran. Contoh berikut adalah template pekerjaan yang menggunakan paket `polars` Python.

Pekerjaan menetapkan `CondaChannels` parameter `CondaPackages` dan yang ditentukan dalam lingkungan antrian yang memberi tahu Deadline Cloud tempat mendapatkan paket.

Bagian dari template pekerjaan yang menetapkan parameter adalah:

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
  to handle this.
  type: STRING
  default: polars
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
  Environment to handle this.
  type: STRING
  default: conda-forge
```

Untuk versi terbaru dari contoh template pekerjaan lengkap, lihat [stage_1_self_contained_template/template.yaml](#). Untuk versi terbaru dari lingkungan antrian yang memuat paket conda, lihat [conda_queue_env_console_equivalent.yaml](#) di repositori pada [deadline-cloud-samples](#) GitHub

Dapatkan Blender dari saluran deadline-cloud

Contoh berikut menunjukkan template pekerjaan yang mendapat Blender dari saluran deadline-cloud conda. Saluran ini mendukung pengirim yang disediakan Deadline Cloud untuk perangkat lunak pembuatan konten digital, meskipun Anda dapat menggunakan saluran yang sama untuk memuat perangkat lunak untuk Anda gunakan sendiri.

Untuk daftar perangkat lunak yang disediakan oleh deadline-cloud channel, lihat [Lingkungan antrian default](#) di Panduan Pengguna Cloud AWS Batas Waktu.

Pekerjaan ini menetapkan CondaPackages parameter yang ditentukan dalam lingkungan antrian untuk memberi tahu Deadline Cloud untuk memuat Blender ke lingkungan.

Bagian dari template pekerjaan yang menetapkan parameter adalah:

```
- name: CondaPackages
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
    channel.
```

Untuk versi terbaru dari contoh template pekerjaan lengkap, lihat [blender_render/template.yaml](#). Untuk versi terbaru dari lingkungan antrean yang memuat paket conda, lihat [conda_queue_env_console_equivalent.yaml](#) di repositori [deadline-cloud-samples](#) GitHub.

Gunakan manajer paket yang berbeda

Manajer paket default untuk Deadline Cloud adalah conda. Jika Anda perlu menggunakan manajer paket yang berbeda, seperti Rez, Anda dapat membuat lingkungan antrian khusus yang berisi skrip yang menggunakan manajer paket Anda sebagai gantinya.

Contoh lingkungan antrian ini memberikan perilaku yang sama dengan lingkungan yang digunakan oleh konsol untuk armada yang dikelola layanan. Ini menggantikan manajer paket conda dengan Rez.

Lingkungan mendefinisikan tiga skrip yang berjalan saat Deadline Cloud memulai sesi pada pekerja. Skrip pertama berjalan ketika `onEnter` tindakan dipanggil. Ini memanggil dua lainnya untuk mengatur variabel lingkungan. Ketika skrip selesai berjalan, Rez lingkungan tersedia dengan semua variabel lingkungan yang ditentukan ditetapkan.

Contoh ini mengasumsikan bahwa Anda memiliki armada yang dikelola pelanggan yang menggunakan sistem file bersama untuk paket Rez.

Untuk versi terbaru dari contoh, lihat [rez_queue_env.yaml](#) di repositori [deadline-cloud-samples](#) GitHub.

Buat saluran conda menggunakan S3

Jika Anda memiliki paket khusus untuk aplikasi yang tidak tersedia di `deadline-cloud` atau `conda-forge` saluran, Anda dapat membuat saluran conda yang berisi paket yang digunakan lingkungan Anda. Anda dapat menyimpan paket di bucket Amazon S3 dan menggunakan AWS Identity and Access Management izin untuk mengontrol akses ke saluran.

Anda dapat menggunakan antrean Deadline Cloud untuk membangun paket untuk saluran conda Anda agar lebih mudah memperbarui dan memelihara paket aplikasi.

Manfaat utama dari pendekatan ini adalah antrian pembuatan paket Anda dapat membuat paket untuk beberapa sistem operasi yang berbeda, dan dengan atau tanpa dukungan CUDA. Sebagai perbandingan, jika Anda membangun paket di workstation Anda, Anda perlu membuat dan mengelola workstation yang berbeda untuk kasus ini.

Contoh berikut menunjukkan cara membuat saluran conda yang menyediakan dan aplikasi untuk lingkungan Anda. Aplikasi dalam contoh adalah Blender 4.2, tetapi salah satu aplikasi terintegrasi Deadline Cloud dapat digunakan.

Anda dapat menggunakan AWS CloudFormation template untuk membuat pertanian Deadline Cloud yang menyertakan antrian pembuatan paket, atau Anda dapat mengikuti petunjuk di bawah ini untuk membuat contoh pertanian sendiri. Untuk AWS CloudFormation template, lihat [Starter AWS Deadline Cloud farm](#) di repositori sampel Deadline Cloud pada GitHub

Topik

- [Buat antrian pembuatan paket](#)
- [Konfigurasi izin antrian produksi untuk paket conda kustom](#)
- [Menambahkan saluran conda ke lingkungan antrian](#)
- [Buat paket conda untuk aplikasi](#)
- [Buat resep conda build untuk Blender](#)
- [Buat resep conda build untuk Autodesk Maya](#)
- [Buat resep conda build untuk Autodesk Maya to Arnold \(MtoA\) plugin](#)

Buat antrian pembuatan paket

Dalam contoh ini Anda membuat antrian Deadline Cloud untuk membangun Blender 4.2 aplikasi. Ini menyederhanakan pengiriman paket yang sudah jadi ke bucket Amazon S3 yang digunakan sebagai saluran conda dan memungkinkan Anda menggunakan armada yang ada untuk membuat paket. Ini mengurangi jumlah komponen infrastruktur untuk dikelola.

Ikuti petunjuk di [Buat antrian](#) di Panduan Pengguna Cloud Deadline. Lakukan perubahan berikut:

- Pada langkah 5, pilih bucket S3 yang ada. Tentukan nama folder root seperti **DeadlineCloudPackageBuild** agar artefak build tetap terpisah dari lampiran Deadline Cloud normal Anda.
- Pada langkah 6, Anda dapat mengaitkan antrian pembuatan paket dengan armada yang sudah ada, atau Anda dapat membuat armada yang sama sekali baru jika armada Anda saat ini tidak cocok.
- Pada langkah 9, buat peran layanan baru untuk antrian pembuatan paket Anda. Anda akan memodifikasi izin untuk memberikan antrian izin yang diperlukan untuk mengunggah paket dan mengindeks ulang saluran conda.

Konfigurasikan izin antrian pembuatan paket

Untuk mengizinkan antrian build paket mengakses /Conda awalan di bucket S3 antrian, Anda harus memodifikasi peran antrian untuk memberikan akses baca/tulis. Peran tersebut memerlukan izin berikut agar pekerjaan pembuatan paket dapat mengunggah paket baru dan mengindeks ulang saluran.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrian pembuatan paket.
2. Pilih peran layanan antrian, lalu pilih Edit antrian.

3. Gulir ke bagian Peran layanan antrian, lalu pilih Lihat peran ini di konsol IAM.
4. Dari daftar kebijakan izin, pilih antrian AmazonDeadlineCloudQueuePolicy untuk Anda.
5. Dari tab Izin, pilih Edit.
6. Perbarui peran layanan antrian ke yang berikut ini. Ganti *amzn-s3-demo-bucket* dan *111122223333* dengan ember dan akun Anda sendiri.

```
{
  "Effect": "Allow",
  "Sid": "CustomCondaChannelReadWrite",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "111122223333"
    }
  }
},
```

Konfigurasi izin antrian produksi untuk paket conda kustom

Antrian produksi Anda memerlukan izin hanya-baca ke /Conda awalan di bucket S3 antrian. Buka halaman AWS Identity and Access Management (IAM) untuk peran yang terkait dengan antrian produksi dan ubah kebijakan dengan yang berikut:

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrian pembuatan paket.
2. Pilih peran layanan antrian, lalu pilih Edit antrian.
3. Gulir ke bagian Peran layanan antrian, lalu pilih Lihat peran ini di konsol IAM.
4. Dari daftar kebijakan izin, pilih antrian AmazonDeadlineCloudQueuePolicy untuk Anda.
5. Dari tab Izin, pilih Edit.

6. Tambahkan bagian baru ke peran layanan antrian seperti berikut ini. Ganti *amzn-s3-demo-bucket* dan *111122223333* dengan ember dan akun Anda sendiri.

```
{
  "Effect": "Allow",
  "Sid": "CustomCondaChannelReadOnly",
  "Action": [
    "s3:GetObject",
    "s3:ListBucket"
  ],
  "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "111122223333"
    }
  }
},
```

Menambahkan saluran conda ke lingkungan antrian

Untuk menggunakan saluran conda S3, Anda perlu menambahkan lokasi `s3://amzn-s3-demo-bucket/Conda/Default` saluran ke `CondaChannels` parameter pekerjaan yang Anda kirimkan ke Deadline Cloud. Pengirim yang dilengkapi dengan Deadline Cloud menyediakan bidang untuk menentukan saluran dan paket conda kustom.

Anda dapat menghindari memodifikasi setiap pekerjaan dengan mengedit lingkungan antrian conda untuk antrian produksi Anda. Untuk antrian yang dikelola layanan, gunakan prosedur berikut:

1. Buka konsol Deadline Cloud dan arahkan ke halaman detail antrian untuk antrean produksi.
2. Pilih tab lingkungan.
3. Pilih lingkungan antrian Conda, lalu pilih Edit.
4. Pilih editor JSON, dan kemudian dalam skrip, temukan definisi parameter `untukCondaChannels`.
5. Edit baris `default: "deadline-cloud"` sehingga dimulai dengan saluran conda S3 yang baru dibuat:

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Armada yang dikelola layanan memungkinkan prioritas saluran yang ketat untuk conda secara default, menggunakan saluran S3 baru menghentikan conda menggunakan saluran. deadline-cloud Pekerjaan apa pun yang berhasil diselesaikan menggunakan blender=3.6 dari deadline-cloud saluran akan gagal sekarang setelah Anda menggunakan Blender 4.2.

Untuk armada yang dikelola pelanggan, Anda dapat mengaktifkan penggunaan paket conda dengan menggunakan salah satu contoh [lingkungan antrian Conda](#) dalam sampel Deadline Cloud GitHub repositori.

Buat paket conda untuk aplikasi

Anda dapat menggabungkan seluruh aplikasi, termasuk dependensi, ke dalam paket conda. Paket Deadline Cloud menyediakan di [saluran deadline-cloud](#) untuk armada yang dikelola layanan menggunakan pendekatan pengemasan ulang biner ini. Ini mengatur file yang sama sebagai instalasi agar sesuai dengan lingkungan virtual conda.

Saat mengemas ulang aplikasi untuk conda, ada dua tujuan:

- Sebagian besar file untuk aplikasi harus terpisah dari struktur lingkungan virtual conda utama. Lingkungan kemudian dapat mencampur aplikasi dengan paket dari sumber lain seperti [conda-forge](#).
- Ketika lingkungan virtual conda diaktifkan, aplikasi harus tersedia dari variabel lingkungan PATH.

Untuk mengemas ulang aplikasi untuk conda

1. Untuk mengemas ulang aplikasi untuk conda, tulis resep build conda yang menginstal aplikasi ke dalam subdirektori seperti. `$CONDA_PREFIX/opt/<application-name>` Ini memisahkannya dari direktori awalan standar seperti `bin lib`
2. Kemudian, tambahkan symlink atau luncurkan skrip `$CONDA_PREFIX/bin` untuk menjalankan binari aplikasi.

Atau, buat skrip `activate.d` yang akan dijalankan conda `activate` perintah untuk menambahkan direktori biner aplikasi ke PATH. Pada Windows, di mana symlink tidak didukung

di mana pun lingkungan dapat dibuat, gunakan peluncuran aplikasi atau aktifkan skrip.d sebagai gantinya.

3. Beberapa aplikasi bergantung pada pustaka yang tidak diinstal secara default pada armada yang dikelola layanan Deadline Cloud. Misalnya, sistem jendela X11 biasanya tidak diperlukan untuk pekerjaan non-interaktif, tetapi beberapa aplikasi masih mengharuskannya untuk berjalan tanpa antarmuka grafis. Anda harus memberikan dependensi tersebut dalam paket yang Anda buat.
4. Pastikan Anda mengikuti perjanjian hak cipta dan lisensi untuk aplikasi yang Anda paket. Sebaiknya gunakan bucket Amazon S3 pribadi untuk saluran conda Anda guna mengontrol distribusi dan membatasi akses paket ke peternakan Anda.

Buat resep conda build untuk Blender

Anda dapat menggunakan aplikasi yang berbeda untuk membuat resep build conda. Blender gratis untuk digunakan dan mudah dikemas dengan conda. Bagian Blender Foundation menyediakan [arsip aplikasi](#) untuk beberapa sistem operasi. Kami membuat contoh resep build conda yang menggunakan file Windows.zip dan Linux .tar.xz. Di bagian ini, pelajari cara menggunakan [Blender 4.2 resep](#) build conda.

File [deadline-cloud.yaml](#) menentukan platform conda dan metadata lainnya untuk mengirimkan pekerjaan paket ke Deadline Cloud. Resep ini mencakup informasi arsip sumber lokal untuk menunjukkan cara kerjanya. Platform conda linux-64 diatur untuk membangun pengiriman pekerjaan default agar sesuai dengan konfigurasi yang paling umum. Tenggat waktu-cloud.yaml terlihat mirip dengan yang berikut ini:

```
condaPlatforms:
  - platform: linux-64
    defaultSubmit: true
    sourceArchiveFilename: blender-4.2.1-linux-x64.tar.xz
    sourceDownloadInstructions: 'Run "curl -LO https://download.blender.org/release/Blender4.2/blender-4.2.1-linux-x64.tar.xz"'
  - platform: win-64
    defaultSubmit: false
    sourceArchiveFilename: blender-4.2.1-windows-x64.zip
    sourceDownloadInstructions: 'Run "curl -LO https://download.blender.org/release/Blender4.2/blender-4.2.1-windows-x64.zip"'
```

Tinjau file di recipe direktori. [Metadata untuk resepnya ada di resep/meta.yaml](#). Anda juga dapat membaca dokumentasi conda build [meta.yaml](#) untuk mempelajari lebih lanjut, seperti bagaimana file

tersebut merupakan template untuk menghasilkan YAMAL. Template digunakan untuk menentukan nomor versi hanya sekali, dan untuk memberikan nilai yang berbeda berdasarkan sistem operasi.

Anda dapat meninjau opsi build yang dipilih meta.yaml untuk menonaktifkan berbagai pemeriksaan penautan relokasi biner dan objek bersama dinamis (DSO). Opsi ini mengontrol cara kerja paket saat diinstal ke lingkungan virtual conda di awalan direktori apa pun. Nilai default menyederhanakan pengemasan setiap pustaka dependensi ke dalam paket terpisah, tetapi ketika mengemas ulang aplikasi biner, Anda perlu mengubahnya.

Jika aplikasi yang Anda kemas memerlukan pustaka ketergantungan tambahan atau Anda mengemas plugin untuk aplikasi secara terpisah, Anda mungkin mengalami kesalahan DSO. Kesalahan ini terjadi ketika ketergantungan tidak ada di jalur pencarian perpustakaan untuk executable atau pustaka yang membutuhkannya. Aplikasi bergantung pada pustaka yang berada di jalur yang ditentukan secara global, seperti /lib atau /usr/lib, ketika diinstal pada sistem. Namun, karena lingkungan virtual conda dapat ditempatkan di mana saja, tidak ada jalur absolut untuk digunakan. Conda menggunakan fitur RPATH relatif, yang keduanya Linux and macOS dukung, untuk menangani ini. Lihat dokumentasi conda build tentang [Membuat paket dapat dipindahkan untuk informasi lebih lanjut](#).

Blender tidak memerlukan penyesuaian RPATH, karena arsip aplikasi dibangun dengan mempertimbangkan hal ini. Untuk aplikasi yang membutuhkannya, Anda dapat menggunakan alat yang sama seperti yang dilakukan conda build: patchelf di Linux dan install_name_tool di macOS.

Selama pembuatan paket, skrip [build.sh](#) atau [build_win.sh](#) (dipanggil oleh build.bat) berjalan untuk menginstal file ke dalam lingkungan yang disiapkan dengan dependensi paket. Skrip ini menyalin file instalasi, membuat symlink dari \$PREFIX/bin, dan mengatur skrip aktivasi. Pada Windows, itu tidak membuat symlink melainkan menambahkan direktori Blender ke PATH dalam skrip aktivasi.

Kami menggunakan bash alih-alih cmd.exe file.bat untuk Windows bagian dari resep build conda, karena ini memberikan lebih banyak konsistensi di seluruh skrip build. Lihat rekomendasi [panduan pengembang Deadline Cloud](#) tentang portabilitas beban kerja untuk tips penggunaan bash Windows. Jika Anda telah menginstal [git untuk Windows](#) untuk mengkloning repositori [deadline-cloud-samples](#)git, Anda sudah memiliki akses ke bash

Dokumentasi [variabel lingkungan build conda](#) mencantumkan nilai yang tersedia untuk digunakan dalam skrip build. Nilai-nilai ini termasuk \$SRC_DIR untuk data arsip sumber, \$PREFIX untuk direktori instalasi, \$RECIPE_DIR untuk mengakses file lain dari resep, \$PKG_NAME dan \$PKG_VERSION untuk nama paket dan versi, dan \$target_platform untuk platform conda target.

Kirim Blender 4.2 paket pekerjaan

Anda dapat membangun sendiri Blender 4.2 paket conda untuk merender pekerjaan, dengan mengunduh Blender arsipkan dan kemudian mengirimkan pekerjaan ke antrian pembuatan paket. Antrian mengirimkan pekerjaan ke armada terkait untuk membangun paket dan mengindeks ulang saluran conda.

Instruksi ini menggunakan git dari shell yang kompatibel dengan bash untuk mendapatkan pekerjaan pembuatan paket OpenJD dan beberapa resep conda dari sampel Deadline Cloud [GitHub repositori](#). Anda juga memerlukan hal berikut:

- Jika Anda menggunakan Windows, versi bash, git BASH, diinstal saat Anda menginstal git.
 - Anda harus menginstal [Deadline Cloud CLI](#).
 - Anda harus masuk ke [monitor Deadline Cloud](#).
1. Buka GUI konfigurasi Deadline Cloud menggunakan perintah berikut dan atur farm default dan antrian ke antrian pembuatan paket Anda.

```
deadline config gui
```

2. Gunakan perintah berikut untuk mengkloning sampel Deadline Cloud GitHub repositori.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. Ubah ke conda_recipes direktori di deadline-cloud-samples direktori.

```
cd deadline-cloud-samples/conda_recipes
```

4. Jalankan skrip yang disebut submit-package-job. Skrip memberikan instruksi untuk mengunduh Blender pertama kali Anda menjalankan skrip.

```
./submit-package-job blender-4.2/
```

5. Ikuti instruksi untuk mengunduh Blender. Ketika Anda memiliki arsip, jalankan submit-package-job skrip lagi.

```
./submit-package-job blender-4.2/
```

Setelah Anda mengirimkan pekerjaan, gunakan monitor Deadline Cloud untuk melihat kemajuan dan status pekerjaan saat dijalankan.

Kiri bawah monitor menunjukkan dua langkah pekerjaan, membangun paket dan kemudian mengindeks ulang. Kanan bawah menunjukkan langkah-langkah individu untuk setiap tugas. Dalam contoh ini, ada satu langkah untuk setiap tugas.

The screenshot shows the 'Job monitor' interface for a job named 'CondaBuild: blender-4.1'. The job is in a 'Succeeded' state with 100% progress (2/2). Below the job summary, there are two panels: 'Steps (1/2)' and 'Tasks (1/1)'. The 'Steps' panel shows two steps: 'PackageBuild' and 'ReindexCo...'. The 'Tasks' panel shows a single task 'Succeeded'.

Job name	Progress	Status	Duration	Priority	Failed tasks	Create time	Start time	End time
CondaBuild: blender-4.1	100% (2/2)	✓ Succeeded	00:22:05	50	0	45m 43s ago	43m 15s ago	21m 9s ago

Step name	Progress	Status	Duration	Failed ta...	Sta
PackageBuild	100% (1/1)	✓ Succeeded	00:20:53	0	43m
ReindexCo...	100% (1/1)	✓ Succeeded	00:00:54	0	22m

Status	Duration	Retries / Ma...	Start time	End time
✓ Succeeded	00:19:55	0/1	42m 18s ago	22m 22s ago

Di kiri bawah monitor adalah dua langkah pekerjaan, membangun paket dan kemudian mengindeks ulang saluran conda. Di kanan bawah adalah tugas individu untuk setiap langkah. Dalam contoh ini hanya ada satu tugas untuk setiap langkah.

Ketika Anda mengklik kanan pada tugas untuk langkah pembuatan paket dan memilih Lihat log, monitor akan menampilkan daftar tindakan sesi yang menunjukkan bagaimana tugas dijadwalkan pada pekerja. Tindakannya adalah:

- Sinkronkan lampiran - Tindakan ini menyalin lampiran pekerjaan input atau memasang sistem file virtual, tergantung pada pengaturan yang digunakan untuk sistem file lampiran pekerjaan.
- Luncurkan Conda - Tindakan ini berasal dari lingkungan antrian yang ditambahkan secara default saat Anda membuat antrian. Pekerjaan tidak menentukan paket conda apa pun, sehingga selesai dengan cepat dan tidak membuat lingkungan virtual conda.
- Launch CondaBuild Env — Tindakan ini menciptakan lingkungan virtual conda kustom yang mencakup perangkat lunak yang diperlukan untuk membangun paket conda dan mengindeks ulang saluran. Ini menginstal dari saluran [conda-forge](#).
- Task run - Tindakan ini membangun Blender paket dan mengunggah hasilnya ke Amazon S3.

Saat tindakan berjalan, mereka mengirim log dalam format terstruktur ke Amazon CloudWatch. Saat pekerjaan selesai, pilih Lihat log untuk semua tugas guna melihat log tambahan tentang penyiapan dan penguraian lingkungan tempat pekerjaan berjalan.

Uji paket Anda dengan Blender 4.2 pekerjaan render

Setelah Anda memiliki Blender 4.2 paket dibangun dan antrian produksi Anda dikonfigurasi untuk menggunakan saluran conda S3, Anda dapat mengirimkan pekerjaan untuk dirender dengan paket. Jika Anda tidak memiliki Blender adegan, unduh Blender 3.5 - Adegan Dapur Nyaman dari [Blender](#) halaman file demo.

Sampel Batas Waktu Cloud GitHub repositori yang Anda unduh sebelumnya berisi contoh pekerjaan untuk merender Blender adegan menggunakan perintah berikut:

```
deadline bundle submit blender_render \  
  -p CondaPackages=blender=4.2 \  
  -p BlenderSceneFile=/path/to/downloaded/blender-3.5-splash.blend \  
  -p Frames=1
```

Anda dapat menggunakan monitor Deadline Cloud untuk melacak kemajuan pekerjaan Anda:

1. Di monitor, pilih tugas untuk pekerjaan yang Anda kirimkan, lalu pilih opsi untuk melihat log.
2. Di sisi kanan tampilan log, pilih tindakan sesi Launch Conda.

Anda dapat melihat bahwa tindakan mencari Blender 4.2 di dua saluran conda yang dikonfigurasi untuk lingkungan antrian, dan menemukan paket di saluran S3.

Buat resep conda build untuk Autodesk Maya

Anda dapat mengemas aplikasi komersial sebagai paket conda. Di [Buat resep build conda untuk Blender](#), Anda belajar cara mengemas aplikasi yang tersedia sebagai file arsip yang dapat dipindahkan sederhana dan di bawah persyaratan lisensi sumber terbuka. Aplikasi komersial sering didistribusikan melalui installer dan mungkin memiliki sistem manajemen lisensi untuk bekerja dengan.

Daftar berikut dibangun di atas dasar-dasar yang tercakup dalam [Buat paket conda untuk aplikasi](#) dengan persyaratan yang umumnya terlibat dengan pengemasan aplikasi komersial. Detail dalam sub-peluru menggambarkan bagaimana Anda dapat menerapkan pedoman Maya.

- Memahami hak lisensi dan pembatasan aplikasi. Anda mungkin perlu mengonfigurasi sistem manajemen lisensi. Jika aplikasi tidak menyertakan penegakan hukum, Anda perlu mengonfigurasi pertanian Anda sesuai dengan hak Anda.
- Baca [Autodesk Manfaat Berlangganan FAQ tentang Hak Cloud](#) untuk memahami hak cloud Maya yang mungkin berlaku untuk Anda. Konfigurasi Deadline Cloud farm Anda seperlunya.
- Autodesk produk bergantung pada file yang disebut `ProductInformation.pit`. Sebagian besar konfigurasi file ini memerlukan akses administrator ke sistem, yang tidak tersedia pada armada yang dikelola layanan. Fitur produk untuk klien tipis menyediakan cara yang dapat direlokasi untuk menangani hal ini. Lihat [Lisensi Klien Tipis untuk Maya dan MotionBuilder](#) untuk mempelajari lebih lanjut.
- Beberapa aplikasi bergantung pada pustaka yang tidak diinstal pada host pekerja armada yang dikelola layanan, sehingga paket harus menyediakannya. Ini bisa berada di dalam paket aplikasi secara langsung, atau ditempatkan dalam paket ketergantungan terpisah.
- Maya tergantung pada sejumlah perpustakaan seperti itu, termasuk freetype dan fontconfig. Ketika pustaka ini tersedia di manajer paket sistem, seperti di `dnf` untuk AL2 023, Anda dapat menggunakannya sebagai sumber untuk aplikasi. Karena paket RPM ini tidak dibuat untuk dapat direlokasi, Anda perlu menggunakan alat seperti `patchelf` untuk memastikan dependensi diselesaikan dalam Maya awalan instalasi.
- Instalasi mungkin memerlukan akses administrator. Karena armada yang dikelola layanan tidak menyediakan akses administrator, Anda perlu melakukan instalasi pada sistem dengan akses ini. Kemudian, buat arsip file yang diperlukan untuk pekerjaan pembuatan paket yang akan digunakan.
- Bagian Windows installer untuk Maya memerlukan akses administrator, jadi membangun paket conda untuk itu melibatkan proses manual untuk terlebih dahulu membuat arsip semacam itu.
- Konfigurasi aplikasi, termasuk bagaimana plugin mendaftar dengannya, dapat didefinisikan di sistem operasi atau tingkat pengguna. Ketika ditempatkan di lingkungan virtual conda, plugin memerlukan cara untuk mengintegrasikan dengan aplikasi dengan cara yang terkandung dan tidak pernah menulis file atau data lain di luar awalan lingkungan virtual. Kami sarankan Anda mengatur ini dari paket conda aplikasi.
- Sampel Maya paket mendefinisikan variabel lingkungan `MAYA_NO_HOME=1` untuk mengisolasi dari konfigurasi tingkat pengguna, dan menambahkan jalur pencarian modul `MAYA_MODULE_PATH` sehingga plugin yang dikemas secara terpisah dapat diintegrasikan dari dalam lingkungan virtual. Sampel MtoA paket menempatkan file `mod` di salah satu direktori ini untuk dimuat di Maya startup.

Tulis resep metada

1. Buka GitHub [deadline-cloud-samples/conda_recipes/maya-2025](https://github.com/deadline-cloud-samples/conda_recipes/maya-2025) direktori di browser Anda atau [di editor](#) teks di klon lokal Anda dari repositori.

File tersebut `deadline-cloud.yaml` menjelaskan platform build conda untuk membangun paket dan dari mana mendapatkan aplikasi. Sampel resep menentukan keduanya Linux and Windows membangun, dan itu hanya Linux dikirimkan secara default.

2. Unduh lengkapnya Maya installer dari Anda Autodesk masuk. Untuk Linux, paket build dapat menggunakan arsip secara langsung, jadi letakkan langsung ke `conda_recipes/archive_files` direktori. Untuk Windows, penginstal memerlukan akses administrator untuk dijalankan. Anda harus menjalankan installer dan mengumpulkan file yang diperlukan ke dalam arsip untuk resep paket yang ingin Anda gunakan. File [README.md](#) dalam resep mendokumentasikan prosedur yang dapat diulang untuk membuat artefak ini. Prosedur ini menggunakan EC2 instans Amazon yang baru diluncurkan untuk menyediakan lingkungan yang bersih untuk instalasi yang kemudian dapat Anda hentikan setelah menyimpan hasilnya. Untuk mengemas aplikasi lain yang memerlukan akses administrator, Anda dapat mengikuti prosedur serupa setelah Anda menentukan kumpulan file yang dibutuhkan aplikasi.
3. Buka file [resep/recipe.yaml](#) dan [resep/meta.yaml](#) untuk meninjau atau mengedit pengaturan untuk `rattler-build` dan untuk `conda-build`. Anda dapat mengatur nama paket dan versi untuk aplikasi yang Anda kemas.

Bagian sumber mencakup referensi ke arsip, termasuk hash sha256 dari file. Setiap kali Anda mengubah file-file ini, misalnya ke versi baru, Anda perlu menghitung dan memperbarui nilai-nilai ini.

Bagian build terutama berisi opsi untuk mematikan opsi relokasi biner default, karena mekanisme otomatis tidak akan berfungsi dengan benar untuk pustaka dan direktori biner tertentu yang digunakan paket.

Terakhir, bagian about memungkinkan Anda memasukkan beberapa metadata tentang aplikasi yang dapat digunakan saat menjelajah atau memproses konten saluran conda.

Tulis skrip pembuatan paket

1. Paket membangun skrip di Maya contoh resep conda build termasuk komentar yang menjelaskan langkah-langkah yang dilakukan skrip. Baca komentar dan perintah untuk menemukan yang berikut:

- Bagaimana resep menangani file RPM dari Autodesk
 - Perubahan yang diterapkan resep untuk membuat instalasi dapat dipindahkan ke lingkungan virtual conda tempat resep diinstal
 - Bagaimana resep menetapkan variabel utilitas seperti MAYA_LOCATION dan MAYA_VERSION yang dapat digunakan perangkat lunak Anda untuk memahami Maya itu sedang berjalan.
2. Untuk Linux, buka file [recipe/build.sh](#) untuk meninjau atau mengedit skrip pembuatan paket.

Untuk Windows, buka file [recipe/build_win.sh](#) untuk meninjau atau mengedit skrip pembuatan paket.

Kirimkan pekerjaan yang membangun Maya paket

1. Masukkan conda_recipes direktori di klon GitHub [deadline-cloud-samples](#) repositori Anda.
2. Pastikan bahwa Deadline Cloud farm Anda dikonfigurasi untuk Deadline Cloud CLI Anda. Jika Anda mengikuti langkah-langkah untuk [Membuat saluran conda menggunakan Amazon S3](#) maka peternakan Anda harus dikonfigurasi untuk CLI Anda.
3. Jalankan perintah berikut untuk mengirimkan pekerjaan yang membangun keduanya Linux and Windows paket.

```
./submit-package-job maya-2025 --all-platforms
```

Buat resep conda build untuk Autodesk Maya to Arnold (MtoA) plugin

Anda dapat mengemas plugin untuk aplikasi komersial sebagai paket conda. Plugin adalah pustaka yang dimuat secara dinamis yang menggunakan antarmuka biner aplikasi (ABI) yang disediakan oleh aplikasi untuk memperluas fungsionalitas aplikasi itu. Bagian Maya to Arnold (MtoA) Plugin menambahkan Arnold perender sebagai opsi di dalam Maya.

Membuat paket untuk plugin seperti mengemas aplikasi, tetapi paket terintegrasi dengan aplikasi host yang terkandung dalam paket yang berbeda. Daftar berikut menjelaskan persyaratan untuk membuat ini berfungsi.

- Sertakan paket aplikasi host sebagai dependensi build dan run dalam resep build `meta.yaml` dan `recipe.yaml`. Gunakan batasan versi sehingga resep build hanya diinstal dengan paket yang kompatibel.
- Bagian MtoA resep pembuatan sampel tergantung pada Maya paket dan menggunakan `==` kendala untuk versi.
- Ikuti konvensi paket aplikasi host untuk mendaftarkan plugin.
 - Bagian Maya paket mengkonfigurasi Maya jalur modul di lingkungan virtual, `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules`, untuk plugin untuk menempatkan `.mod` file di. Bagian MtoA sample build recipe membuat file `mtoa.mod` di direktori ini.

Tulis metadata resep

1. Buka GitHub [deadline-cloud-samples/conda_recipes/maya-mtoa-2025](https://github.com/autodesk-research/deadline-cloud-samples/tree/master/conda_recipes/maya-mtoa-2025) direktori di browser Anda atau di editor teks di klon lokal repositori Anda.

Resepnya mengikuti pola yang sama dengan Maya conda build recipe, dan menggunakan arsip sumber yang sama untuk menginstal plugin.

2. Buka file [resep/recipe.yaml](#) dan [resep/meta.yaml](#) untuk meninjau atau mengedit pengaturan untuk [rattler-build](#) dan untuk conda-build. File-file ini menentukan ketergantungan maya selama pembuatan paket dan saat membuat lingkungan virtual untuk menjalankan plugin.

Tulis skrip pembuatan paket

- Paket membangun skrip di MtoA contoh resep conda build termasuk komentar yang menjelaskan langkah-langkah yang dilakukan skrip. Baca komentar dan perintah untuk mempelajari cara menginstal resep MtoA dan membuat file `mtoa.mod` di direktori yang ditentukan oleh Maya paket.

Arnold and Maya menggunakan teknologi lisensi yang sama, jadi Maya resep conda build sudah menyertakan informasi yang dibutuhkan oleh Arnold.

Perbedaan antara Linux and Windows skrip build mirip dengan perbedaan untuk Maya resep conda build.

Kirimkan pekerjaan yang membangun Maya MtoA paket plugin

1. Masukkan `conda_recipes` direktori di klon GitHub [deadline-cloud-samples](https://github.com/autodesk-research/deadline-cloud-samples) repositori Anda.

2. Pastikan bahwa Anda telah membangun paket untuk Maya aplikasi host dari bagian sebelumnya.
3. Pastikan bahwa Deadline Cloud farm Anda dikonfigurasi untuk Deadline Cloud CLI Anda. Jika Anda mengikuti langkah-langkah untuk [Membuat saluran conda menggunakan Amazon S3](#) maka peternakan Anda harus dikonfigurasi untuk CLI Anda.
4. Jalankan perintah berikut untuk mengirimkan pekerjaan yang membangun keduanya Linux and Windows paket.

```
./submit-package-job maya-mtoa-2025 --all-platforms
```

Uji paket Anda dengan Maya pekerjaan render

Setelah Anda memiliki Maya 2025 dan MtoA paket dibangun, Anda dapat mengirimkan pekerjaan untuk dirender dengan paket. [Meja putar dengan Maya/Arnold](#) contoh bundel pekerjaan merender animasi dengan Maya and Arnold. Sampel ini juga digunakan FFmpeg untuk menyandikan video. Anda dapat menambahkan saluran conda-forge ke daftar default CondaChannels di lingkungan antrian conda Anda untuk menyediakan sumber paket. ffmpeg

Dari `job_bundles` direktori di git clone Anda [deadline-cloud-samples](#), jalankan perintah berikut.

```
deadline bundle submit turntable_with_maya_arnold
```

Anda dapat menggunakan monitor Deadline Cloud untuk melacak kemajuan pekerjaan Anda:

1. Di monitor, pilih tugas untuk pekerjaan yang Anda kirimkan, lalu pilih opsi untuk melihat log.
2. Di sisi kanan tampilan log, pilih tindakan sesi Launch Conda.

Anda dapat melihat bahwa tindakan yang dicari maya and maya-mtoa di saluran conda yang dikonfigurasi untuk lingkungan antrian, dan menemukan paket di saluran S3.

Buat lowongan kerja untuk dikirimkan ke Deadline Cloud

Anda mengirimkan pekerjaan ke Deadline Cloud menggunakan paket pekerjaan. Paket pekerjaan adalah kumpulan file, termasuk template [pekerjaan Open Job Description \(OpenJD\)](#) dan file aset apa pun yang diperlukan untuk merender pekerjaan.

Template pekerjaan menjelaskan bagaimana pekerja memproses dan mengakses aset, dan menyediakan skrip yang dijalankan pekerja. Paket Job memungkinkan artis, direktur teknis, dan pengembang pipeline untuk dengan mudah mengirimkan pekerjaan kompleks ke Deadline Cloud dari workstation lokal atau farm render lokal mereka. Ini sangat berguna bagi tim yang bekerja pada efek visual skala besar, animasi, atau proyek rendering media lainnya yang membutuhkan sumber daya komputasi sesuai permintaan yang dapat diskalakan.

Anda dapat membuat bundel pekerjaan menggunakan sistem file lokal untuk menyimpan file dan editor teks untuk membuat template pekerjaan. Setelah membuat bundel, kirimkan pekerjaan ke Deadline Cloud menggunakan Deadline Cloud CLI atau alat seperti pengirim Deadline Cloud

Anda dapat menyimpan aset Anda dalam sistem file yang dibagikan di antara pekerja Anda, atau Anda dapat menggunakan lampiran pekerjaan Deadline Cloud untuk mengotomatiskan pemindahan aset ke bucket S3 tempat pekerja Anda dapat mengaksesnya. Lampiran Job juga membantu memindahkan output dari pekerjaan Anda kembali ke workstation Anda.

Bagian berikut memberikan petunjuk terperinci tentang membuat dan mengirimkan paket pekerjaan ke Deadline Cloud.

Topik

- [Templat Open Job Description \(OpenJD\) untuk Deadline Cloud](#)
- [Menggunakan file dalam pekerjaan Anda](#)
- [Gunakan lampiran pekerjaan untuk berbagi file](#)
- [Buat batas sumber daya untuk pekerjaan](#)
- [Cara mengirimkan pekerjaan ke Deadline Cloud](#)
- [Jadwalkan pekerjaan di Deadline Cloud](#)
- [Ubah pekerjaan di Deadline Cloud](#)

Templat Open Job Description (OpenJD) untuk Deadline Cloud

Paket pekerjaan adalah salah satu alat yang Anda gunakan untuk menentukan pekerjaan untuk AWS Deadline Cloud. Mereka mengelompokkan template [Open Job Description \(OpenJD\)](#) dengan informasi tambahan seperti file dan direktori yang digunakan pekerjaan Anda dengan lampiran pekerjaan. Anda menggunakan antarmuka baris perintah Deadline Cloud (CLI) untuk menggunakan bundel pekerjaan untuk mengirimkan pekerjaan agar antrian dapat dijalankan.

Bundel pekerjaan adalah struktur direktori yang berisi template pekerjaan OpenJD, file lain yang menentukan pekerjaan, dan file khusus pekerjaan yang diperlukan sebagai input untuk pekerjaan Anda. Anda dapat menentukan file yang menentukan pekerjaan Anda sebagai file YAMAL atau JSON.

Satu-satunya file yang diperlukan adalah salah satu `template.yaml` atau `template.json`. Anda juga dapat menyertakan file-file berikut:

```
/template.yaml (or template.json)
/asset_references.yaml (or asset_references.json)
/parameter_values.yaml (or parameter_values.json)
/other job-specific files and directories
```

Gunakan bundel pekerjaan untuk pengiriman pekerjaan khusus dengan Deadline Cloud CLI dan lampiran pekerjaan, atau Anda dapat menggunakan antarmuka pengiriman grafis. Misalnya, berikut ini adalah sampel Blender dari GitHub. Untuk menjalankan sampel menggunakan perintah berikut di [direktori sampel Blender](#):

```
deadline bundle gui-submit blender_render
```

The screenshot shows a macOS-style window titled "Submit to AWS Deadline Cloud". It has three tabs: "Shared job settings" (selected), "Job-specific settings", and "Job attachments".

Job Properties

- Name:
- Description:
- Priority: (with up/down arrows)
- Initial state: (with a dropdown arrow)
- Maximum failed tasks count: (with up/down arrows)
- Maximum retries per task: (with up/down arrows)
- Maximum worker count:
☐ No max worker count
☒ Set max worker count
 (with up/down arrows)

Deadline Cloud settings

- Farm:
- Queue:

Authentication and Status

- Credential source: **HOST_PROVIDED**
- Authentication status: **AUTHENTICATED**
- AWS Deadline Cloud API: **AUTHORIZED**

Buttons

- Login
- Logout
- Settings...
- Export bundle
- Submit

Panel pengaturan khusus pekerjaan dihasilkan dari userInterface properti parameter pekerjaan yang ditentukan dalam templat pekerjaan.

Untuk mengirimkan pekerjaan menggunakan baris perintah, Anda dapat menggunakan perintah yang mirip dengan berikut ini

```
deadline bundle submit \
  --yes \
  --name Demo \
  -p BlenderSceneFile=location of scene file \
  -p OutputDir=file pathe for job output \
  blender_render/
```

Atau Anda dapat menggunakan `deadline.client.api.create_job_from_job_bundle` fungsi dalam paket `deadline` Python.

Semua plugin pengirim pekerjaan yang disediakan dengan Deadline Cloud, seperti plugin Autodesk Maya, menghasilkan bundel pekerjaan untuk kiriman Anda dan kemudian gunakan paket Deadline Cloud Python untuk mengirimkan pekerjaan Anda ke Deadline Cloud. Anda dapat melihat bundel pekerjaan yang dikirimkan di direktori riwayat pekerjaan stasiun kerja Anda atau dengan menggunakan pengirim. Anda dapat menemukan direktori riwayat pekerjaan Anda dengan perintah berikut:

```
deadline config get settings.job_history_dir
```

Ketika pekerjaan Anda berjalan pada pekerja Deadline Cloud, ia memiliki akses ke variabel lingkungan yang memberikan informasi tentang pekerjaan tersebut. Variabel lingkungan adalah:

Nama variabel	Tersedia
DEADLINE_FARM_ID	Semua tindakan
DEADLINE_FLEET_ID	Semua tindakan
DEADLINE_WORKER_ID	Semua tindakan
DEADLINE_QUEUE_ID	Semua tindakan
DEADLINE_JOB_ID	Semua tindakan
DEADLINE_SESSION_ID	Semua tindakan
DEADLINE_SESSIONACTION_ID	Semua tindakan
DEADLINE_TASK_ID	Tindakan tugas

Topik

- [Elemen template Job untuk bundel pekerjaan](#)
- [Nilai parameter elemen untuk bundel pekerjaan](#)
- [Elemen referensi aset untuk bundel pekerjaan](#)

Elemen template Job untuk bundel pekerjaan

Template pekerjaan mendefinisikan lingkungan runtime dan proses yang berjalan sebagai bagian dari pekerjaan Deadline Cloud. Anda dapat membuat parameter dalam template sehingga dapat digunakan untuk membuat pekerjaan yang hanya berbeda dalam nilai input, seperti fungsi dalam bahasa pemrograman.

Saat Anda mengirimkan pekerjaan ke Deadline Cloud, itu berjalan di lingkungan antrian apa pun yang diterapkan ke antrian. Lingkungan antrian dibangun menggunakan spesifikasi lingkungan eksternal Open Job Description (OpenJD). Untuk detailnya, lihat [template Lingkungan di repositori OpenJD GitHub](#).

Untuk pengenalan yang membuat pekerjaan dengan template pekerjaan OpenJD, lihat [Pengantar untuk membuat pekerjaan di repositori GitHub OpenJD](#). Informasi tambahan dapat ditemukan di [Bagaimana pekerjaan dijalankan](#). Ada contoh template pekerjaan di dalam direktori GitHub repositori OpenJD. `samples`

Anda dapat menentukan template pekerjaan dalam format YAMAL (`template.yaml`) atau format JSON (`template.json`). Contoh di bagian ini ditampilkan dalam format YAMAL.

Misalnya, template pekerjaan untuk `blender_render` sampel mendefinisikan parameter input `BlenderSceneFile` sebagai jalur file:

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
      - label: Blender Scene Files
```

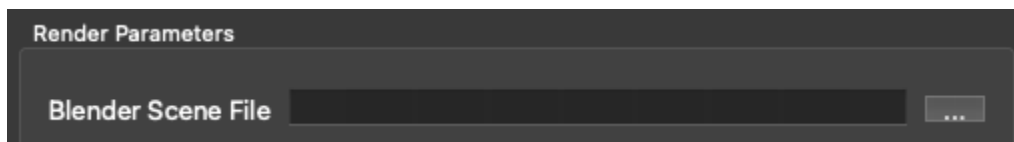
```

    patterns: ["*.blend"]
  - label: All Files
    patterns: ["*"]
  description: >
    Choose the Blender scene file to render. Use the 'Job Attachments' tab
    to add textures and other files that the job needs.

```

`userInterfaceProperti` mendefinisikan perilaku antarmuka pengguna yang dihasilkan secara otomatis untuk kedua baris perintah menggunakan `deadline bundle gui-submit` perintah dan dalam plugin pengiriman pekerjaan untuk aplikasi seperti Autodesk Maya.

Dalam contoh ini, widget UI untuk memasukkan nilai untuk `BlenderSceneFile` parameter adalah dialog pemilihan file yang hanya menampilkan file. `.blend`



Untuk lebih banyak contoh penggunaan `userInterface` elemen, lihat contoh [gui_control_showcase](#) di repositori pada [deadline-cloud-samples](#) GitHub

`dataFlowProperti objectType` dan mengontrol perilaku lampiran pekerjaan saat Anda mengirimkan pekerjaan dari bundel pekerjaan. Dalam hal ini, `objectType: FILE` dan `dataFlow: IN` berarti bahwa nilai `BlenderSceneFile` adalah file input untuk lampiran pekerjaan.

Sebaliknya, definisi `OutputDir` parameter memiliki `objectType: DIRECTORY` dan `dataFlow: OUT`:

```

- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
    label: Output Directory
    groupLabel: Render Parameters
    default: "./output"
    description: Choose the render output directory.

```

Nilai `OutputDir` parameter digunakan oleh lampiran pekerjaan sebagai direktori tempat pekerjaan menulis file output.

Untuk informasi selengkapnya tentang dataFlow properti objectType dan properti, lihat [JobPathParameterDefinitions spesifikasi Open Job Description](#)

Contoh template blender_render pekerjaan lainnya mendefinisikan alur kerja pekerjaan sebagai langkah tunggal dengan setiap frame dalam animasi yang dirender sebagai tugas terpisah:

```
steps:
- name: RenderBlender
  parameterSpace:
    taskParameterDefinitions:
      - name: Frame
        type: INT
        range: "{{Param.Frames}}"
  script:
    actions:
      onRun:
        command: bash
        # Note: {{Task.File.Run}} is a variable that expands to the filename on the
worker host's
        # disk where the contents of the 'Run' embedded file, below, is written.
        args: ['{{Task.File.Run}}']
    embeddedFiles:
      - name: Run
        type: TEXT
        data: |
          # Configure the task to fail if any individual command fails.
          set -xeuo pipefail

          mkdir -p '{{Param.OutputDir}}'

          blender --background '{{Param.BlenderSceneFile}}' \
            --render-output '{{Param.OutputDir}}/{{Param.OutputPattern}}' \
            --render-format {{Param.Format}} \
            --use-extension 1 \
            --render-frame {{Task.Param.Frame}}
```

Misalnya, jika nilai Frames parameternya 1-10, ia mendefinisikan 10 tugas. Masing-masing memiliki tugas memiliki nilai yang berbeda untuk Frame parameter. Untuk menjalankan tugas:

1. Semua referensi variabel dalam data properti file tertanam diperluas, misalnya --render-frame 1.
2. Isi data properti ditulis ke file di direktori kerja sesi pada disk.

3. onRunPerintah tugas menyelesaikan bash *location of embedded file* dan kemudian berjalan.

Untuk informasi selengkapnya tentang file yang disematkan, sesi, dan lokasi yang dipetakan jalur, lihat [Bagaimana pekerjaan dijalankan](#) dalam spesifikasi Open [Job Description](#).

Ada lebih banyak contoh template pekerjaan di repositori [deadline-cloud-samples/job_bundles](#), serta sampel [template yang disediakan](#) dengan spesifikasi Open Job Descriptions.

Nilai parameter elemen untuk bundel pekerjaan

Anda dapat menggunakan file parameter untuk mengatur nilai dari beberapa parameter pekerjaan dalam template pekerjaan atau argumen permintaan [CreateJob](#) operasi dalam bundel pekerjaan sehingga Anda tidak perlu menetapkan nilai saat mengirimkan pekerjaan. UI untuk pengiriman pekerjaan memungkinkan Anda untuk memodifikasi nilai-nilai ini.

Anda dapat menentukan template pekerjaan dalam format YAMAL (`parameter_values.yaml`) atau format JSON (`parameter_values.json`). Contoh di bagian ini ditampilkan dalam format YAMAL.

Di YAMAL, format file adalah:

```
parameterValues:
- name: <string>
  value: <integer>, <float>, or <string>
- name: <string>
  value: <integer>, <float>, or <string>ab
... repeating as necessary
```

Setiap elemen `parameterValues` daftar harus salah satu dari yang berikut:

- Parameter pekerjaan didefinisikan dalam template pekerjaan.
- Parameter pekerjaan yang ditentukan dalam lingkungan antrian untuk antrian yang Anda kirimkan pekerjaan ke..
- Parameter khusus diteruskan ke `CreateJob` operasi saat membuat pekerjaan.
 - `deadline:priority`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter [prioritas](#).
 - `deadline:targetTaskRunStatus`— Nilai harus berupa string. Itu diteruskan ke `CreateJob` operasi sebagai parameter [targetTaskRunStatus](#).

- `deadline:maxFailedTasksCount`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter [maxFailedTasksCount](#).
- `deadline:maxRetriesPerTask`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai parameter [maxRetriesPerTugas](#).
- `deadline:maxWorkercount`— Nilai harus berupa bilangan bulat. Itu diteruskan ke `CreateJob` operasi sebagai [maxWorkerCount](#) parameter.

Template pekerjaan selalu merupakan template daripada pekerjaan tertentu untuk dijalankan. File nilai parameter memungkinkan bundel pekerjaan untuk bertindak sebagai templat jika beberapa parameter tidak memiliki nilai yang ditentukan dalam file ini, atau sebagai pengiriman pekerjaan tertentu jika semua parameter memiliki nilai.

Misalnya, [sampel blender_render](#) tidak memiliki file parameter dan template pekerjaannya mendefinisikan parameter tanpa nilai default. Template ini harus digunakan sebagai template untuk membuat pekerjaan. Setelah Anda membuat pekerjaan menggunakan bundel pekerjaan ini, Deadline Cloud menulis bundel pekerjaan baru ke direktori riwayat pekerjaan.

Misalnya, ketika Anda mengirimkan pekerjaan dengan perintah berikut:

```
deadline bundle gui-submit blender_render/
```

Bundel pekerjaan baru berisi `parameter_values.yaml` file yang berisi parameter yang ditentukan:

```
% cat ~/.deadline/job_history/(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
parameter_values.yaml
parameterValues:
- name: deadline:targetTaskRunStatus
  value: READY
- name: deadline:maxFailedTasksCount
  value: 10
- name: deadline:maxRetriesPerTask
  value: 5
- name: deadline:priority
  value: 75
- name: BlenderSceneFile
  value: /private/tmp/bundle_demo/bmw27_cpu.blend
- name: Frames
  value: 1-10
- name: OutputDir
```

```
value: /private/tmp/bundle_demo/output
- name: OutputPattern
  value: output_####
- name: Format
  value: PNG
- name: CondaPackages
  value: blender
- name: RezPackages
  value: blender
```

Anda dapat membuat pekerjaan yang sama dengan perintah berikut:

```
deadline bundle submit ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-
JobBundle-Demo/
```

Note

Paket pekerjaan yang Anda kirimkan disimpan ke direktori riwayat pekerjaan Anda. Anda dapat menemukan lokasi direktori tersebut dengan perintah berikut:

```
deadline config get settings.job_history_dir
```

Elemen referensi aset untuk bundel pekerjaan

Anda dapat menggunakan [lampiran pekerjaan](#) Deadline Cloud untuk mentransfer file bolak-balik antara workstation dan Deadline Cloud. File referensi aset mencantumkan file input dan direktori, serta direktori keluaran untuk lampiran Anda. Jika Anda tidak mencantumkan semua file dan direktori dalam file ini, Anda dapat memilikinya saat Anda mengirimkan pekerjaan dengan `deadline bundle gui-submit` perintah.

File ini tidak berpengaruh jika Anda tidak menggunakan lampiran pekerjaan.

Anda dapat menentukan template pekerjaan dalam format YAML (`asset_references.yaml`) atau format JSON (`asset_references.json`). Contoh di bagian ini ditampilkan dalam format YAML.

Di YAML, format file adalah:

```
assetReferences:
```

```

inputs:
    # Filenames on the submitting workstation whose file contents are needed as
    # inputs to run the job.
    filenames:
        - list of file paths
    # Directories on the submitting workstation whose contents are needed as inputs
    # to run the job.
    directories:
        - list of directory paths

outputs:
    # Directories on the submitting workstation where the job writes output files
    # if running locally.
    directories:
        - list of directory paths

# Paths referenced by the job, but not necessarily input or output.
# Use this if your job uses the name of a path in some way, but does not explicitly
need
# the contents of that path.
referencedPaths:
    - list of directory paths

```

Saat memilih file input atau output untuk diunggah ke Amazon S3, Deadline Cloud membandingkan jalur file dengan jalur yang tercantum dalam profil penyimpanan Anda. Setiap lokasi sistem file SHARED -type dalam profil penyimpanan mengabstraksi berbagi file jaringan yang dipasang di workstation dan host pekerja Anda. Deadline Cloud hanya mengunggah file yang tidak ada di salah satu pembagian file ini.

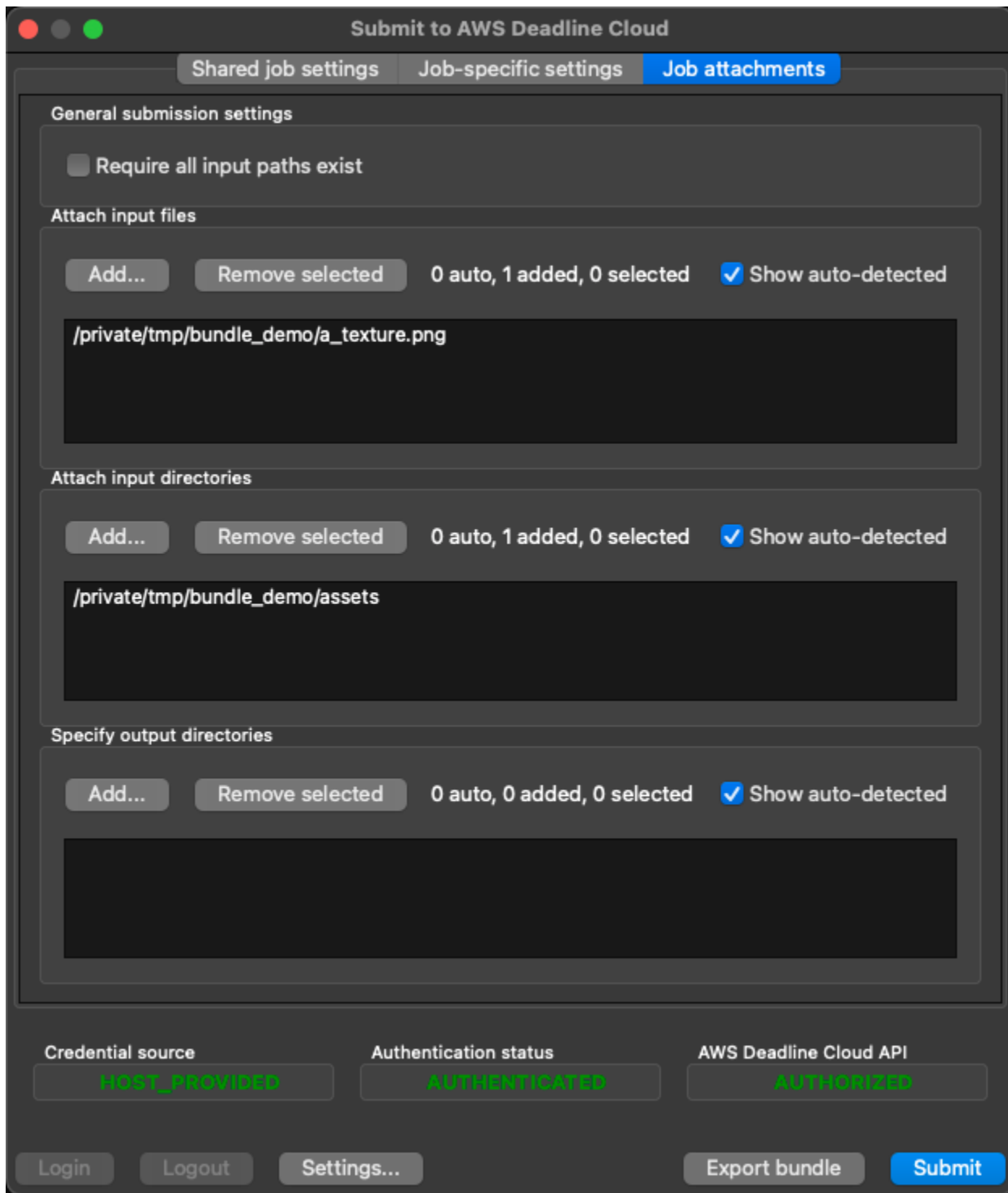
Untuk informasi selengkapnya tentang membuat dan menggunakan profil penyimpanan, lihat [Penyimpanan bersama di Deadline Cloud](#) di Panduan Pengguna Cloud AWS Deadline.

Example - File referensi aset yang dibuat oleh Deadline Cloud GUI

Gunakan perintah berikut untuk mengirimkan pekerjaan menggunakan sampel [blender_render](#).

```
deadline bundle gui-submit blender_render/
```

Tambahkan beberapa file tambahan ke pekerjaan di tab Lampiran Job:



Setelah mengirimkan pekerjaan, Anda dapat melihat `asset_references.yaml` file dalam bundel pekerjaan di direktori riwayat pekerjaan untuk melihat aset dalam file YAMAL:

```
% cat ~/.deadline/job_history/(default\)/2024-06/2024-06-20-01-JobBundle-Demo/asset_references.yaml
```

```
assetReferences:
  inputs:
    filenames:
      - /private/tmp/bundle_demo/a_texture.png
    directories:
      - /private/tmp/bundle_demo/assets
  outputs:
    directories: []
  referencedPaths: []
```

Menggunakan file dalam pekerjaan Anda

Banyak pekerjaan yang Anda kirimkan ke AWS Deadline Cloud memiliki file input dan output. File input dan direktori output Anda mungkin terletak pada kombinasi sistem file bersama dan drive lokal. Pekerjaan perlu menemukan konten di lokasi tersebut. Deadline Cloud menyediakan dua fitur, [lampiran pekerjaan](#) dan [profil penyimpanan](#) yang bekerja sama untuk membantu pekerjaan Anda menemukan file yang mereka butuhkan.

Lampiran Job menawarkan beberapa manfaat

- Memindahkan file antar host menggunakan Amazon S3
- Transfer file dari stasiun kerja Anda ke host pekerja dan sebaliknya
- Tersedia untuk pekerjaan dalam antrian tempat Anda mengaktifkan fitur
- Terutama digunakan dengan armada yang dikelola layanan, tetapi juga kompatibel dengan armada yang dikelola pelanggan.

Gunakan profil penyimpanan untuk memetakan tata letak lokasi sistem file bersama di workstation dan host pekerja Anda. Ini membantu pekerjaan Anda menemukan file dan direktori bersama ketika lokasi mereka berbeda antara workstation dan host pekerja Anda, seperti pengaturan lintas platform dengan workstation berbasis dan host pekerja Windows berbasis. Linux Peta profil penyimpanan konfigurasi sistem file Anda juga digunakan oleh lampiran pekerjaan untuk mengidentifikasi file yang diperlukan untuk berpindah antar host melalui Amazon S3.

Jika Anda tidak menggunakan lampiran pekerjaan, dan Anda tidak perlu memetakan ulang lokasi file dan direktori antara workstation dan host pekerja maka Anda tidak perlu memodelkan fileshares Anda dengan profil penyimpanan.

Topik

- [Contoh infrastruktur proyek](#)
- [Profil penyimpanan dan pemetaan jalur](#)

Contoh infrastruktur proyek

Untuk mendemonstrasikan penggunaan lampiran pekerjaan dan profil penyimpanan, siapkan lingkungan pengujian dengan dua proyek terpisah. Anda dapat menggunakan konsol Deadline Cloud untuk membuat sumber daya pengujian.

1. Jika Anda belum melakukannya, buat peternakan uji. Untuk membuat peternakan, ikuti prosedur di [Buat peternakan](#).
2. Buat dua antrian untuk pekerjaan di masing-masing dari dua proyek. Untuk membuat antrian, ikuti prosedur di [Buat](#) antrian.
 - a. Buat antrian pertama yang disebut **Q1**. Gunakan konfigurasi berikut, gunakan default untuk semua item lainnya.
 - Untuk lampiran pekerjaan, pilih Buat bucket Amazon S3 baru.
 - Pilih Aktifkan asosiasi dengan armada yang dikelola pelanggan.
 - Untuk menjalankan sebagai pengguna, masukkan **jobuser** untuk pengguna dan grup POSIX.
 - Untuk peran layanan antrian, buat peran baru bernama **AssetDemoFarm-Q1-Role**
 - Kosongkan kotak centang lingkungan antrian Conda default.
 - b. Buat antrian kedua yang disebut **Q2**. Gunakan konfigurasi berikut, gunakan default untuk semua item lainnya.
 - Untuk lampiran pekerjaan, pilih Buat bucket Amazon S3 baru.
 - Pilih Aktifkan asosiasi dengan armada yang dikelola pelanggan.
 - Untuk menjalankan sebagai pengguna, masukkan **jobuser** untuk pengguna dan grup POSIX.
 - Untuk peran layanan antrian, buat peran baru bernama **AssetDemoFarm-Q2-Role**
 - Kosongkan kotak centang lingkungan antrian Conda default.
3. Buat satu armada yang dikelola pelanggan yang menjalankan pekerjaan dari kedua antrian. Untuk membuat armada, ikuti prosedur di [Buat armada yang dikelola pelanggan](#). Gunakan konfigurasi berikut:

- Untuk Nama, gunakan **DemoFleet**.
- Untuk jenis Armada pilih Customer managed
- Untuk peran layanan Armada, buat peran baru bernama AssetDemoFarm-Fleet-Role.
- Jangan mengaitkan armada dengan antrian apa pun.

Lingkungan pengujian mengasumsikan bahwa ada tiga sistem file yang dibagi antara host menggunakan berbagi file jaringan. Dalam contoh ini, lokasi memiliki nama-nama berikut:

- FSCommon- berisi aset pekerjaan masukan yang umum untuk kedua proyek.
- FS1- berisi input dan output aset pekerjaan untuk proyek 1.
- FS2- berisi input dan output aset pekerjaan untuk proyek 2.

Lingkungan pengujian juga mengasumsikan bahwa ada tiga workstation, sebagai berikut:

- WSA11- Workstation Linux berbasis yang digunakan oleh pengembang untuk semua proyek. Lokasi sistem file bersama adalah:
 - FSCommon: /shared/common
 - FS1: /shared/projects/project1
 - FS2: /shared/projects/project2
- WS1- Sebuah workstation Windows berbasis yang digunakan untuk proyek 1. Lokasi sistem file bersama adalah:
 - FSCommon: S:\
 - FS1: Z:\
 - FS2: Tidak tersedia
- WS1- Workstation macOS berbasis yang digunakan untuk proyek 2. Lokasi sistem file bersama adalah:
 - FSCommon: /Volumes/common
 - FS1: Tidak tersedia
 - FS2: /Volumes/projects/project2

Terakhir, tentukan lokasi sistem file bersama untuk pekerja di armada Anda. Contoh-contoh berikut mengacu pada konfigurasi ini sebagai `WorkerConfig`. Lokasi bersama adalah:

- FSCommon: /mnt/common
- FS1: /mnt/projects/project1
- FS2: /mnt/projects/project2

Anda tidak perlu menyiapkan sistem file bersama, workstation, atau pekerja yang cocok dengan konfigurasi ini. Lokasi bersama tidak perlu ada untuk demonstrasi.

Profil penyimpanan dan pemetaan jalur

Gunakan profil penyimpanan untuk memodelkan sistem file di workstation dan host pekerja Anda. Setiap profil penyimpanan menjelaskan sistem operasi dan tata letak sistem file dari salah satu konfigurasi sistem Anda. Topik ini menjelaskan cara menggunakan profil penyimpanan untuk memodelkan konfigurasi sistem file host Anda sehingga Deadline Cloud dapat menghasilkan aturan pemetaan jalur untuk pekerjaan Anda, dan bagaimana aturan pemetaan jalur tersebut dihasilkan dari profil penyimpanan Anda.

Saat mengirimkan pekerjaan ke Deadline Cloud, Anda dapat memberikan ID profil penyimpanan opsional untuk pekerjaan tersebut. Profil penyimpanan ini menjelaskan sistem file workstation pengiriman. Ini menjelaskan konfigurasi sistem file asli yang digunakan jalur file dalam template pekerjaan.

Anda juga dapat mengaitkan profil penyimpanan dengan armada. Profil penyimpanan menjelaskan konfigurasi sistem file dari semua host pekerja di armada. Jika Anda memiliki pekerja dengan konfigurasi sistem file yang berbeda, pekerja tersebut harus ditugaskan ke armada yang berbeda di peternakan Anda.

Aturan pemetaan jalur menjelaskan bagaimana jalur harus dipetakan ulang dari cara mereka ditentukan dalam pekerjaan ke lokasi aktual jalur pada host pekerja. Deadline Cloud membandingkan konfigurasi sistem file yang dijelaskan dalam profil penyimpanan pekerjaan dengan profil penyimpanan armada yang menjalankan pekerjaan untuk mendapatkan aturan pemetaan jalur ini.

Topik

- [Model lokasi sistem file bersama dengan profil penyimpanan](#)
- [Konfigurasi profil penyimpanan untuk armada](#)
- [Konfigurasi profil penyimpanan untuk antrian](#)
- [Turunkan aturan pemetaan jalur dari profil penyimpanan](#)

Model lokasi sistem file bersama dengan profil penyimpanan

Profil penyimpanan memodelkan konfigurasi sistem file dari salah satu konfigurasi host Anda. Ada empat konfigurasi host yang berbeda dalam [infrastruktur proyek sampel](#). Dalam contoh ini Anda membuat profil penyimpanan terpisah untuk masing-masing. Anda dapat membuat profil penyimpanan menggunakan salah satu dari berikut ini:

- [CreateStorageProfile API](#)
- [AWS::Deadline::StorageProfile](#) AWS CloudFormation sumber daya
- [AWS konsol](#)

Profil penyimpanan terdiri dari daftar lokasi sistem file yang masing-masing memberi tahu Deadline Cloud lokasi dan jenis lokasi sistem file yang relevan untuk pekerjaan yang dikirim dari atau dijalankan pada host. Profil penyimpanan seharusnya hanya memodelkan lokasi yang relevan untuk pekerjaan. Misalnya, FSCommon lokasi bersama terletak di workstation WS1 di `S:\`, jadi lokasi sistem file yang sesuai adalah:

```
{
  "name": "FSCommon",
  "path": "S:\\",
  "type": "SHARED"
}
```

Gunakan perintah berikut untuk membuat profil penyimpanan untuk konfigurasi workstation WS1WS2, WS3 dan konfigurasi pekerja WorkerConfig menggunakan in [AWS CLI](#): [AWS CloudShell](#)

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WSAll \
  --os-family LINUX \
  --file-system-locations \
  '[
    {"name": "FSCommon", "type":"SHARED", "path":"/shared/common"},
    {"name": "FS1", "type":"SHARED", "path":"/shared/projects/project1"},
    {"name": "FS2", "type":"SHARED", "path":"/shared/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
```

```
--display-name WS1 \
--os-family WINDOWS \
--file-system-locations \
'[
  {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
  {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
]'

aws deadline create-storage-profile --farm-id $FARM_ID \
--display-name WS2 \
--os-family MACOS \
--file-system-locations \
'[
  {"name": "FSCommon", "type":"SHARED", "path":"/Volumes/common"},
  {"name": "FS2", "type":"SHARED", "path":"/Volumes/projects/project2"}
]'

aws deadline create-storage-profile --farm-id $FARM_ID \
--display-name WorkerCfg \
--os-family LINUX \
--file-system-locations \
'[
  {"name": "FSCommon", "type":"SHARED", "path":"/mnt/common"},
  {"name": "FS1", "type":"SHARED", "path":"/mnt/projects/project1"},
  {"name": "FS2", "type":"SHARED", "path":"/mnt/projects/project2"}
]'
```

Note

Anda harus merujuk ke lokasi sistem file di profil penyimpanan Anda menggunakan nilai yang sama untuk name properti di semua profil penyimpanan di pertanian Anda. Deadline Cloud membandingkan nama untuk menentukan bahwa lokasi sistem file dari profil penyimpanan yang berbeda merujuk ke lokasi yang sama saat membuat aturan pemetaan jalur.

Konfigurasi profil penyimpanan untuk armada

Anda dapat mengonfigurasi armada untuk menyertakan profil penyimpanan yang memodelkan lokasi sistem file pada semua pekerja di armada. Konfigurasi sistem file host dari semua pekerja dalam armada harus sesuai dengan profil penyimpanan armada mereka. Pekerja dengan konfigurasi sistem file yang berbeda harus berada dalam armada terpisah.

Untuk mengatur konfigurasi armada Anda agar menggunakan profil WorkerConfig penyimpanan, gunakan fitur [AWS CLI](#) dan [AWS CloudShell](#):

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerConfig
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

FLEET_WORKER_MODE=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
    --query '.configuration.customerManaged.mode' \
)
FLEET_WORKER_CAPABILITIES=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
    --query '.configuration.customerManaged.workerCapabilities' \
)

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
  "{
    \"customerManaged\": {
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

Konfigurasi profil penyimpanan untuk antrian

Konfigurasi antrian mencakup daftar nama peka huruf besar/kecil dari lokasi sistem file bersama yang pekerjaan yang dikirimkan ke antrian memerlukan akses ke. misalnya, pekerjaan yang dikirimkan ke antrian Q1 memerlukan lokasi sistem file dan. FSCommon FS1 Pekerjaan yang dikirimkan ke antrian Q2 memerlukan lokasi sistem file FSCommon danFS2.

Untuk mengatur konfigurasi antrian agar memerlukan lokasi sistem file ini, gunakan skrip berikut:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
```

```

QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of QUEUE2_ID to queue Q2's identifier
QUEUE2_ID=queue-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --required-file-system-location-names-to-add FSComm FS1

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --required-file-system-location-names-to-add FSComm FS2

```

Konfigurasi antrian juga mencakup daftar profil penyimpanan yang diizinkan yang berlaku untuk pekerjaan yang dikirimkan dan armada yang terkait dengan antrian tersebut. Hanya profil penyimpanan yang menentukan lokasi sistem file untuk semua lokasi sistem file yang diperlukan untuk antrian yang diizinkan dalam daftar antrian profil penyimpanan yang diizinkan.

Pekerjaan gagal jika Anda mengirimkannya dengan profil penyimpanan yang tidak ada dalam daftar profil penyimpanan yang diizinkan untuk antrian. Anda selalu dapat mengirimkan pekerjaan tanpa profil penyimpanan ke antrian. Konfigurasi workstation berlabel WSA11 dan WS1 keduanya memiliki lokasi sistem file yang diperlukan (FSCommon dan FS1) untuk antrian. Q1 Mereka harus diizinkan untuk mengirimkan pekerjaan ke antrian. Demikian pula, konfigurasi workstation WSA11 dan WS2 memenuhi persyaratan untuk antrian. Q2 Mereka harus diizinkan untuk mengirimkan pekerjaan ke antrian itu. Perbarui kedua konfigurasi antrian untuk memungkinkan pekerjaan dikirimkan dengan profil penyimpanan ini menggunakan skrip berikut:

```

# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS1 to the identifier of the WS1 storage profile
WS1_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS2 to the identifier of the WS2 storage profile
WS2_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS1_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS2_ID

```

Jika Anda menambahkan profil WS2 penyimpanan ke daftar profil penyimpanan yang diizinkan untuk antrian, Q1 itu gagal:

```
$ aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
```

```
--allowed-storage-profile-ids-to-add $WS2_ID
```

An error occurred (ValidationException) when calling the UpdateQueue operation: Storage profile id: sp-*00112233445566778899aabbccddeeff* does not have required file system location: FS1

Ini karena profil WS2 penyimpanan tidak berisi definisi untuk lokasi sistem file bernama antrian FS1 yang Q1 diperlukan.

Mengaitkan armada yang dikonfigurasi dengan profil penyimpanan yang tidak ada dalam daftar antrian profil penyimpanan yang diizinkan juga gagal. Misalnya:

```
$ aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID
```

An error occurred (ValidationException) when calling the CreateQueueFleetAssociation operation: Mismatch between storage profile ids.

Untuk memperbaiki kesalahan, tambahkan profil penyimpanan yang diberi nama WorkerConfig ke daftar profil penyimpanan yang diizinkan untuk antrian Q1 dan antrian. Q2 Kemudian, kaitkan armada dengan antrian ini sehingga pekerja di armada dapat menjalankan pekerjaan dari kedua antrian.

```
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerCfg
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

Turunkan aturan pemetaan jalur dari profil penyimpanan

Aturan pemetaan jalur menjelaskan bagaimana jalur harus dipetakan ulang dari pekerjaan ke lokasi sebenarnya jalur pada host pekerja. Saat tugas dijalankan pada pekerja, profil penyimpanan dari pekerjaan tersebut dibandingkan dengan profil penyimpanan armada pekerja untuk mendapatkan aturan pemetaan jalur untuk tugas tersebut.

Deadline Cloud membuat aturan pemetaan untuk setiap lokasi sistem file yang diperlukan dalam konfigurasi antrian. Misalnya, pekerjaan yang dikirimkan dengan profil WSAll penyimpanan ke antrian Q1 memiliki aturan pemetaan jalur:

- FSComm: /shared/common -> /mnt/common
- FS1: /shared/projects/project1 -> /mnt/projects/project1

Deadline Cloud membuat aturan untuk lokasi FSComm dan sistem FS1 file, tetapi bukan lokasi sistem FS2 file meskipun profil WSAll dan WorkerConfig penyimpanan ditentukan FS2. Ini karena daftar antrian Q1 lokasi sistem file yang diperlukan adalah ["FSComm", "FS1"].

Anda dapat mengonfirmasi aturan pemetaan jalur yang tersedia untuk pekerjaan yang dikirimkan dengan profil penyimpanan tertentu dengan mengirimkan pekerjaan yang mencetak [file aturan pemetaan jalur Open Job Description](#), lalu membaca log sesi setelah pekerjaan selesai:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSALL storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
```

```

        "onRun": {
            "command": "/bin/cat",
            "args": [ "{{Session.PathMappingRulesFile}}" ]
        }
    }
}
]
}'

```

Jika Anda menggunakan [Deadline Cloud CLI](#) untuk mengirimkan pekerjaan, pengaturan `settings.storage_profile_id` konfigurasinya menetapkan profil penyimpanan yang akan dimiliki oleh pekerjaan yang dikirimkan dengan CLI. Untuk mengirimkan pekerjaan dengan profil WSA11 penyimpanan, atur:

```
deadline config set settings.storage_profile_id $WSALL_ID
```

Untuk menjalankan pekerja yang dikelola pelanggan seolah-olah sedang berjalan di infrastruktur sampel, ikuti prosedur di [Jalankan agen pekerja di](#) Panduan Pengguna Cloud Tenggat Waktu untuk menjalankan pekerja. AWS CloudShell Jika Anda mengikuti instruksi tersebut sebelumnya, hapus `~/demoenv-persist` direktori `~/demoenv-logs` dan direktori terlebih dahulu. Juga, tetapkan nilai-nilai variabel `DEV_FARM_ID` dan `DEV_CMF_ID` lingkungan yang referensi arah sebagai berikut sebelum melakukannya:

```
DEV_FARM_ID=$FARM_ID
DEV_CMF_ID=$FLEET_ID
```

Setelah pekerjaan berjalan, Anda dapat melihat aturan pemetaan jalur di file log pekerjaan:

```
cat demoenv-logs/${QUEUE1_ID}/*.log
...
JSON log results (see below)
...
```

Log berisi pemetaan untuk sistem FSComm file FS1 dan file. Diformat ulang agar mudah dibaca, entri log terlihat seperti ini:

```
{
  "version": "pathmapping-1.0",
  "path_mapping_rules": [
```

```
{
  "source_path_format": "POSIX",
  "source_path": "/shared/projects/project1",
  "destination_path": "/mnt/projects/project1"
},
{
  "source_path_format": "POSIX",
  "source_path": "/shared/common",
  "destination_path": "/mnt/common"
}
]
```

Anda dapat mengirimkan pekerjaan dengan profil penyimpanan yang berbeda untuk melihat bagaimana aturan pemetaan jalur berubah.

Gunakan lampiran pekerjaan untuk berbagi file

Gunakan lampiran pekerjaan untuk membuat file yang tidak ada di direktori bersama tersedia untuk pekerjaan Anda, dan untuk menangkap file output jika tidak ditulis ke direktori bersama. Lampiran Job menggunakan Amazon S3 untuk mengirim file antar host. File disimpan dalam bucket S3, dan Anda tidak perlu mengunggah file jika kontennya tidak berubah.

Anda harus menggunakan lampiran pekerjaan saat menjalankan pekerjaan pada [armada yang dikelola layanan](#) karena host tidak berbagi lokasi sistem file. Lampiran Job juga berguna dengan [armada yang dikelola pelanggan](#) ketika file input atau output pekerjaan disimpan pada sistem file jaringan bersama, seperti ketika [bundel pekerjaan](#) Anda berisi skrip shell atau Python.

Saat Anda mengirimkan paket pekerjaan dengan [Deadline Cloud CLI](#) atau pengirim Deadline Cloud, lampiran pekerjaan menggunakan profil penyimpanan pekerjaan dan lokasi sistem file yang diperlukan antrian untuk mengidentifikasi file input yang tidak ada di host pekerja dan harus diunggah ke Amazon S3 sebagai bagian dari pengiriman pekerjaan. Profil penyimpanan ini juga membantu Deadline Cloud mengidentifikasi file keluaran di lokasi host pekerja yang harus diunggah ke Amazon S3 sehingga tersedia untuk stasiun kerja Anda.

Contoh lampiran pekerjaan menggunakan konfigurasi profil pertanian, armada, antrian, dan penyimpanan dari dan. [Contoh infrastruktur proyek Profil penyimpanan dan pemetaan jalur](#) Anda harus melalui bagian-bagian itu sebelum yang satu ini.

Dalam contoh berikut, Anda menggunakan paket pekerjaan sampel sebagai titik awal, lalu memodifikasinya untuk mengeksplorasi fungsionalitas lampiran pekerjaan. Paket Job adalah cara

terbaik bagi pekerjaan Anda untuk menggunakan lampiran pekerjaan. Mereka menggabungkan template [pekerjaan Open Job Description](#) dalam direktori dengan file tambahan yang mencantumkan file dan direktori yang dibutuhkan oleh pekerjaan menggunakan bundel pekerjaan. Untuk informasi selengkapnya tentang paket pekerjaan, lihat [Templat Open Job Description \(OpenJD\) untuk Deadline Cloud](#).

Mengirimkan file dengan pekerjaan

Dengan Deadline Cloud, Anda dapat mengaktifkan alur kerja pekerjaan untuk mengakses file input yang tidak tersedia di lokasi sistem file bersama di host pekerja. Lampiran Job memungkinkan rendering pekerjaan untuk mengakses file yang hanya berada di drive workstation lokal atau lingkungan armada yang dikelola layanan. Saat mengirimkan bundel pekerjaan, Anda dapat menyertakan daftar file input dan direktori yang diperlukan oleh pekerjaan. Deadline Cloud mengidentifikasi file yang tidak dibagikan ini, mengunggahnya dari mesin lokal ke Amazon S3, dan mengunduhnya ke host pekerja. Ini merampingkan proses mentransfer aset input ke render node, memastikan semua file yang diperlukan dapat diakses untuk eksekusi pekerjaan terdistribusi.

Anda dapat menentukan file untuk pekerjaan secara langsung di bundel pekerjaan, menggunakan parameter dalam template pekerjaan yang Anda berikan menggunakan variabel lingkungan atau skrip, dan menggunakan `assets_references` file pekerjaan. Anda dapat menggunakan salah satu metode ini atau kombinasi ketiganya. Anda dapat menentukan profil penyimpanan untuk bundel untuk pekerjaan sehingga hanya mengunggah file yang telah berubah di workstation lokal.

Bagian ini menggunakan contoh bundel pekerjaan GitHub untuk menunjukkan bagaimana Deadline Cloud mengidentifikasi file dalam pekerjaan Anda untuk diunggah, bagaimana file tersebut diatur di Amazon S3, dan bagaimana file tersebut tersedia untuk host pekerja yang memproses pekerjaan Anda.

Topik

- [Bagaimana Deadline Cloud mengunggah file ke Amazon S3](#)
- [Bagaimana Deadline Cloud memilih file yang akan diunggah](#)
- [Bagaimana pekerjaan menemukan file input lampiran pekerjaan](#)

Bagaimana Deadline Cloud mengunggah file ke Amazon S3

Contoh ini menunjukkan bagaimana Deadline Cloud mengunggah file dari workstation atau host pekerja Anda ke Amazon S3 sehingga file tersebut dapat dibagikan. Ini menggunakan bundel pekerjaan sampel dari GitHub dan Deadline Cloud CLI untuk mengirimkan pekerjaan.

Mulailah dengan mengkloning [GitHubrepositori sampel Deadline Cloud](#) ke [AWS CloudShell](#) lingkungan Anda, lalu salin bundel `job_attachments_devguide` pekerjaan ke direktori home Anda:

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide ~/
```

Instal [Deadline Cloud CLI](#) untuk mengirimkan bundel pekerjaan:

```
pip install deadline --upgrade
```

Bundel `job_attachments_devguide` pekerjaan memiliki satu langkah dengan tugas yang menjalankan skrip bash shell yang lokasi sistem filenya diteruskan sebagai parameter pekerjaan. Definisi parameter pekerjaan adalah:

```
...
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
...
```

INNilai `dataFlow` properti memberi tahu lampiran pekerjaan bahwa nilai `ScriptFile` parameter adalah masukan ke pekerjaan. Nilai `default` properti adalah lokasi relatif ke direktori bundel pekerjaan, tetapi juga bisa menjadi jalur absolut. Definisi parameter ini mendeklarasikan `script.sh` file dalam direktori bundel pekerjaan sebagai file input yang diperlukan agar pekerjaan dapat dijalankan.

Selanjutnya, pastikan bahwa Deadline Cloud CLI tidak memiliki profil penyimpanan yang dikonfigurasi kemudian kirimkan pekerjaan ke antrian: Q1

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''
```

```
deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
  job_attachments_devguide/
```

Output dari Deadline Cloud CLI setelah perintah ini dijalankan terlihat seperti:

```
Submitting to Queue: Q1
...
Hashing Attachments [#####] 100%
Hashing Summary:
  Processed 1 file totaling 39.0 B.
  Skipped re-processing 0 files totaling 0.0 B.
  Total processing time of 0.0327 seconds at 1.19 KB/s.

Uploading Attachments [#####] 100%
Upload Summary:
  Processed 1 file totaling 39.0 B.
  Skipped re-processing 0 files totaling 0.0 B.
  Total processing time of 0.25639 seconds at 152.0 B/s.

Waiting for Job to be created...
Submitted job bundle:
  job_attachments_devguide/
Job creation completed successfully
job-74148c13342e4514b63c7a7518657005
```

Saat Anda mengirimkan pekerjaan, Deadline Cloud pertama-tama melakukan hash `script.sh` file dan kemudian mengunggahnya ke Amazon S3.

Deadline Cloud memperlakukan bucket S3 sebagai penyimpanan yang dapat dialamatkan konten. File diunggah ke objek S3. Nama objek berasal dari hash dari isi file. Jika dua file memiliki konten yang identik, mereka memiliki nilai hash yang sama terlepas dari di mana file tersebut berada atau apa namanya. Ini memungkinkan Deadline Cloud untuk menghindari mengunggah file jika sudah tersedia.

Anda dapat menggunakan [AWS CLI](#) untuk melihat objek yang diunggah ke Amazon S3:

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)
```

```
aws s3 ls s3://$Q1_S3_BUCKET --recursive
```

Dua objek diunggah ke S3:

- DeadlineCloud/Data/87cb19095dd5d78fc56384ef0e6241.xxh128— Isi dariscript.sh. [Nilai 87cb19095dd5d78fc56384ef0e6241 dalam kunci objek adalah hash dari isi file, dan ekstensi xxh128 menunjukkan bahwa nilai hash dihitung sebagai 128 bit xxhash.](#)
- DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input— Objek manifes untuk pengajuan pekerjaan. Nilai<farm-id>,<queue-id>, dan <guid> merupakan pengidentifikasi pertanian Anda, pengidentifikasi antrian, dan nilai heksidesimal acak. Nilai a1d221c7fd97b08175b3872a37428e8c dalam contoh ini adalah nilai hash yang dihitung dari string/home/cloudshell-user/job_attachments_devguide, direktori tempat script.sh berada.

Objek manifes berisi informasi untuk file input pada jalur root tertentu yang diunggah ke S3 sebagai bagian dari pengiriman pekerjaan. Unduh file manifes ini (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`). Isinya mirip dengan:

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "87cb19095dd5d78fc56384ef0e6241",
      "mtime": 1721147454416085,
      "path": "script.sh",
      "size": 39
    }
  ],
  "totalSize": 39
}
```

Ini menunjukkan bahwa file `script.sh` telah diunggah, dan hash dari konten file itu. 87cb19095dd5d78fc56384ef0e6241 Nilai hash ini cocok dengan nilai dalam nama DeadlineCloud/Data/87cb19095dd5d78fc56384ef0e6241.xxh128 objek. Ini digunakan oleh Deadline Cloud untuk mengetahui objek mana yang akan diunduh untuk konten file ini.

Skema lengkap untuk file ini [tersedia di GitHub](#).

Bila Anda menggunakan [CreateJob operasi](#), Anda dapat mengatur lokasi objek manifes. Anda dapat menggunakan [GetJoboperasi](#) untuk melihat lokasi:

```
{
  "attachments": {
    "file system": "COPIED",
    "manifests": [
      {
        "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
        "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input",
        "rootPath": "/home/cloudshell-user/job_attachments_devguide",
        "rootPathFormat": "posix"
      }
    ]
  },
  ...
}
```

Bagaimana Deadline Cloud memilih file yang akan diunggah

File dan direktori yang dipertimbangkan lampiran pekerjaan untuk diunggah ke Amazon S3 sebagai input ke pekerjaan Anda adalah:

- Nilai-nilai dari semua parameter pekerjaan PATH -type didefinisikan dalam template pekerjaan bundel pekerjaan dengan dataFlow nilai IN atau INOUT.
- File dan direktori terdaftar sebagai input dalam file referensi aset bundel pekerjaan.

Jika Anda mengirimkan pekerjaan tanpa profil penyimpanan, semua file yang dipertimbangkan untuk diunggah akan diunggah. Jika Anda mengirimkan pekerjaan dengan profil penyimpanan, file tidak akan diunggah ke Amazon S3 jika mereka berada di lokasi sistem file tipe penyimpanan profil SHARED penyimpanan yang juga diperlukan lokasi sistem file untuk antrian. Lokasi ini diharapkan tersedia di host pekerja yang menjalankan pekerjaan, jadi tidak perlu mengunggahnya ke S3.

Dalam contoh ini, Anda membuat lokasi sistem SHARED file WSAll di CloudShell lingkungan AWS Anda dan kemudian menambahkan file ke lokasi sistem file tersebut. Gunakan perintah berikut ini.

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
```

```
sudo mkdir -p /shared/common /shared/projects/project1 /shared/projects/project2
sudo chown -R cloudshell-user:cloudshell-user /shared

for d in /shared/common /shared/projects/project1 /shared/projects/project2; do
    echo "File contents for $d" > ${d}/file.txt
done
```

Selanjutnya, tambahkan file referensi aset ke bundel pekerjaan yang menyertakan semua file yang Anda buat sebagai input untuk pekerjaan tersebut. Gunakan perintah berikut ini.

```
cat > ${HOME}/job_attachments_devguide/asset_references.yaml << EOF
assetReferences:
  inputs:
    filenames:
      - /shared/common/file.txt
    directories:
      - /shared/projects/project1
      - /shared/projects/project2
EOF
```

Selanjutnya, konfigurasi Deadline Cloud CLI untuk mengirimkan pekerjaan dengan WSAll profil penyimpanan, lalu kirimkan bundel pekerjaan:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
job_attachments_devguide/
```

Deadline Cloud mengunggah dua file ke Amazon S3 saat Anda mengirimkan pekerjaan. Anda dapat mengunduh objek manifes untuk pekerjaan dari S3 untuk melihat file yang diunggah:

```
for manifest in $( \
    aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID \
    --query 'attachments.manifests[].inputManifestPath' \
```

```

    | jq -r '.[[]]'
  ); do
    echo "Manifest object: $manifest"
    aws s3 cp --quiet s3://$Q1_S3_BUCKET/DeadlineCloud/Manifests/$manifest /dev/stdout |
    jq .
  done

```

Dalam contoh ini, ada satu file manifest dengan konten berikut:

```

{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "87cb19095dd5d78fc56384ef0e6241",
      "mtime": 1721147454416085,
      "path": "home/cloudshell-user/job_attachments_devguide/script.sh",
      "size": 39
    },
    {
      "hash": "af5a605a3a4e86ce7be7ac5237b51b79",
      "mtime": 1721163773582362,
      "path": "shared/projects/project2/file.txt",
      "size": 44
    }
  ],
  "totalSize": 83
}

```

Gunakan [GetJob operasi](#) untuk manifest untuk melihat bahwa rootPath adalah “/”.

```

aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID --query
'attachments.manifests[*]'

```

Jalur root untuk kumpulan file input selalu merupakan subpath umum terpanjang dari file-file tersebut. Jika pekerjaan Anda dikirim dari Windows sebagai gantinya dan ada file input tanpa subpath umum karena mereka berada di drive yang berbeda, Anda melihat jalur root terpisah pada setiap drive. Jalur dalam manifest selalu relatif terhadap jalur root manifest, sehingga file input yang diunggah adalah:

- /home/cloudshell-user/job_attachments_devguide/script.sh— File skrip dalam bundel pekerjaan.

- `/shared/projects/project2/file.txt`— File di lokasi sistem SHARED file di profil WSA11 penyimpanan yang tidak ada dalam daftar lokasi sistem file yang diperlukan untuk antrianQ1.

File di lokasi sistem file FSCommon (`/shared/common/file.txt`) dan FS1 (`/shared/projects/project1/file.txt`) tidak ada dalam daftar. Ini karena lokasi sistem file tersebut berada SHARED di profil WSA11 penyimpanan dan keduanya berada dalam daftar lokasi sistem file yang diperlukan dalam antrianQ1.

Anda dapat melihat lokasi sistem file yang dipertimbangkan SHARED untuk pekerjaan yang dikirimkan dengan profil penyimpanan tertentu dengan [GetStorageProfileForQueue operasi](#). Untuk kueri profil penyimpanan WSA11 untuk antrian Q1 gunakan perintah berikut:

```
aws deadline get-storage-profile --farm-id $FARM_ID --storage-profile-id $WSALL_ID

aws deadline get-storage-profile-for-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID --
storage-profile-id $WSALL_ID
```

Bagaimana pekerjaan menemukan file input lampiran pekerjaan

Agar pekerjaan dapat menggunakan file yang diunggah Deadline Cloud ke Amazon S3 menggunakan lampiran pekerjaan, pekerjaan Anda memerlukan file-file tersebut yang tersedia melalui sistem file di host pekerja. Saat [sesi](#) untuk pekerjaan Anda berjalan di host pekerja, Deadline Cloud mengunduh file input untuk pekerjaan tersebut ke direktori sementara di drive lokal host pekerja dan menambahkan aturan pemetaan jalur untuk setiap jalur root pekerjaan ke lokasi sistem filenya di drive lokal.

Untuk contoh ini, mulai agen pekerja Deadline Cloud di CloudShell tab AWS. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan, lalu hapus log pekerjaan dari direktori log:

```
rm -rf ~/devdemo-logs/queue-*
```

Skip berikut memodifikasi bundel pekerjaan untuk menampilkan semua file di direktori kerja sementara sesi dan isi file aturan pemetaan jalur, dan kemudian mengirimkan pekerjaan dengan bundel yang dimodifikasi:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
```

```
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

cat > ~/job_attachments_devguide/script.sh << EOF
#!/bin/bash

echo "Session working directory is: \$(pwd)"
echo
echo "Contents:"
find . -type f
echo
echo "Path mapping rules file: \$1"
jq . \$1
EOF

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/bash
        args:
          - "{{Param.ScriptFile}}"
          - "{{Session.PathMappingRulesFile}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
job_attachments_devguide/
```

Anda dapat melihat log pekerjaan yang dijalankan setelah dijalankan oleh pekerja di AWS CloudShell lingkungan Anda:

```
cat demoenv-logs/queue-*/session*.log
```

Log menunjukkan bahwa hal pertama yang terjadi dalam sesi adalah dua file input untuk pekerjaan yang diunduh ke pekerja:

```
2024-07-17 01:26:37,824 INFO =====
2024-07-17 01:26:37,825 INFO ----- Job Attachments Download for Job
2024-07-17 01:26:37,825 INFO =====
2024-07-17 01:26:37,825 INFO Syncing inputs using Job Attachments
2024-07-17 01:26:38,116 INFO Downloaded 142.0 B / 186.0 B of 2 files (Transfer rate:
0.0 B/s)
2024-07-17 01:26:38,174 INFO Downloaded 186.0 B / 186.0 B of 2 files (Transfer rate:
733.0 B/s)
2024-07-17 01:26:38,176 INFO Summary Statistics for file downloads:
Processed 2 files totaling 186.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.09752 seconds at 1.91 KB/s.
```

Berikutnya adalah output dari `script.sh` run by the job:

- File masukan yang diunggah saat pekerjaan dikirimkan terletak di bawah direktori yang namanya dimulai dengan “assetroot” di direktori sementara sesi.
- Jalur file input telah dipindahkan relatif ke direktori “assetroot” alih-alih relatif terhadap jalur root untuk manifes input pekerjaan (). “/”
- File aturan pemetaan jalur berisi aturan tambahan yang memetakan ulang “/” ke jalur absolut direktori “assetroot”.

Misalnya:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
```

```

2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }

```

Note

Jika pekerjaan yang Anda kirimkan memiliki beberapa manifes dengan jalur root yang berbeda, ada direktori bernama “assetroot” yang berbeda untuk setiap jalur root.

Jika Anda perlu mereferensikan lokasi sistem file yang dipindahkan dari salah satu file input, direktori, atau lokasi sistem file Anda, Anda dapat memproses file aturan pemetaan jalur dalam pekerjaan Anda dan melakukan pemetaan ulang sendiri, atau menambahkan parameter PATH jenis pekerjaan ke template pekerjaan di bundel pekerjaan Anda dan meneruskan nilai yang Anda perlukan untuk memetakan ulang sebagai nilai parameter itu. Misalnya, contoh berikut memodifikasi bundel pekerjaan untuk memiliki salah satu parameter pekerjaan ini dan kemudian mengirimkan pekerjaan dengan lokasi sistem file /shared/projects/project2 sebagai nilainya:

```

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"

```

```
parameterDefinitions:
- name: LocationToRemap
  type: PATH
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
          - "The location of {{RawParam.LocationToRemap}} in the session is
            {{Param.LocationToRemap}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
job_attachments_devguide/ \
-p LocationToRemap=/shared/projects/project2
```

File log untuk menjalankan pekerjaan ini berisi outputnya:

```
2024-07-17 01:40:35,283 INFO Output:
2024-07-17 01:40:35,284 INFO The location of /shared/projects/project2 in the session
is /sessions/session-5b33f/assetroot-assetroot-3751a
```

Mendapatkan file output dari pekerjaan

Contoh ini menunjukkan bagaimana Deadline Cloud mengidentifikasi file output yang dihasilkan oleh pekerjaan Anda, memutuskan apakah akan mengunggah file tersebut ke Amazon S3, dan bagaimana Anda bisa mendapatkan file output tersebut di workstation Anda.

Gunakan bundel `job_attachments_devguide_output` pekerjaan alih-alih bundel `job_attachments_devguide` pekerjaan untuk contoh ini. Mulailah dengan membuat salinan bundel di AWS CloudShell lingkungan Anda dari tiruan repositori sampel GitHub Deadline Cloud:

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/
```

Perbedaan penting antara bundel pekerjaan ini dan bundel `job_attachments_devguide` pekerjaan adalah penambahan parameter pekerjaan baru di templat pekerjaan:

```
...
parameterDefinitions:
```

```
...
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

`dataFlow` properti parameter memiliki nilai `OUT`. Deadline Cloud menggunakan nilai parameter `dataFlow` pekerjaan dengan nilai `OUT` atau `INOUT` sebagai output dari pekerjaan Anda. Jika lokasi sistem file diteruskan sebagai nilai ke jenis parameter pekerjaan ini dipetakan ulang ke lokasi sistem file lokal pada pekerja yang menjalankan pekerjaan, maka Deadline Cloud akan mencari file baru di lokasi dan mengunggahnya ke Amazon S3 sebagai output pekerjaan.

Untuk melihat cara kerjanya, pertama-tama mulai agen pekerja Deadline Cloud di AWS CloudShell tab. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan. Kemudian hapus log pekerjaan dari direktori log:

```
rm -rf ~/devdemo-logs/queue-*
```

Selanjutnya, kirimkan pekerjaan dengan bundel pekerjaan ini. Setelah pekerja berjalan dalam CloudShell proses Anda, lihat log:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output
```

Log menunjukkan bahwa file terdeteksi sebagai output dan diunggah ke Amazon S3:

```
2024-07-17 02:13:10,873 INFO -----
2024-07-17 02:13:10,873 INFO Uploading output files to Job Attachments
2024-07-17 02:13:10,873 INFO -----
```

```

2024-07-17 02:13:10,873 INFO Started syncing outputs using Job Attachments
2024-07-17 02:13:10,955 INFO Found 1 file totaling 117.0 B in output directory: /
sessions/session-7efa/assetroot-assetroot-3751a/output_dir
2024-07-17 02:13:10,956 INFO Uploading output manifest to
DeadlineCloud/Manifests/farm-0011/queue-2233/job-4455/step-6677/
task-6677-0/2024-07-17T02:13:10.835545Z_sessionaction-8899-1/
c6808439dfc59f86763aff5b07b9a76c_output
2024-07-17 02:13:10,988 INFO Uploading 1 output file to S3: s3BucketName/DeadlineCloud/
Data
2024-07-17 02:13:11,011 INFO Uploaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
B/s)
2024-07-17 02:13:11,011 INFO Summary Statistics for file uploads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.02281 seconds at 5.13 KB/s.

```

Log juga menunjukkan bahwa Deadline Cloud membuat objek manifes baru di bucket Amazon S3 yang dikonfigurasi untuk digunakan oleh lampiran pekerjaan pada antrian. Q1 Nama objek manifes berasal dari pertanian, antrian, pekerjaan, langkah, tugas, stempel waktu, dan sessionaction pengidentifikasi tugas yang menghasilkan output. Unduh file manifes ini untuk melihat di mana Deadline Cloud menempatkan file output untuk tugas ini:

```

# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

# Fill this in with the object name from your log
OBJECT_KEY="DeadlineCloud/Manifests/..."

aws s3 cp --quiet s3://$Q1_S3_BUCKET/$OBJECT_KEY /dev/stdout | jq .

```

Manifes terlihat seperti:

```

{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,

```

```

    "path": "output_dir/output.txt",
    "size": 117
  },
  "totalSize": 117
}
```

Ini menunjukkan bahwa konten file output disimpan ke Amazon S3 dengan cara yang sama seperti file input pekerjaan disimpan. Mirip dengan file input, file output disimpan dalam S3 dengan nama objek yang berisi hash file dan awalan. DeadlineCloud/Data

```

$ aws s3 ls --recursive s3://$Q1_S3_BUCKET | grep 34178940e1ef9956db8ea7f7c97ed842
2024-07-17 02:13:11      117 DeadlineCloud/
Data/34178940e1ef9956db8ea7f7c97ed842.xxh128
```

Anda dapat mengunduh output pekerjaan ke workstation Anda menggunakan monitor Deadline Cloud atau Deadline Cloud CLI:

```
deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID
```

Nilai parameter OutputDir pekerjaan dalam pekerjaan yang dikirimkan adalah ./output_dir, sehingga output diunduh ke direktori yang disebut output_dir dalam direktori bundel pekerjaan. Jika Anda menentukan jalur absolut atau lokasi relatif yang berbeda sebagai nilainyaOutputDir, maka file output akan diunduh ke lokasi itu sebagai gantinya.

```

$ deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id
$JOB_ID
```

Downloading output from Job 'Job Attachments Explorer: Output'

Summary of files to download:

```

/home/cloudshell-user/job_attachments_devguide_output/output_dir/output.txt (1
file)
```

You are about to download files which may come from multiple root directories. Here are a list of the current root directories:

```
[0] /home/cloudshell-user/job_attachments_devguide_output
```

> Please enter the index of root directory to edit, y to proceed without changes, or n to cancel the download (0, y, n) [y]:

```
Downloading Outputs [#####] 100%
```

Download Summary:

Downloaded 1 files totaling 117.0 B.

Total download time of 0.14189 seconds at 824.0 B/s.

Download locations (total file counts):

/home/cloudshell-user/job_attachments_devguide_output (1 file)

Menggunakan file dari langkah dalam langkah dependen

Contoh ini menunjukkan bagaimana satu langkah dalam pekerjaan dapat mengakses output dari langkah yang bergantung pada pekerjaan yang sama.

Untuk membuat output dari satu langkah tersedia untuk yang lain, Deadline Cloud menambahkan tindakan tambahan ke sesi untuk mengunduh output tersebut sebelum menjalankan tugas dalam sesi. Anda memberi tahu langkah mana untuk mengunduh output dengan mendeklarasikan langkah-langkah tersebut sebagai dependensi dari langkah yang perlu menggunakan output.

Gunakan bundel `job_attachments_devguide_output` pekerjaan untuk contoh ini. Mulailah dengan membuat salinan di AWS CloudShell lingkungan Anda dari tiruan repositori sampel GitHub Deadline Cloud. Ubah untuk menambahkan langkah dependen yang hanya berjalan setelah langkah yang ada dan menggunakan output langkah itu:

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/

cat >> job_attachments_devguide_output/template.yaml << EOF
- name: DependentStep
  dependencies:
  - dependsOn: Step
  script:
    actions:
      onRun:
        command: /bin/cat
        args:
        - "{{Param.OutputDir}}/output.txt"
EOF
```

Pekerjaan yang dibuat dengan bundel pekerjaan yang dimodifikasi ini berjalan sebagai dua sesi terpisah, satu untuk tugas di langkah "Langkah" dan kemudian yang kedua untuk tugas di langkah "DependentStep".

Pertama mulai agen pekerja Deadline Cloud di CloudShell tab. Biarkan pekerjaan yang dikirimkan sebelumnya selesai berjalan, lalu hapus log pekerjaan dari direktori log:

```
rm -rf ~/devdemo-logs/queue-*
```

Selanjutnya, kirimkan pekerjaan menggunakan bundel `job_attachments_devguide_output` pekerjaan yang dimodifikasi. Tunggu sampai selesai berjalan pada pekerja di CloudShell lingkungan Anda. Lihatlah log untuk dua sesi:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output

# Wait for the job to finish running, and then:

cat demoenv-logs/queue-*/session-*
```

Dalam log sesi untuk tugas di langkah bernama `DependentStep`, ada dua tindakan unduhan terpisah yang dijalankan:

```
2024-07-17 02:52:05,666 INFO =====
2024-07-17 02:52:05,666 INFO ----- Job Attachments Download for Job
2024-07-17 02:52:05,667 INFO =====
2024-07-17 02:52:05,667 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:05,928 INFO Downloaded 207.0 B / 207.0 B of 1 file (Transfer rate: 0.0
B/s)
2024-07-17 02:52:05,929 INFO Summary Statistics for file downloads:
Processed 1 file totaling 207.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03954 seconds at 5.23 KB/s.

2024-07-17 02:52:05,979 INFO
2024-07-17 02:52:05,979 INFO =====
2024-07-17 02:52:05,979 INFO ----- Job Attachments Download for Step
2024-07-17 02:52:05,979 INFO =====
2024-07-17 02:52:05,980 INFO Syncing inputs using Job Attachments
```

```
2024-07-17 02:52:06,133 INFO Downloaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0 B/s)
2024-07-17 02:52:06,134 INFO Summary Statistics for file downloads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03227 seconds at 3.62 KB/s.
```

Tindakan pertama mengunduh `script.sh` file yang digunakan oleh langkah bernama “Langkah.” Tindakan kedua mengunduh output dari langkah itu. Deadline Cloud menentukan file mana yang akan diunduh dengan menggunakan manifes keluaran yang dihasilkan oleh langkah tersebut sebagai manifes input.

Di akhir log yang sama, Anda dapat melihat output dari langkah bernama “DependentStep”:

```
2024-07-17 02:52:06,213 INFO Output:
2024-07-17 02:52:06,216 INFO Script location: /sessions/session-5b33f/
assetroot-assetroot-3751a/script.sh
```

Buat batas sumber daya untuk pekerjaan

Pekerjaan yang dikirimkan ke Deadline Cloud mungkin bergantung pada sumber daya yang dibagi di antara beberapa pekerjaan. Misalnya, sebuah peternakan mungkin memiliki lebih banyak pekerja daripada lisensi mengambang untuk sumber daya tertentu. Atau server file bersama mungkin hanya dapat melayani data ke sejumlah pekerja terbatas pada saat yang bersamaan. Dalam beberapa kasus, satu atau lebih pekerjaan dapat mengklaim semua sumber daya ini, menyebabkan kesalahan karena sumber daya yang tidak tersedia ketika pekerja baru mulai.

Untuk membantu mengatasi hal ini, Anda dapat menggunakan batas untuk sumber daya terbatas ini. Deadline Cloud memperhitungkan ketersediaan sumber daya terbatas dan menggunakan informasi tersebut untuk memastikan bahwa sumber daya tersedia saat pekerja baru memulai sehingga pekerjaan memiliki kemungkinan gagal yang lebih rendah karena sumber daya yang tidak tersedia.

Batas dibuat untuk seluruh peternakan. Pekerjaan yang dikirimkan ke antrian hanya dapat memperoleh batas yang terkait dengan antrian. Jika Anda menentukan batas untuk pekerjaan yang tidak terkait dengan antrian, pekerjaan tersebut tidak kompatibel dan tidak akan berjalan.

Untuk menggunakan batas, Anda

- [Buat batas](#)

- [Kaitkan batas dan antrian](#)
- [Kirim pekerjaan yang membutuhkan batasan](#)

Note

Jika Anda menjalankan pekerjaan yang memiliki sumber daya terbatas dalam antrian yang tidak terkait dengan batas, pekerjaan itu dapat menghabiskan semua sumber daya. Jika Anda memiliki sumber daya terbatas, pastikan bahwa semua langkah dalam pekerjaan dalam antrian yang menggunakan sumber daya dikaitkan dengan batas.

Untuk batas yang ditentukan di peternakan, terkait dengan antrian, dan ditentukan dalam pekerjaan, salah satu dari empat hal dapat terjadi:

- Jika Anda membuat batas, kaitkan dengan antrian, dan tentukan batas dalam templat pekerjaan, pekerjaan akan berjalan dan hanya menggunakan sumber daya yang ditentukan dalam batas.
- Jika Anda membuat batas, tentukan dalam templat pekerjaan, tetapi jangan mengaitkan batas dengan antrian, pekerjaan ditandai tidak kompatibel dan tidak akan berjalan.
- Jika Anda membuat batas, jangan mengaitkannya dengan antrian, dan jangan tentukan batas dalam templat pekerjaan, pekerjaan berjalan tetapi tidak menggunakan batas.
- Jika Anda tidak menggunakan batas sama sekali, pekerjaan berjalan.

Jika Anda mengaitkan batas ke beberapa antrian, antrian berbagi sumber daya yang dibatasi oleh batas. Misalnya, jika Anda membuat batas 100, dan satu antrian menggunakan 60 sumber daya, antrian lain hanya dapat menggunakan 40 sumber daya. Ketika sumber daya dirilis, itu dapat diambil oleh tugas dari antrian apa pun.

Deadline Cloud menyediakan dua AWS CloudFormation metrik untuk membantu Anda memantau sumber daya yang disediakan oleh batas. Anda dapat memantau jumlah sumber daya saat ini yang digunakan dan jumlah maksimum sumber daya yang tersedia dalam batas. Untuk informasi selengkapnya, lihat [Metrik batas sumber daya](#) di Panduan Pengembang Cloud Batas Waktu.

Anda menerapkan batas untuk langkah pekerjaan dalam template pekerjaan. Saat Anda menentukan nama persyaratan jumlah batas di `amounts` bagian langkah dan batas yang sama `amountRequirementName` dikaitkan dengan antrian pekerjaan, tugas yang dijadwalkan untuk langkah ini dibatasi oleh batas sumber daya. `hostRequirements`

Jika sebuah langkah membutuhkan sumber daya yang dibatasi oleh batas yang tercapai, tugas dalam langkah itu tidak akan diambil oleh pekerja tambahan.

Anda dapat menerapkan lebih dari satu batas untuk langkah pekerjaan. Misalnya, jika langkah menggunakan dua lisensi perangkat lunak yang berbeda, Anda dapat menerapkan batas terpisah untuk setiap lisensi. Jika sebuah langkah membutuhkan dua batasan dan batas untuk salah satu sumber daya tercapai, tugas dalam langkah itu tidak akan diambil oleh pekerja tambahan sampai sumber daya tersedia.

Menghentikan dan menghapus batas

Saat Anda menghentikan atau menghapus asosiasi antara antrian dan batas, pekerjaan yang menggunakan batas menghentikan penjadwalan tugas dari langkah-langkah yang memerlukan batas ini dan memblokir pembuatan sesi baru untuk satu langkah.

Tugas yang berada dalam status READY tetap siap, dan tugas secara otomatis dilanjutkan dengan asosiasi antara antrian dan batas menjadi aktif kembali. Anda tidak perlu meminta pekerjaan apa pun.

Ketika Anda menghentikan atau menghapus asosiasi antara antrian dan batas, Anda memiliki dua pilihan tentang cara menghentikan menjalankan tugas:

- Hentikan dan batalkan tugas — Pekerja dengan sesi yang memperoleh batas membatalkan semua tugas.
- Berhenti dan selesaikan tugas yang sedang berjalan — Pekerja dengan sesi yang memperoleh batas menyelesaikan tugas mereka.

Ketika Anda menghapus batas menggunakan konsol, pekerja pertama-tama berhenti menjalankan tugas segera atau akhirnya ketika mereka selesai. Ketika asosiasi dihapus, hal berikut terjadi:

- Langkah-langkah yang membutuhkan batas ditandai tidak kompatibel.
- Seluruh pekerjaan yang berisi langkah-langkah tersebut dibatalkan, termasuk langkah-langkah yang tidak memerlukan batas.
- Pekerjaan ditandai tidak kompatibel.

Jika antrian yang terkait dengan batas memiliki armada terkait dengan kemampuan armada yang sesuai dengan jumlah persyaratan nama batas, armada tersebut akan terus memproses pekerjaan dengan batas yang ditentukan.

Buat batas

Anda membuat batas menggunakan konsol Deadline Cloud atau [CreateLimit operasi di Deadline Cloud API](#). Batas didefinisikan untuk pertanian, tetapi terkait dengan antrian. Setelah Anda membuat batas, Anda dapat mengaitkannya dengan satu atau lebih antrian.

Untuk membuat batas

1. Dari dasbor Deadline Cloud console (<https://console.aws.amazon.com/deadlinecloud/home>), pilih farm yang ingin Anda buat antrean.
2. Pilih peternakan untuk menambahkan batas, pilih tab Batas, lalu pilih Buat batas.
3. Berikan detail untuk batasnya. Nama persyaratan Jumlah adalah nama yang digunakan dalam template pekerjaan untuk mengidentifikasi batas. Itu harus dimulai dengan awalan **amount**. diikuti dengan nama jumlah. Nama persyaratan jumlah harus unik dalam antrian yang terkait dengan batas.
4. Jika Anda memilih Tetapkan jumlah maksimum, itu adalah jumlah total sumber daya yang diizinkan oleh batas ini. Jika Anda memilih Tidak ada jumlah maksimum, penggunaan sumber daya tidak terbatas. Bahkan ketika penggunaan sumber daya tidak terbatas, CloudWatch metrik CurrentCount Amazon dipancarkan sehingga Anda dapat melacak penggunaan. Untuk informasi selengkapnya, lihat [CloudWatchmetrik](#) di Panduan Pengembang Cloud Deadline.
5. Jika Anda sudah tahu antrian yang harus menggunakan batas, Anda dapat memilihnya sekarang. Anda tidak perlu mengaitkan antrian untuk membuat batas.
6. Pilih Buat batas.

Kaitkan batas dan antrian

Setelah Anda membuat batas, Anda dapat mengaitkan satu atau lebih antrian dengan batas. Hanya antrian yang terkait dengan batas yang menggunakan nilai yang ditentukan dalam batas.

Anda membuat asosiasi dengan antrian menggunakan konsol Deadline Cloud atau [CreateQueueLimitAssociation operasi di Deadline Cloud API](#).

Untuk mengaitkan antrian dengan batas

1. Dari dasbor Deadline Cloud console (<https://console.aws.amazon.com/deadlinecloud/home>), pilih farm tempat Anda ingin mengaitkan batas dengan antrian.
2. Pilih tab Limits, pilih limit untuk mengaitkan antrian dengan, lalu pilih Edit limit.

3. Di bagian antrian Associate, pilih antrian yang akan dikaitkan dengan batas.
4. Pilih Simpan perubahan.

Kirim pekerjaan yang membutuhkan batasan

Anda menerapkan batas dengan menentukannya sebagai persyaratan tuan rumah untuk pekerjaan atau langkah pekerjaan. Jika Anda tidak menentukan batas dalam satu langkah dan langkah itu menggunakan sumber daya terkait, penggunaan langkah tidak dihitung terhadap batas saat pekerjaan dijadwalkan..

Beberapa submitter Deadline Cloud memungkinkan Anda untuk menetapkan persyaratan host. Anda dapat menentukan nama persyaratan jumlah batas di pengirim untuk menerapkan batas.

Jika pengirim Anda tidak mendukung penambahan persyaratan host, Anda juga dapat menerapkan batasan dengan mengedit templat pekerjaan untuk pekerjaan itu.

Untuk menerapkan batas pada langkah pekerjaan dalam bundel pekerjaan

1. Buka template pekerjaan untuk pekerjaan menggunakan editor teks. Template pekerjaan terletak di direktori bundel pekerjaan untuk pekerjaan itu. Untuk informasi selengkapnya, lihat [Paket Job](#) di Panduan Pengembang Cloud Deadline.
2. Temukan definisi langkah untuk langkah yang menerapkan batas.
3. Tambahkan yang berikut ini ke definisi langkah. Ganti *amount.name* dengan nama persyaratan jumlah batas Anda. Untuk penggunaan umum, Anda harus menetapkan min nilai ke 1.

YAML

```
hostRequirements:
  amounts:
    - name: amount.name
      min: 1
```

JSON

```
"hostRequirements": {
  "amounts": [
    {
      "name": "amount.name",
      "min": "1"
```

```
}  
}  
}
```

Anda dapat menambahkan beberapa batasan ke langkah pekerjaan sebagai berikut. Ganti *amount.name_1* dan *amount.name_2* dengan nama persyaratan jumlah batas Anda.

YAML

```
hostRequirements:  
  amounts:  
    - name: amount.name_1  
      min: 1  
    - name: amount.name_2  
      min: 1
```

JSON

```
"hostRequirements": {  
  "amounts": [  
    {  
      "name": "amount.name_1",  
      "min": "1"  
    },  
    {  
      "name": "amount.name_2",  
      "min": "1"  
    }  
  ]  
}
```

4. Simpan perubahan pada template pekerjaan.

Cara mengirimkan pekerjaan ke Deadline Cloud

Ada banyak cara berbeda untuk mengirimkan pekerjaan ke AWS Deadline Cloud. Bagian ini menjelaskan beberapa cara Anda dapat mengirimkan pekerjaan menggunakan alat yang disediakan oleh Deadline Cloud atau dengan membuat alat kustom Anda sendiri untuk beban kerja Anda.

- Dari terminal — untuk saat Anda pertama kali mengembangkan paket pekerjaan, atau saat pengguna mengirimkan pekerjaan merasa nyaman menggunakan baris perintah
- Dari skrip - untuk menyesuaikan dan mengotomatiskan beban kerja
- Dari aplikasi — untuk saat pekerjaan pengguna berada dalam aplikasi, atau ketika konteks aplikasi penting.

Contoh berikut menggunakan pustaka `deadline` Python dan alat baris `deadline` perintah. Keduanya tersedia dari [PyPi](#) dan [di-host GitHub](#).

Topik

- [Kirim pekerjaan ke Deadline Cloud dari terminal](#)
- [Kirim pekerjaan ke Deadline Cloud menggunakan skrip](#)
- [Kirim pekerjaan dalam aplikasi](#)

Kirim pekerjaan ke Deadline Cloud dari terminal

Dengan hanya menggunakan bundel pekerjaan dan Deadline Cloud CLI, Anda atau pengguna Anda yang lebih teknis dapat dengan cepat mengulangi penulisan bundel pekerjaan untuk menguji pengiriman pekerjaan. Gunakan perintah berikut untuk mengirimkan bundel pekerjaan:

```
deadline bundle submit <path-to-job-bundle>
```

Jika Anda mengirimkan bundel pekerjaan dengan parameter yang tidak memiliki default dalam bundel, Anda dapat menentukannya dengan opsi/. `-p --parameter`

```
deadline bundle submit <path-to-job-bundle> -p <parameter-name>=<parameter-value> -  
p ...
```

Untuk daftar lengkap opsi yang tersedia, jalankan perintah bantuan:

```
deadline bundle submit --help
```

Kirim pekerjaan ke Deadline Cloud menggunakan GUI

Deadline Cloud CLI juga dilengkapi dengan antarmuka pengguna grafis yang memungkinkan pengguna untuk melihat parameter yang harus mereka berikan sebelum mengirimkan pekerjaan. Jika

pengguna Anda memilih untuk tidak berinteraksi dengan baris perintah, Anda dapat menulis pintasan desktop yang membuka dialog untuk mengirimkan bundel pekerjaan tertentu:

```
deadline bundle gui-submit <path-to-job-bundle>
```

Gunakan `--browse` opsi ini sehingga pengguna dapat memilih bundel pekerjaan:

```
deadline bundle gui-submit --browse
```

Untuk daftar lengkap opsi yang tersedia, jalankan perintah bantuan:

```
deadline bundle gui-submit --help
```

Kirim pekerjaan ke Deadline Cloud menggunakan skrip

Untuk mengotomatiskan pengiriman pekerjaan ke Deadline Cloud, Anda dapat membuat skrip menggunakan alat seperti bash, Powershell, dan file batch.

Anda dapat menambahkan fungsionalitas seperti mengisi parameter pekerjaan dari variabel lingkungan atau aplikasi lain. Anda juga dapat mengirimkan beberapa pekerjaan berturut-turut, atau membuat skrip pembuatan bundel pekerjaan untuk dikirimkan.

Kirim pekerjaan menggunakan Python

Deadline Cloud juga memiliki pustaka Python open-source untuk berinteraksi dengan layanan. [Kode sumber tersedia di GitHub](#).

Pustaka tersedia di pypi melalui pip (`pip install deadline`). Ini adalah perpustakaan yang sama yang digunakan oleh alat Deadline Cloud CLI:

```
from deadline.client import api

job_bundle_path = "/path/to/job/bundle"
job_parameters = [
    {
        "name": "parameter_name",
        "value": "parameter_value"
    },
]
```

```
job_id = api.create_job_from_job_bundle(  
    job_bundle_path,  
    job_parameters  
)  
print(job_id)
```

Untuk membuat dialog seperti `deadline bundle gui-submit` perintah, Anda dapat menggunakan `show_job_bundle_submitter` fungsi dari [deadline.client.ui.job_bundle_submitter](#).

Contoh berikut memulai aplikasi Qt dan menunjukkan pengirim bundel pekerjaan:

```
# The GUI components must be installed with pip install "deadline[gui]"  
import sys  
from qtpy.QtWidgets import QApplication  
from deadline.client.ui.job_bundle_submitter import show_job_bundle_submitter  
  
app = QApplication(sys.argv)  
submitter = show_job_bundle_submitter(browse=True)  
submitter.show()  
app.exec()  
print(submitter.create_job_response)
```

Untuk membuat dialog Anda sendiri, Anda dapat menggunakan `SubmitJobToDeadlineDialog` kelas di [deadline.client.ui.dialogs.submit_job_to_deadline_dialog](#). Anda dapat meneruskan nilai, menyematkan tab spesifik pekerjaan Anda sendiri, dan menentukan bagaimana bundel pekerjaan dibuat (atau diteruskan).

Kirim pekerjaan dalam aplikasi

Untuk memudahkan pengguna mengirimkan pekerjaan, Anda dapat menggunakan runtime scripting atau sistem plugin yang disediakan oleh aplikasi. Pengguna memiliki antarmuka yang akrab dan Anda dapat membuat alat canggih yang membantu pengguna saat mengirimkan beban kerja.

Sematkan bundel pekerjaan dalam aplikasi

Contoh ini menunjukkan pengiriman bundel pekerjaan yang Anda sediakan dalam aplikasi.

Untuk memberi pengguna akses ke bundel pekerjaan ini, buat skrip yang disematkan dalam item menu yang meluncurkan CLI Deadline Cloud.

Skrip berikut memungkinkan pengguna untuk memilih bundel pekerjaan:

```
deadline bundle gui-submit --install-gui
```

Untuk menggunakan bundel pekerjaan tertentu dalam item menu sebagai gantinya, gunakan yang berikut ini:

```
deadline bundle gui-submit </path/to/job/bundle> --install-gui
```

Ini membuka dialog di mana pengguna dapat memodifikasi parameter pekerjaan, input, dan output, dan kemudian mengirimkan pekerjaan. Anda dapat memiliki item menu yang berbeda untuk bundel pekerjaan yang berbeda bagi pengguna untuk mengirimkan dalam aplikasi.

Jika job yang Anda kirimkan dengan bundel pekerjaan berisi parameter dan referensi aset yang serupa di seluruh kiriman, Anda dapat mengisi nilai default di bundel pekerjaan yang mendasarinya.

Dapatkan informasi dari aplikasi

Untuk menarik informasi dari aplikasi sehingga pengguna tidak perlu menambahkannya secara manual ke kiriman, Anda dapat mengintegrasikan Deadline Cloud dengan aplikasi sehingga pengguna Anda dapat mengirimkan pekerjaan menggunakan antarmuka yang sudah dikenal tanpa perlu keluar dari aplikasi atau menggunakan alat baris perintah.

[Jika aplikasi Anda memiliki runtime scripting yang mendukung Python dan pyside/pyqt, Anda dapat menggunakan komponen GUI dari pustaka klien Deadline Cloud untuk membuat UI.](#) Sebagai contoh, lihat [Deadline Cloud untuk integrasi Maya](#) pada GitHub.

Pustaka klien Deadline Cloud menyediakan operasi yang melakukan hal berikut untuk membantu Anda memberikan pengalaman pengguna terintegrasi yang kuat:

- Tarik parameter lingkungan antrian, parameter pekerjaan, dan referensi aset membentuk variabel lingkungan dan dengan memanggil SDK aplikasi.
- Atur parameter dalam bundel pekerjaan. Untuk menghindari memodifikasi bundel asli, Anda harus membuat salinan bundel dan mengirimkan salinannya.

Jika Anda menggunakan `deadline bundle gui-submit` perintah untuk mengirimkan bundel pekerjaan, Anda harus secara terprogram `asset_references.yaml` file `parameter_values.yaml` dan untuk meneruskan informasi dari aplikasi. Untuk informasi selengkapnya tentang file-file ini, lihat [Templat Open Job Description \(OpenJD\) untuk Deadline Cloud](#).

Jika Anda memerlukan kontrol yang lebih kompleks daripada yang ditawarkan oleh OpenJD, perlu mengabstraksi pekerjaan dari pengguna, atau ingin membuat integrasi cocok dengan gaya visual aplikasi, Anda dapat menulis dialog Anda sendiri yang memanggil pustaka klien Deadline Cloud untuk mengirimkan pekerjaan.

Jadwalkan pekerjaan di Deadline Cloud

Setelah pekerjaan dibuat, AWS Deadline Cloud menjadwalkannya untuk diproses pada satu atau lebih armada yang terkait dengan antrian. Armada yang memproses tugas tertentu dipilih berdasarkan kemampuan yang dikonfigurasi untuk armada dan persyaratan tuan rumah dari langkah tertentu.

Pekerjaan dalam antrian dijadwalkan dalam urutan prioritas upaya terbaik, tertinggi ke terendah. Ketika dua pekerjaan memiliki prioritas yang sama, pekerjaan tertua dijadwalkan terlebih dahulu.

Bagian berikut memberikan rincian proses penjadwalan pekerjaan.

Tentukan kompatibilitas armada

Setelah pekerjaan dibuat, Deadline Cloud memeriksa persyaratan host untuk setiap langkah dalam pekerjaan terhadap kemampuan armada yang terkait dengan antrian pekerjaan yang diajukan. Jika armada memenuhi persyaratan tuan rumah, pekerjaan dimasukkan ke READY negara bagian.

Jika ada langkah dalam pekerjaan yang memiliki persyaratan yang tidak dapat dipenuhi oleh armada yang terkait dengan antrian, status langkah diatur ke NOT_COMPATIBLE. Selain itu, sisa langkah dalam pekerjaan dibatalkan.

Kemampuan untuk armada ditetapkan pada tingkat armada. Bahkan jika seorang pekerja dalam armada memenuhi persyaratan pekerjaan, itu tidak akan diberikan tugas dari pekerjaan jika armadanya tidak memenuhi persyaratan pekerjaan.

Template pekerjaan berikut memiliki langkah yang menentukan persyaratan host untuk langkah tersebut:

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
```

```

actions:
  onRun:
    args:
      - '1'
    command: /usr/bin/sleep
hostRequirements:
  amounts:
    # Capabilities starting with "amount." are amount capabilities. If they start with
    "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
    usage.
    - name: amount.worker.vcpu
      min: 4
      max: 8
  attributes:
    - name: attr.worker.os.family
      anyOf:
        - linux

```

Pekerjaan ini dapat dijadwalkan ke armada dengan kemampuan sebagai berikut:

```

{
  "vCpuCount": {"min": 4, "max": 8},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}

```

Pekerjaan ini tidak dapat dijadwalkan ke armada dengan salah satu kemampuan berikut:

```

{
  "vCpuCount": {"min": 4},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}

```

The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

```

{
  "vCpuCount": {"max": 8},
  "memoryMiB": {"min": 1024},
  "osFamily": "linux",
  "cpuArchitectureType": "x86_64"
}

```

```
}
    The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host
    requirement.

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
    The osFamily doesn't match.
```

Penskalaan armada

Ketika pekerjaan ditugaskan ke armada yang dikelola layanan yang kompatibel, armada diskalakan secara otomatis. Jumlah pekerja di armada berubah berdasarkan jumlah tugas yang tersedia untuk dijalankan armada.

Ketika pekerjaan ditugaskan ke armada yang dikelola pelanggan, pekerja mungkin sudah ada atau dapat dibuat menggunakan penskalaan otomatis berbasis peristiwa. Untuk informasi selengkapnya, lihat [Menggunakan EventBridge untuk menangani peristiwa penskalaan otomatis](#) di Panduan Pengguna Penskalaan EC2 Otomatis Amazon.

Sesi

Tugas dalam suatu pekerjaan dibagi menjadi satu atau lebih sesi. Pekerja menjalankan sesi untuk mengatur lingkungan, menjalankan tugas, dan kemudian meruntuhkan lingkungan. Setiap sesi terdiri dari satu atau lebih tindakan yang harus dilakukan seorang pekerja.

Saat pekerja menyelesaikan tindakan bagian, tindakan sesi tambahan dapat dikirim ke pekerja. Pekerja menggunakan kembali lingkungan yang ada dan lampiran pekerjaan dalam sesi untuk menyelesaikan tugas dengan lebih efisien.

Lampiran Job dibuat oleh pengirim yang Anda gunakan sebagai bagian dari paket pekerjaan Deadline Cloud CLI Anda. Anda juga dapat membuat lampiran pekerjaan menggunakan -- attachments opsi untuk create-job AWS CLI perintah. Lingkungan didefinisikan di dua tempat: lingkungan antrian yang dilampirkan ke antrian tertentu, dan lingkungan pekerjaan dan langkah yang ditentukan dalam templat pekerjaan.

Ada empat jenis tindakan sesi:

- `syncInputJobAttachments`— Mengunduh lampiran pekerjaan input ke pekerja.
- `envEnter`— Melakukan `onEnter` tindakan untuk suatu lingkungan.
- `taskRun`— Melakukan `onRun` tindakan untuk suatu tugas.
- `envExit`— Melakukan `onExit` tindakan untuk suatu lingkungan.

Template pekerjaan berikut memiliki lingkungan langkah. Ini memiliki `onEnter` definisi untuk mengatur lingkungan langkah, `onRun` definisi yang mendefinisikan tugas yang akan dijalankan, dan `onExit` definisi untuk meruntuhkan lingkungan langkah. Sesi yang dibuat untuk pekerjaan ini akan mencakup `envEnter` tindakan, satu atau lebih `taskRun` tindakan, dan kemudian `envExit` tindakan.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
    actions:
      onEnter:
        command: MayaAdaptor
        args:
        - daemon
        - start
        - --init-data
        - file//{{Env.File.initData}}
      onExit:
        command: MayaAdaptor
        args:
        - daemon
        - stop
  parameterSpace:
    taskParameterDefinitions:
```

```
- name: Frame
  range: 1-5
  type: INT
script:
  embeddedFiles:
  - name: runData
    filename: run-data.yaml
    type: TEXT
    data: |
      frame: {{Task.Param.Frame}}
actions:
  onRun:
    command: MayaAdaptor
    args:
      - daemon
      - run
      - --run-data
      - file//{{ Task.File.runData }}
```

Tindakan sesi pipelining

Tindakan sesi pipelining memungkinkan penjadwal pra-menetapkan beberapa tindakan sesi ke pekerja. Pekerja kemudian dapat menjalankan tindakan ini secara berurutan, mengurangi atau menghilangkan waktu idle antar tugas.

Untuk membuat tugas awal, penjadwal membuat sesi dengan satu tugas, pekerja menyelesaikan tugas, dan kemudian penjadwal menganalisis durasi tugas untuk menentukan tugas masa depan.

Agar penjadwal efektif, ada aturan durasi tugas. Untuk tugas di bawah satu menit, penjadwal menggunakan pola pertumbuhan power-of-2. Misalnya, untuk tugas 1 detik, penjadwal memberikan 2 tugas baru, lalu 4, lalu 8. Untuk tugas lebih dari satu menit, penjadwal hanya menetapkan satu tugas baru dan pipelining tetap dinonaktifkan.

Untuk menghitung ukuran pipa, penjadwal melakukan hal berikut:

- Menggunakan durasi tugas rata-rata dari tugas yang diselesaikan
- Bertujuan untuk membuat pekerja sibuk selama satu menit
- Mempertimbangkan hanya tugas dalam sesi yang sama
- Tidak berbagi data durasi di seluruh pekerja

Dengan piplelining tindakan sesi, pekerja segera memulai tugas baru dan tidak ada waktu tunggu di antara permintaan penjadwal. Ini juga memberikan peningkatan efisiensi pekerja dan distribusi tugas yang lebih baik untuk proses yang berjalan lama.

Selain itu, jika ada pekerjaan prioritas baru yang lebih tinggi yang tersedia, pekerja akan menyelesaikan semua pekerjaan yang ditugaskan sebelumnya sebelum sesi saat ini berakhir dan sesi baru dari pekerjaan prioritas yang lebih tinggi ditugaskan.

Ketergantungan langkah

Deadline Cloud mendukung mendefinisikan dependensi antar langkah sehingga satu langkah menunggu hingga langkah lain selesai sebelum memulai. Anda dapat menentukan lebih dari satu ketergantungan untuk satu langkah. Langkah dengan ketergantungan tidak dijadwalkan sampai semua dependensinya selesai.

Jika template pekerjaan mendefinisikan ketergantungan melingkar, pekerjaan ditolak dan status pekerjaan disetel ke. `CREATE_FAILED`

Template pekerjaan berikut membuat pekerjaan dengan dua langkah. StepBtergantung padaStepA. StepBhanya berjalan setelah StepA selesai dengan sukses.

Setelah pekerjaan dibuat, StepA berada di `READY` negara bagian dan StepB berada di `PENDING` negara bagian. Setelah StepA selesai, StepB pindah ke `READY` negara bagian. Jika StepA gagal, atau StepA jika dibatalkan, StepB pindah ke `CANCELED` negara bagian.

Anda dapat mengatur ketergantungan pada beberapa langkah. Misalnya, jika StepC tergantung pada keduanya StepA danStepB, StepC tidak akan dimulai sampai dua langkah lainnya selesai.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
```

```
    data: |
      #!/bin/env bash

      set -euo pipefail

      sleep 1
      echo Task A Done!
- name: B
  dependencies:
    - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
  embeddedFiles:
    - name: run
      type: TEXT
      data: |
        #!/bin/env bash

        set -euo pipefail

        sleep 1
        echo Task B Done!
```

Ubah pekerjaan di Deadline Cloud

Anda dapat menggunakan update perintah AWS Command Line Interface (AWS CLI) berikut untuk mengubah konfigurasi pekerjaan, atau untuk menetapkan status target pekerjaan, langkah, atau tugas:

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

Dalam contoh update perintah berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Example — Meminta pekerjaan

Semua tugas dalam pekerjaan beralih ke READY status, kecuali ada dependensi langkah. Langkah-langkah dengan dependensi beralih ke salah satu READY atau PENDING saat dipulihkan.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status PENDING
```

Example — Batalkan pekerjaan

Semua tugas dalam pekerjaan yang tidak memiliki status SUCCEEDED atau FAILED ditandai CANCELED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status CANCELED
```

Example — Tandai pekerjaan gagal

Semua tugas dalam pekerjaan yang memiliki status SUCCEEDED dibiarkan tidak berubah. Semua tugas lainnya ditandai FAILED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status FAILED
```

Example — Tandai pekerjaan yang sukses

Semua tugas dalam pekerjaan pindah ke SUCCEEDED negara bagian.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status SUCCEEDED
```

```
--target-task-run-status SUCCEEDED
```

Example — Menangguhkan pekerjaan

Tugas dalam pekerjaan diSUCCEEDED,CANCELED, atau FAILED negara bagian tidak berubah. Semua tugas lainnya ditandaiSUSPENDED.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--target-task-run-status SUSPENDED
```

Example — Mengubah prioritas pekerjaan

Memperbarui prioritas pekerjaan dalam antrian untuk mengubah urutan yang dijadwalkan. Pekerjaan prioritas yang lebih tinggi umumnya dijadwalkan terlebih dahulu.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--priority 100
```

Example — Ubah jumlah tugas gagal yang diizinkan

Memperbarui jumlah maksimum tugas yang gagal yang dapat dimiliki pekerjaan sebelum tugas yang tersisa dibatalkan.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--max-failed-tasks-count 200
```

Example — Ubah jumlah percobaan ulang tugas yang diizinkan

Memperbarui jumlah maksimum percobaan ulang untuk tugas sebelum tugas gagal. Tugas yang telah mencapai jumlah percobaan ulang maksimum tidak dapat diulang sampai nilai ini meningkat.

```
aws deadline update-job \  

```

```
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--max-retries-per-task 10
```

Example — Arsipkan pekerjaan

Memperbarui status siklus hidup pekerjaan ke. ARCHIVED Pekerjaan yang diarsipkan tidak dapat dijadwalkan atau diubah. Anda hanya dapat mengarsipkan pekerjaan yang ada di FAILED, CANCELED, SUCCEEDED, atau SUSPENDED negara bagian.

```
aws deadline update-job \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--lifecycle-status ARCHIVED
```

Example — Meminta satu langkah

Semua tugas di langkah beralih ke READY status, kecuali ada dependensi langkah. Tugas dalam langkah-langkah dengan dependensi beralih ke salah satu READY atau PENDING, dan tugas dipulihkan.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status PENDING
```

Example — Batalkan langkah

Semua tugas dalam langkah yang tidak memiliki status SUCCEEDED atau FAILED ditandai CANCELED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status CANCELED
```

Example — Tandai langkah gagal

Semua tugas dalam langkah yang memiliki status SUCCEEDED dibiarkan tidak berubah. Semua tugas lainnya ditandai FAILED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status FAILED
```

Example — Tandai langkah sukses

Semua tugas dalam langkah ditandai SUCCEEDED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status SUCCEEDED
```

Example — Tangguhkan satu langkah

Tugas dalam langkah di SUCCEEDED, CANCELED, atau FAILED status tidak berubah. Semua tugas lainnya ditandai SUSPENDED.

```
aws deadline update-step \  
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--target-task-run-status SUSPENDED
```

Example — Mengubah status tugas

Saat Anda menggunakan perintah `update-task` Deadline Cloud CLI, tugas beralih ke status yang ditentukan.

```
aws deadline update-task \  

```

```
--farm-id farmID \  
--queue-id queueID \  
--job-id jobID \  
--step-id stepID \  
--task-id taskID \  
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

Membuat dan menggunakan armada yang dikelola pelanggan Deadline Cloud

Saat membuat armada yang dikelola pelanggan (CMF), Anda memiliki kendali penuh atas pipeline pemrosesan Anda. Anda menentukan lingkungan jaringan dan perangkat lunak untuk setiap pekerja. Deadline Cloud bertindak sebagai repositori dan penjadwal untuk pekerjaan Anda.

Pekerja dapat berupa instans Amazon Elastic Compute Cloud (Amazon EC2), pekerja di fasilitas lokasi bersama, atau pekerja lokal. Setiap pekerja harus menjalankan agen pekerja Deadline Cloud. Semua pekerja harus memiliki akses ke titik [akhir layanan Deadline Cloud](#).

Topik berikut menunjukkan cara membuat CMF dasar menggunakan EC2 instans Amazon.

Topik

- [Buat armada yang dikelola pelanggan](#)
- [Pengaturan dan konfigurasi host pekerja](#)
- [Kelola akses ke rahasia pengguna Windows pekerjaan](#)
- [Instal dan konfigurasi perangkat lunak yang diperlukan untuk pekerjaan](#)
- [Mengkonfigurasi kredensial AWS](#)
- [Konfigurasi jaringan untuk mengizinkan koneksi titik akhir AWS](#)
- [Uji konfigurasi host pekerja Anda](#)
- [Buat sebuah Amazon Machine Image](#)
- [Buat infrastruktur armada dengan grup Amazon EC2 Auto Scaling](#)

Buat armada yang dikelola pelanggan

Untuk membuat armada yang dikelola pelanggan (CMF), selesaikan langkah-langkah berikut.

Deadline Cloud console

Untuk menggunakan konsol Deadline Cloud untuk membuat armada yang dikelola pelanggan

1. Buka [konsol](#) Deadline Cloud.
2. Pilih Peternakan. Daftar pajangan pertanian yang tersedia.

3. Pilih nama Peternakan tempat Anda ingin bekerja.
4. Pilih tab Armada, lalu pilih Buat armada.
5. Masukkan Nama untuk armada Anda.
6. (Opsional) Masukkan Deskripsi untuk armada Anda.
7. Pilih Pelanggan yang dikelola untuk jenis Armada.
8. Pilih akses layanan armada Anda.
 - a. Sebaiknya gunakan opsi Buat dan gunakan peran layanan baru untuk setiap armada untuk kontrol izin yang lebih terperinci. Opsi ini dipilih secara default.
 - b. Anda juga dapat menggunakan peran layanan yang ada dengan memilih Pilih peran layanan.
9. Tinjau pilihan Anda, lalu pilih Berikutnya.
10. Pilih sistem operasi untuk armada Anda. Semua pekerja armada harus memiliki sistem operasi yang sama.
11. Pilih arsitektur CPU host.
12. Pilih kemampuan perangkat keras vCPU dan memori minimum dan maksimum untuk memenuhi tuntutan beban kerja armada Anda.
13. Pilih jenis Auto Scaling. Untuk informasi selengkapnya, lihat [Menggunakan EventBridge untuk menangani peristiwa Auto Scaling](#).
 - Tanpa penskalaan: Anda membuat armada lokal dan ingin memilih keluar dari Deadline Cloud Auto Scaling.
 - Rekomendasi penskalaan: Anda membuat armada Amazon Elastic Compute Cloud EC2 (Amazon).
14. (Opsional) Pilih panah untuk memperluas bagian Tambahkan kemampuan.
15. (Opsional) Pilih kotak centang untuk Tambahkan kemampuan GPU - Opsional, lalu masukkan minimum dan maksimum GPUs dan memori.
16. Tinjau pilihan Anda, lalu pilih Berikutnya.
17. (Opsional) Tentukan kemampuan pekerja khusus, lalu pilih Berikutnya.
18. Menggunakan dropdown, pilih satu atau lebih antrian untuk dikaitkan dengan armada.

Note

Kami merekomendasikan untuk mengaitkan armada hanya dengan antrian yang semuanya berada dalam batas kepercayaan yang sama. Ini memastikan batas keamanan yang kuat antara menjalankan pekerjaan pada pekerja yang sama.

19. Tinjau asosiasi antrian, lalu pilih Berikutnya.
20. (Opsional) Untuk lingkungan antrian Conda Default, kami akan membuat lingkungan untuk antrian Anda yang akan menginstal paket Conda yang diminta oleh pekerjaan.

Note

Lingkungan antrian Conda digunakan untuk menginstal paket Conda yang diminta oleh pekerjaan. Biasanya, Anda harus menghapus centang pada lingkungan antrian Conda pada antrian yang terkait dengan CMFs karena tidak CMFs akan memiliki perintah Conda yang diperlukan diinstal secara default.

21. (Opsional) Tambahkan tag ke CMF Anda. Untuk informasi selengkapnya, lihat [Menandai AWS sumber daya Anda](#).
22. Tinjau konfigurasi armada Anda dan buat perubahan apa pun, lalu pilih Buat armada.
23. Pilih tab Armada, lalu catat ID Armada.

AWS CLI

Untuk menggunakan AWS CLI untuk membuat armada yang dikelola pelanggan

1. Buka terminal.
2. Buat `fleet-trust-policy.json` di editor baru.
 - a. Tambahkan kebijakan IAM berikut, ganti **ITALICIZED** teks dengan ID AWS akun dan ID pertanian Deadline Cloud.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::ITALICIZED:root"      },  
      "Action": "iam:CreateAccessKey",  
      "Resource": "*"
    }
  ]
}
```

```

    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn":
            "arn:aws:deadline:*:111122223333:farm/FARM_ID"
        }
      }
    }
  ]
}

```

b. Simpan perubahan Anda.

3. Buat `fleet-policy.json`.

a. Tambahkan kebijakan IAM berikut.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "deadline:AssumeFleetRoleForWorker",
        "deadline:UpdateWorker",
        "deadline>DeleteWorker",
        "deadline:UpdateWorkerSchedule",
        "deadline:BatchGetJobEntity",
        "deadline:AssumeQueueRoleForWorker"
      ],
      "Resource": "arn:aws::*:ACCOUNT_ID:*",
      "Condition": {
        "StringEquals": {

```

```

        "aws:PrincipalAccount": "${aws:ResourceAccount}"
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalAccount": "${aws:ResourceAccount}"
        }
      }
    }
  ]
}

```

b. Simpan perubahan Anda.

4. Tambahkan peran IAM untuk digunakan pekerja di armada Anda.

```

aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
FleetWorkerPolicy --policy-document file://fleet-policy.json

```

5. Buat `create-fleet-request.json`.

a. Tambahkan kebijakan IAM berikut, ganti teks *ITALICIZED* dengan nilai CMF Anda.

 Note

Anda dapat menemukan **ROLE_ARN** di `create-cmf-fleet.json`. Untuk itu **OS_FAMILY**, Anda harus memilih salah satu dari `linux`, `macos` atau `windows`.

```
{
  "farmId": "FARM_ID",
  "displayName": "FLEET_NAME",
  "description": "FLEET_DESCRIPTION",
  "roleArn": "ROLE_ARN",
  "minWorkerCount": 0,
  "maxWorkerCount": 10,
  "configuration": {
    "customerManaged": {
      "mode": "NO_SCALING",
      "workerCapabilities": {
        "vCpuCount": {
          "min": 1,
          "max": 4
        },
        "memoryMiB": {
          "min": 1024,
          "max": 4096
        },
        "osFamily": "OS_FAMILY",
        "cpuArchitectureType": "x86_64",
      },
    },
  },
}
```

b. Simpan perubahan Anda.

6. Buat armada Anda.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

Pengaturan dan konfigurasi host pekerja

Host pekerja mengacu pada mesin host yang menjalankan pekerja Deadline Cloud. Bagian ini menjelaskan cara mengatur host pekerja dan mengonfigurasinya untuk kebutuhan spesifik Anda. Setiap host pekerja menjalankan program yang disebut agen pekerja. Agen pekerja bertanggung jawab untuk:

- Mengelola siklus hidup pekerja.
- Sinkronisasi pekerjaan yang ditugaskan, kemajuan dan hasilnya.
- Memantau pekerjaan yang sedang berjalan.
- Meneruskan log ke tujuan yang dikonfigurasi.

Kami menyarankan Anda menggunakan agen pekerja Deadline Cloud yang disediakan. Agen pekerja adalah open source dan kami mendorong permintaan fitur, tetapi Anda juga dapat mengembangkan dan menyesuaikan agar sesuai dengan kebutuhan Anda.

Untuk menyelesaikan tugas di bagian berikut, Anda memerlukan yang berikut:

Linux

- A LinuxInstans Amazon Elastic Compute Cloud (Amazon EC2) berbasis Amazon. Kami merekomendasikan Amazon Linux 2023.
- sudo hak istimewa
- Python 3.9 atau lebih tinggi

Windows

- A WindowsInstans Amazon Elastic Compute Cloud (Amazon EC2) berbasis Amazon. Kami merekomendasikan Windows Server 2022.
- Akses administrator ke host pekerja
- Python 3.9 atau lebih tinggi diinstal untuk semua pengguna

Membuat dan mengkonfigurasi lingkungan virtual Python

Anda dapat membuat lingkungan virtual Python di Linux jika Anda telah menginstal Python 3.9 atau lebih besar dan menempatkannya di file Anda. PATH

Note

Pada Windows, file agen harus diinstal ke direktori paket situs global Python. Lingkungan virtual Python saat ini tidak didukung.

Untuk membuat dan mengaktifkan lingkungan virtual Python

1. Buka terminal sebagai root pengguna (atau gunakan `sudo/su`).
2. Buat dan aktifkan lingkungan virtual Python.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Instal Deadline Agen pekerja Cloud

Setelah Anda mengatur Python Anda dan menciptakan lingkungan virtual di Linux, instal paket Python agen pekerja Deadline Cloud.

Untuk menginstal paket Python agen pekerja

Linux

1. Buka terminal sebagai root pengguna (atau gunakan `sudo/su`).
2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. Buka prompt perintah administrator atau PowerShell terminal.

2. Unduh dan instal paket agen pekerja Deadline Cloud dari PyPI:

```
python -m pip install deadline-cloud-worker-agent
```

Saat Windows host pekerja membutuhkan nama jalur panjang (lebih dari 250 karakter), Anda harus mengaktifkan nama jalur panjang sebagai berikut:

Untuk mengaktifkan jalur panjang untuk Windows tuan rumah pekerja

1. Pastikan bahwa kunci registri jalur panjang diaktifkan. Untuk informasi selengkapnya, lihat [Setelan registri untuk mengaktifkan jalur log](#) di situs web Microsoft.
2. Instal Windows SDK untuk Aplikasi Desktop C++ x86. Untuk informasi selengkapnya, lihat [Windows SDK](#) di Windows Pusat Pengembang.
3. Buka lokasi instalasi Python di lingkungan Anda tempat agen pekerja diinstal. Nilai default-nya C:\Program Files\Python311. Ada file yang dapat dieksekusi bernama. `pythonservice.exe`
4. Buat file baru yang disebut `pythonservice.exe.manifest` di lokasi yang sama. Tambahkan yang berikut ini:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
    version="1.0.0.0"/>
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. Buka prompt perintah dan jalankan perintah berikut di lokasi file manifes yang Anda buat:

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

Anda akan melihat output yang serupa dengan yang berikut:

```
Microsoft (R) Manifest Tool
```

Copyright (c) Microsoft Corporation.
All rights reserved.

Pekerja sekarang dapat mengakses jalur panjang. Untuk membersihkan, hapus `python-service.exe.manifest` file dan hapus instalasi SDK.

Konfigurasi agen pekerja Cloud Deadline

Anda dapat mengonfigurasi pengaturan agen pekerja Deadline Cloud dengan tiga cara. Kami menyarankan Anda menggunakan pengaturan sistem operasi dengan menjalankan `install-deadline-worker` alat.

Agen pekerja tidak mendukung berjalan sebagai pengguna domain di Windows. Untuk menjalankan pekerjaan sebagai pengguna domain, Anda dapat menentukan akun pengguna domain saat mengonfigurasi pengguna antrian untuk menjalankan pekerjaan. Untuk informasi selengkapnya, lihat langkah 7 dalam [antrian Deadline Cloud](#) di Panduan Pengguna Cloud AWS Deadline.

Argumen baris perintah - Anda dapat menentukan argumen saat menjalankan agen pekerja Deadline Cloud dari baris perintah. Beberapa pengaturan konfigurasi tidak tersedia melalui argumen baris perintah. Untuk melihat semua argumen baris perintah yang tersedia, masukkan `deadline-worker-agent --help`.

Variabel lingkungan — Anda dapat mengonfigurasi agen pekerja Deadline Cloud dengan menyetel variabel lingkungan yang dimulai dengan `DEADLINE_WORKER_`. Misalnya, untuk melihat semua argumen baris perintah yang tersedia, Anda dapat menggunakan `export DEADLINE_WORKER_VERBOSE=true` untuk mengatur output agen pekerja ke verbose. Untuk contoh dan informasi selengkapnya, lihat `/etc/amazon/deadline/worker.toml.example` Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` pada Windows.

File konfigurasi - Ketika Anda menginstal agen pekerja, itu membuat file konfigurasi yang terletak di `/etc/amazon/deadline/worker.toml` on Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml` pada Windows. Agen pekerja memuat file konfigurasi ini saat dimulai. Anda dapat menggunakan contoh file konfigurasi (`/etc/amazon/deadline/worker.toml.example` on Linux atau `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` pada Windows) untuk menyesuaikan file konfigurasi agen pekerja default untuk kebutuhan spesifik Anda.

Terakhir, kami sarankan Anda mengaktifkan auto shutdown untuk agen pekerja setelah perangkat lunak Anda digunakan dan berfungsi seperti yang diharapkan. Hal ini memungkinkan armada pekerja

untuk meningkatkan skala ketika diperlukan dan untuk menutup ketika pekerjaan selesai. Penskalaan otomatis membantu memastikan Anda hanya menggunakan sumber daya yang dibutuhkan. Untuk mengaktifkan instance yang dimulai oleh grup penskalaan otomatis untuk dimatikan, Anda harus menambahkan `shutdown_on_stop=true` ke file `worker.toml` konfigurasi.

Untuk mengaktifkan auto shutdown

Sebagai **root** pengguna:

- Instal agen pekerja dengan parameter **--allow-shutdown**.

Linux

Masukkan:

```
/opt/deadline/worker/bin/install-deadline-worker \  
  --farm-id FARM_ID \  
  --fleet-id FLEET_ID \  
  --region REGION \  
  --allow-shutdown
```

Windows

Masukkan:

```
install-deadline-worker ^  
  --farm-id FARM_ID ^  
  --fleet-id FLEET_ID ^  
  --region REGION ^  
  --allow-shutdown
```

Buat pengguna dan grup pekerjaan

Bagian ini menjelaskan hubungan pengguna dan grup yang diperlukan antara pengguna agen dan yang `jobRunAsUser` ditentukan pada antrian Anda.

Agan pekerja Deadline Cloud harus dijalankan sebagai pengguna khusus agen khusus di host. Anda harus mengonfigurasi `jobRunAsUser` properti antrian Deadline Cloud sehingga pekerja akan menjalankan pekerjaan antrian sebagai pengguna dan grup sistem operasi tertentu. Ini berarti

Anda dapat mengontrol izin sistem file bersama yang dimiliki pekerjaan Anda. Ini juga menyediakan sebagai batas keamanan penting antara pekerjaan Anda dan pengguna agen pekerja.

Linux pengguna pekerjaan dan grup

Untuk mengatur pengguna agen pekerja lokal dan `jobRunAsUser`, pastikan Anda memenuhi persyaratan berikut. Jika Anda menggunakan Linux Pluggable Authentication Module (PAM) seperti Active Directory atau LDAP, prosedur Anda mungkin berbeda.

Pengguna agen pekerja dan `jobRunAsUser` grup bersama disetel saat Anda menginstal agen pekerja. Defaultnya adalah `deadline-worker-agent` dan `deadline-job-users`, tetapi Anda dapat mengubahnya saat menginstal agen pekerja.

```
install-deadline-worker \  
  --user AGENT_USER_NAME \  
  --group JOB_USERS_GROUP
```

Perintah harus dijalankan sebagai pengguna root.

- Masing-masing `jobRunAsUser` harus memiliki kelompok utama yang cocok. Membuat pengguna dengan `adduser` perintah biasanya membuat grup utama yang cocok.

```
adduser -r -m jobRunAsUser
```

- Kelompok utama `jobRunAsUser` adalah kelompok sekunder untuk pengguna agen pekerja. Grup bersama memungkinkan agen pekerja untuk membuat file tersedia untuk pekerjaan saat sedang berjalan.

```
usermod -a -G jobRunAsUser deadline-worker-agent
```

- `jobRunAsUser` harus menjadi anggota kelompok kerja bersama.

```
usermod -a -G deadline-job-users jobRunAsUser
```

- Tidak `jobRunAsUser` boleh termasuk dalam kelompok utama pengguna agen pekerja. File sensitif yang ditulis oleh agen pekerja dimiliki oleh kelompok utama agen. Jika `jobRunAsUser` adalah bagian dari grup ini, file agen pekerja dapat diakses oleh pekerjaan yang berjalan pada pekerja.
- Default Wilayah AWS harus sesuai dengan Wilayah peternakan tempat pekerja tersebut berada. Ini harus diterapkan ke semua `jobRunAsUser` akun pada pekerja.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

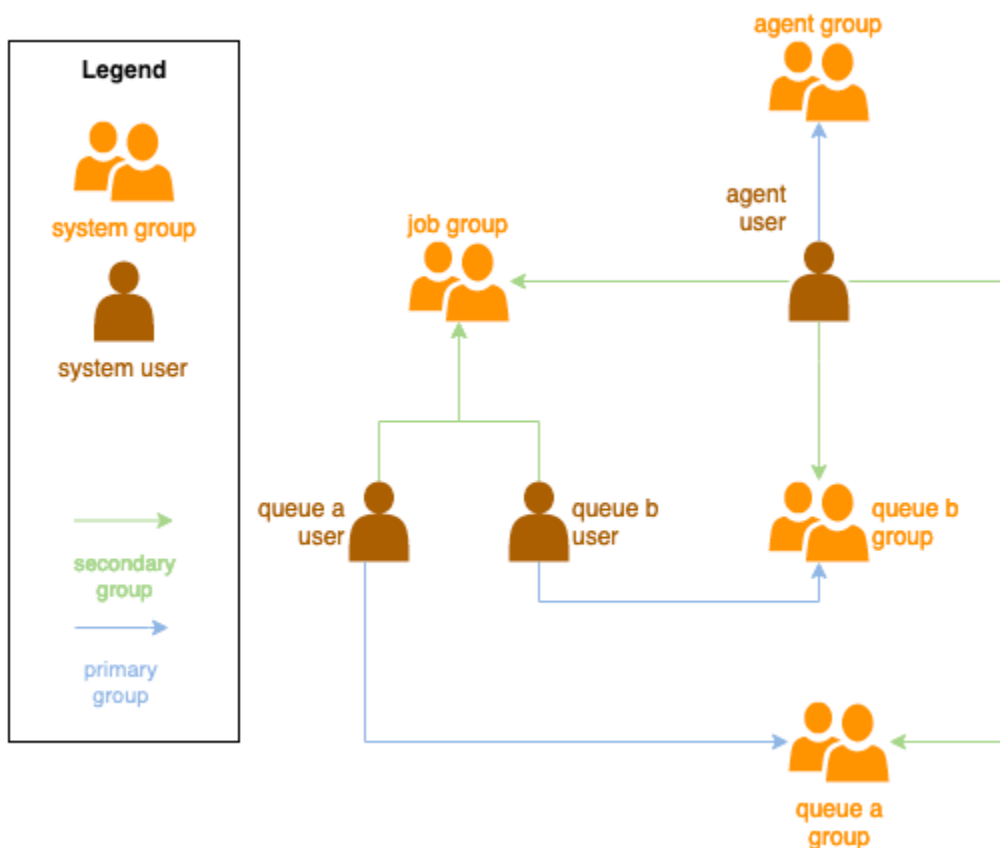
- Pengguna agen pekerja harus dapat menjalankan sudo perintah sebagai *jobRunAsUser*. Jalankan perintah berikut untuk membuka editor untuk membuat aturan sudoers baru:

```
visudo -f /etc/sudoers.d/deadline-worker-job-user
```

Tambahkan yang berikut ini ke file:

```
# Allows the Deadline Cloud worker agent OS user to run commands
# as the queue OS user without requiring a password.
deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
```

Diagram berikut menggambarkan hubungan antara pengguna agen dan *jobRunAsUser* pengguna dan grup untuk antrian yang terkait dengan armada.



Windows pengguna

Untuk menggunakan Windows pengguna sebagai `jobRunAsUser`, itu harus memenuhi persyaratan berikut:

- Semua `jobRunAsUser` pengguna antrian harus ada.
- Kata sandi mereka harus sesuai dengan nilai rahasia yang ditentukan dalam `JobRunAsUser` bidang antrian mereka. Untuk petunjuknya, lihat langkah 7 dalam [antrian Deadline Cloud di Panduan Pengguna](#) Cloud AWS Deadline.
- Agen-pengguna harus dapat masuk sebagai pengguna tersebut.

Kelola akses ke rahasia pengguna Windows pekerjaan

Saat Anda mengonfigurasi antrian dengan a Windows `jobRunAsUser`, Anda harus menentukan AWS rahasia Secrets Manager. Nilai rahasia ini diharapkan menjadi objek yang dikodekan JSON dari bentuk:

```
{  
  "password": "JOB_USER_PASSWORD"  
}
```

Agar Pekerja menjalankan pekerjaan sesuai antrian yang dikonfigurasi `jobRunAsUser`, peran IAM armada harus memiliki izin untuk mendapatkan nilai rahasia. Jika rahasia dienkripsi menggunakan kunci KMS yang dikelola pelanggan, maka peran IAM armada juga harus memiliki izin untuk mendekripsi menggunakan kunci KMS.

Sangat disarankan untuk mengikuti prinsip hak istimewa paling sedikit untuk rahasia ini. Ini berarti bahwa akses untuk mengambil nilai rahasia dari antrian `jobRunAsUser` → `windows` → `passwordArn` harus:

- diberikan untuk peran armada ketika asosiasi antrian-armada dibuat antara armada dan antrian
- dicabut dari peran armada ketika asosiasi antrian-armada dihapus antara armada dan antrian

Selanjutnya, AWS rahasia Secrets Manager yang berisi `jobRunAsUser` kata sandi harus dihapus ketika tidak lagi digunakan.

Berikan akses ke rahasia kata sandi

Armada Cloud deadline memerlukan akses ke `jobRunAsUser` kata sandi yang disimpan dalam rahasia kata sandi antrian saat antrian dan armada terkait. Sebaiknya gunakan kebijakan sumber daya AWS Secrets Manager untuk memberikan akses ke peran armada. Jika Anda benar-benar mematuhi pedoman ini, lebih mudah untuk menentukan peran armada mana yang memiliki akses ke rahasia.

Untuk memberikan akses ke rahasia

1. Buka konsol AWS Secret Manager ke rahasia.
2. Di bagian “Izin sumber daya”, tambahkan pernyataan kebijakan formulir:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    //...
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "FLEET_ROLE_ARN"
      },
      "Action" : "secretsmanager:GetSecretValue",
      "Resource" : "*"
    }
    //...
  ]
}
```

Cabut akses ke rahasia kata sandi

Ketika armada tidak lagi memerlukan akses ke antrian, hapus akses ke rahasia kata sandi untuk antrian `jobRunAsUser`. Sebaiknya gunakan kebijakan sumber daya AWS Secrets Manager untuk memberikan akses ke peran armada. Jika Anda benar-benar mematuhi pedoman ini, lebih mudah untuk menentukan peran armada mana yang memiliki akses ke rahasia.

Untuk mencabut akses ke rahasia

1. Buka konsol AWS Secret Manager ke rahasia.

2. Di bagian Izin sumber daya, hapus pernyataan kebijakan formulir:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    //...
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "FLEET_ROLE_ARN"
      },
      "Action" : "secretsmanager:GetSecretValue",
      "Resource" : "*"
    }
    //...
  ]
}
```

Instal dan konfigurasi perangkat lunak yang diperlukan untuk pekerjaan

Setelah menyiapkan agen pekerja Deadline Cloud, Anda dapat menyiapkan host pekerja dengan perangkat lunak apa pun yang diperlukan untuk menjalankan pekerjaan.

Saat Anda mengirimkan pekerjaan ke antrian dengan yang terkait `jobRunAsUser`, pekerjaan berjalan sebagai pengguna tersebut. Ketika pekerjaan dikirimkan dengan perintah yang bukan jalur absolut, perintah itu harus ditemukan di PATH pengguna itu.

Di Linux, Anda dapat menentukan PATH untuk pengguna di salah satu dari berikut ini:

- mereka `~/.bashrc` atau `~/.bash_profile`
- file konfigurasi sistem seperti `/etc/profile.d/*` dan `/etc/profile`
- skrip startup shell: `/etc/bashrc`.

Di Windows, Anda dapat menentukan PATH untuk pengguna di salah satu dari berikut ini:

- variabel lingkungan khusus pengguna mereka
- variabel lingkungan seluruh sistem

Instal adaptor alat pembuatan konten digital

Deadline Cloud menyediakan OpenJobDescription adaptor untuk menggunakan aplikasi pembuatan konten digital (DCC) populer. Untuk menggunakan adaptor ini dalam armada yang dikelola pelanggan, Anda harus menginstal perangkat lunak DCC dan adaptor aplikasi. Kemudian, pastikan program yang dapat dieksekusi perangkat lunak tersedia di jalur pencarian sistem (misalnya, dalam variabel PATH lingkungan).

Untuk memasang adaptor DCC pada armada yang dikelola pelanggan

1. Buka terminal.
 - a. Di Linux, buka terminal sebagai root pengguna (atau gunakan `sudo/su`)
 - b. WindowsAktif, buka prompt perintah administrator atau PowerShell terminal.
2. Instal paket adaptor Deadline Cloud.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-  
cloud-for-blender deadline-cloud-for-3ds-max
```

Mengkonfigurasi kredensial AWS

Fase awal siklus hidup pekerja adalah bootstrap. Pada fase ini, perangkat lunak agen pekerja menciptakan pekerja di armada Anda, dan memperoleh AWS kredensial dari peran armada Anda untuk operasi lebih lanjut.

AWS credentials for Amazon EC2

Untuk membuat peran IAM untuk Amazon EC2 dengan izin host pekerja Deadline Cloud

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi, pilih Peran di panel navigasi, lalu pilih Buat peran.
3. Pilih AWS layanan.
4. Pilih EC2 sebagai Layanan atau kasus penggunaan, lalu pilih Berikutnya.
5. Untuk memberikan izin yang diperlukan, lampirkan kebijakan `AWSDeadlineCloud-WorkerHost` AWS terkelola.

On-premises AWS credentials

Pekerja lokal Anda menggunakan kredensial untuk mengakses Deadline Cloud. Untuk akses yang paling aman, sebaiknya gunakan IAM Roles Anywhere untuk mengautentikasi pekerja Anda. Untuk informasi selengkapnya, lihat [Peran IAM Di Mana Saja](#).

Untuk pengujian, Anda dapat menggunakan kunci akses pengguna IAM untuk AWS kredensi. Kami menyarankan Anda menetapkan kedaluwarsa untuk pengguna IAM dengan menyertakan kebijakan inline yang membatasi.

Important

Perhatikan peringatan berikut:

- JANGAN gunakan kredensi root akun Anda untuk mengakses AWS sumber daya. Kredensi ini menyediakan akses akun yang tidak terbatas dan sulit dicabut.
- JANGAN menaruh kunci akses literal atau informasi kredensi dalam file aplikasi Anda. Jika Anda melakukannya, Anda membuat risiko secara tidak sengaja mengekspos kredensialnya jika, misalnya, Anda mengunggah proyek ke repositori publik.
- JANGAN sertakan file yang berisi kredensial di area proyek Anda.
- Amankan kunci akses Anda. Jangan berikan kunci akses Anda kepada pihak yang tidak berwenang, bahkan untuk membantu [menemukan pengenalan akun Anda](#). Dengan melakukan tindakan ini, Anda mungkin memberi seseorang akses permanen ke akun Anda.
- Ketahuilah bahwa kredensial apa pun yang disimpan dalam file AWS kredensial bersama disimpan dalam teks biasa.

Untuk detail selengkapnya, lihat [Praktik terbaik untuk mengelola kunci AWS akses di Referensi AWS Umum](#).

Membuat pengguna IAM

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi, pilih Pengguna dan pilih Buat pengguna.
3. Beri nama pengguna. Kosongkan kotak centang untuk Menyediakan akses pengguna ke AWS Management Console, lalu pilih Berikutnya.

4. Pilih Lampirkan kebijakan secara langsung.
5. Dari daftar kebijakan izin, pilih AWSDeadlineCloud-WorkerHostkebijakan, lalu pilih Berikutnya.
6. Tinjau detail pengguna dan kemudian pilih Buat pengguna.

Batasi akses pengguna ke jendela waktu terbatas

Kunci akses pengguna IAM apa pun yang Anda buat adalah kredensi jangka panjang. Untuk memastikan bahwa kredensial ini kedaluwarsa jika salah penanganan, Anda dapat membuat kredensial ini terikat waktu dengan membuat kebijakan inline yang menentukan tanggal setelah kunci tidak lagi valid.

1. Buka pengguna IAM yang baru saja Anda buat. Di tab Izin, pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.
2. Di editor JSON, tentukan izin berikut. Untuk menggunakan kebijakan ini, ganti nilai `aws:CurrentTime` stempel waktu dalam kebijakan contoh dengan waktu dan tanggal Anda sendiri.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Buat kunci akses

1. Pada halaman detail pengguna, pilih tab Security credentials. Pada bagian Access key, pilih Buat access key.
2. Tunjukkan bahwa Anda ingin menggunakan kunci untuk Lainnya, lalu pilih Berikutnya, lalu pilih Buat kunci akses.
3. Pada halaman Ambil kunci akses, pilih Tampilkan untuk mengungkapkan nilai kunci akses rahasia pengguna Anda. Anda dapat menyalin kredensialnya atau mengunduh file.csv.

Simpan kunci akses pengguna

- Menyimpan kunci akses pengguna dalam file AWS kredensial pengguna agen pada sistem host pekerja:
 - PadaLinux, file tersebut terletak di `~/.aws/credentials`
 - PadaWindows, file tersebut terletak di `%USERPROFILE\.aws\credentials`

Ganti kunci berikut:

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

Important

Ketika Anda tidak lagi membutuhkan pengguna IAM ini, kami sarankan Anda menghapusnya agar selaras dengan praktik [terbaik AWS keamanan](#). Kami menyarankan Anda meminta pengguna manusia Anda untuk menggunakan kredensi sementara [AWS IAM Identity Centers](#) saat mengakses. AWS

Konfigurasi jaringan untuk mengizinkan koneksi titik akhir AWS

Deadline Cloud memerlukan konektivitas yang aman ke berbagai titik akhir AWS layanan untuk pengoperasian yang tepat. Untuk menggunakan Deadline Cloud, Anda harus memastikan bahwa

lingkungan jaringan Anda memungkinkan pekerja Deadline Cloud Anda untuk terhubung ke titik akhir ini.

Jika Anda memiliki pengaturan firewall jaringan yang memblokir koneksi keluar, Anda mungkin perlu menambahkan pengecualian firewall untuk titik akhir tertentu. Untuk Deadline Cloud, Anda harus menambahkan pengecualian untuk layanan berikut:

- [Titik akhir Cloud Batas Waktu](#)
- [Titik akhir Amazon CloudWatch Logs](#)
- [Titik akhir Layanan Penyimpanan Sederhana Amazon](#)

Jika pekerjaan Anda menggunakan AWS layanan lain, Anda mungkin perlu menambahkan pengecualian untuk layanan tersebut juga. Anda dapat menemukan titik akhir ini di [titik akhir Layanan dan bagian kuota dari panduan Referensi](#) Umum AWS. Setelah Anda mengidentifikasi titik akhir yang diperlukan, buat aturan keluar di firewall Anda untuk mengizinkan lalu lintas ke titik akhir tertentu ini.

Memastikan bahwa titik akhir ini dapat diakses diperlukan untuk pengoperasian yang benar. Selain itu, pertimbangkan untuk menerapkan langkah-langkah keamanan yang tepat, seperti menggunakan virtual private cloud (VPCs), grup keamanan, dan daftar kontrol akses jaringan (ACLs) untuk menjaga lingkungan yang aman sambil memungkinkan lalu lintas Deadline Cloud yang diperlukan.

Uji konfigurasi host pekerja Anda

Setelah Anda menginstal agen pekerja, menginstal perangkat lunak yang diperlukan untuk memproses pekerjaan Anda, dan mengonfigurasi AWS kredensi untuk agen pekerja, Anda harus menguji bahwa instalasi dapat memproses pekerjaan Anda sebelum membuat AMI untuk armada Anda. Anda harus menguji yang berikut:

- Agen pekerja Deadline Cloud dikonfigurasi dengan benar untuk dijalankan sebagai layanan sistem.
- Bahwa pekerja melakukan polling antrian terkait untuk bekerja.
- Bahwa pekerja berhasil memproses pekerjaan yang dikirim ke antrian yang terkait dengan armada.

Setelah Anda menguji konfigurasi dan berhasil memproses pekerjaan perwakilan, Anda dapat menggunakan pekerja yang dikonfigurasi untuk membuat AMI untuk EC2 pekerja Amazon, atau sebagai model untuk pekerja lokal Anda.

 Note

Jika Anda menguji konfigurasi host pekerja dari armada penskalaan otomatis, Anda mungkin mengalami kesulitan menguji pekerja Anda dalam situasi berikut:

- Jika tidak ada pekerjaan dalam antrian, Deadline Cloud menghentikan agen pekerja segera setelah pekerja mulai.
- Jika agen pekerja dikonfigurasi untuk mematikan host saat berhenti, agen mematikan mesin jika tidak ada pekerjaan dalam antrian.

Untuk menghindari masalah ini, gunakan armada pementasan yang tidak menskalakan otomatis untuk mengonfigurasi dan menguji pekerja Anda. Setelah menguji host pekerja, pastikan untuk mengatur ID armada yang benar sebelum memanggang AML.

Untuk menguji konfigurasi host pekerja Anda

1. Jalankan agen pekerja dengan memulai layanan sistem operasi.

Linux

Dari shell root jalankan perintah berikut:

```
systemctl start deadline-worker
```

Windows

Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

```
sc.exe start DeadlineWorker
```

2. Pantau pekerja untuk memastikannya dimulai dan polling untuk pekerjaan.

Linux

Dari shell root jalankan perintah berikut:

```
systemctl status deadline-worker
```

Perintah harus mengembalikan respons seperti:

```
Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
```

Jika respons tidak terlihat seperti itu, periksa file log menggunakan perintah berikut:

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

```
sc.exe query DeadlineWorker
```

Perintah harus mengembalikan respons seperti:

```
STATE      : 4 RUNNING
```

Jika respons tidak berisi RUNNING, periksa file log pekerja. Buka dan administrator PowerShell prompt dan jalankan perintah berikut:

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-agent.log
```

3. Kirim pekerjaan ke antrian yang terkait dengan armada Anda. Pekerjaan harus mewakili pekerjaan yang diproses armada.
4. Pantau kemajuan pekerjaan [menggunakan monitor Deadline Cloud](#) atau CLI. Jika pekerjaan gagal, periksa sesi dan log pekerja.
5. Perbarui konfigurasi host pekerja sesuai kebutuhan hingga pekerjaan selesai dengan sukses.
6. Ketika pekerjaan uji berhasil, Anda dapat menghentikan pekerja:

Linux

Dari shell root jalankan perintah berikut:

```
systemctl stop deadline-worker
```

Windows

Dari prompt perintah administrator atau PowerShell terminal, masukkan perintah berikut:

```
sc.exe stop DeadlineWorker
```

Buat sebuah Amazon Machine Image

Untuk membuat Amazon Machine Image (AMI) untuk digunakan dalam armada yang dikelola pelanggan (CMF EC2) Amazon Elastic Compute Cloud (Amazon), selesaikan tugas di bagian ini. Anda harus membuat EC2 instance Amazon sebelum melanjutkan. Untuk informasi selengkapnya, lihat [Meluncurkan instans Anda](#) di Panduan EC2 Pengguna Amazon untuk Instans Linux.

Important

Membuat AMI membuat snapshot dari volume terlampir EC2 instans Amazon. Perangkat lunak apa pun yang diinstal pada instance tetap ada sehingga instance, yang digunakan kembali saat Anda meluncurkan instance dari AMI. Kami merekomendasikan untuk mengadopsi strategi patching dan secara teratur memperbarui yang baru AMI dengan perangkat lunak yang diperbarui sebelum mendaftar ke armada Anda.

Siapkan EC2 instans Amazon

Sebelum Anda membangun sebuah AMI, Anda harus menghapus status pekerja. Negara pekerja tetap ada di antara peluncuran agen pekerja. Jika keadaan ini berlanjut ke AMI, maka semua instance yang diluncurkan darinya akan berbagi status yang sama.

Kami juga menyarankan Anda menghapus file log yang ada. File log dapat tetap berada di EC2 instans Amazon saat Anda menyiapkan AMI. Menghapus file-file ini meminimalkan kebingungan saat mendiagnosis kemungkinan masalah dalam armada pekerja yang menggunakan AMI.

Anda juga harus mengaktifkan layanan sistem agen pekerja sehingga agen pekerja Deadline Cloud diluncurkan saat Amazon EC2 dimulai.

Terakhir, kami sarankan Anda mengaktifkan auto shutdown agen pekerja. Hal ini memungkinkan armada pekerja untuk meningkatkan skala saat dibutuhkan dan dimatikan saat pekerjaan rendering

selesai. Penskalaan otomatis ini membantu memastikan Anda hanya menggunakan sumber daya sesuai kebutuhan.

Untuk menyiapkan EC2 instance Amazon

1. Buka EC2 konsol Amazon.
2. Luncurkan EC2 instans Amazon. Untuk informasi selengkapnya, lihat [Meluncurkan instans Anda](#).
3. Siapkan host untuk terhubung ke penyedia identitas Anda (iDP), lalu pasang sistem file bersama yang dibutuhkannya.
4. Ikuti tutorial untuk [Instal Deadline Agen pekerja Cloud](#), kemudian [Konfigurasi agen pekerja](#), dan [Buat pengguna dan grup pekerjaan](#).
5. Jika Anda sedang mempersiapkan AMI berdasarkan Amazon Linux 2023 untuk menjalankan perangkat lunak yang kompatibel dengan Platform Referensi VFX, Anda perlu memperbarui beberapa persyaratan. Untuk selengkapnya, lihat [Kompatibilitas Platform Referensi VFX](#) di Panduan Pengguna Cloud AWS Deadline.
6. Buka terminal.
 - a. Di Linux, buka terminal sebagai root pengguna (atau gunakan `sudo/su`)
 - b. Pada Windows, buka prompt perintah administrator atau PowerShell terminal.
7. Pastikan layanan pekerja tidak berjalan dan dikonfigurasi untuk memulai saat boot:
 - a. Di Linux, jalankan

```
systemctl stop deadline-worker  
systemctl enable deadline-worker
```

- b. Pada Windows, lari

```
sc.exe stop DeadlineWorker  
sc.exe config DeadlineWorker start= auto
```

8. Hapus status pekerja.

- a. Di Linux, jalankan

```
rm -rf /var/lib/deadline/*
```

- b. Pada Windows, lari

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Hapus file log.

- a. Di Linux, jalankan

```
rm -rf /var/log/amazon/deadline/*
```

- b. Pada Windows, lari

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. Pada Windows, disarankan untuk menjalankan aplikasi Amazon EC2 Launch Settings yang ditemukan di menu Start untuk menyelesaikan persiapan host akhir dan shutdown instance.

 Note

Anda HARUS memilih Shutdown tanpa Sysprep dan tidak pernah memilih Shutdown dengan Sysprep. Mematikan dengan Sysprep akan menyebabkan semua pengguna lokal menjadi tidak dapat digunakan. Untuk informasi selengkapnya, lihat [bagian Sebelum Anda Memulai topik Buat AMI kustom dari Panduan Pengguna untuk Instans Windows](#).

Membangun AMI

Untuk membangun AMI

1. Buka EC2 konsol Amazon.
2. Pilih Instans di panel navigasi, lalu pilih instans Anda.
3. Pilih status Instance, lalu Stop instance.
4. Setelah instance Dihentikan, pilih Tindakan.
5. Pilih Gambar dan template, lalu Buat gambar.
6. Masukkan nama Gambar.
7. (Opsional) Masukkan deskripsi untuk gambar Anda.
8. Pilih Buat citra.

Buat infrastruktur armada dengan grup Amazon EC2 Auto Scaling

Bagian ini menjelaskan cara membuat armada EC2 Auto Scaling Amazon.

Gunakan template AWS CloudFormation YAMB di bawah ini untuk membuat grup Amazon EC2 Auto Scaling (Auto Scaling), Amazon Virtual Private Cloud (Amazon VPC) dengan dua subnet, profil instans, dan peran akses instans. Ini diperlukan untuk meluncurkan instance menggunakan Auto Scaling di subnet.

Anda harus meninjau dan memperbarui daftar jenis instance agar sesuai dengan kebutuhan rendering Anda.

Untuk penjelasan lengkap tentang resource dan parameter yang digunakan dalam template CloudFormation YAMB, lihat [referensi tipe resource Deadline Cloud](#) di AWS CloudFormation Panduan Pengguna.

Untuk membuat armada EC2 Auto Scaling Amazon

1. Gunakan contoh berikut untuk membuat CloudFormation template yang mendefinisikan FarmID, FleetID, dan AMIID parameter. Simpan template ke .YAML file di komputer lokal Anda.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIID:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
```

```

GroupDescription: !Join
- ' '
- - Security group created for Deadline Cloud workers in the fleet
- !Ref FleetId
GroupName: !Join
- ''
- - deadlineWorkerSecurityGroup-
- !Ref FleetId
SecurityGroupEgress:
- CidrIp: 0.0.0.0/0
  IpProtocol: '-1'
SecurityGroupIngress: []
VpcId: !Ref deadlineVPC
deadlineIGW:
  Type: 'AWS::EC2::InternetGateway'
  Properties: {}
deadlineVPCGatewayAttachment:
  Type: 'AWS::EC2::VPCGatewayAttachment'
  Properties:
    VpcId: !Ref deadlineVPC
    InternetGatewayId: !Ref deadlineIGW
deadlinePublicRouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref deadlineVPC
deadlinePublicRoute:
  Type: 'AWS::EC2::Route'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref deadlineIGW
  DependsOn:
    - deadlineIGW
    - deadlineVPCGatewayAttachment
deadlinePublicSubnet0:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.16.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
      - a
deadlineSubnetRouteTableAssociation0:

```

```

    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
deadlinePublicSubnet1:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref deadlineVPC
    CidrBlock: 100.100.20.0/22
    AvailabilityZone: !Join
      - ''
      - - !Ref 'AWS::Region'
        - c
deadlineSubnetRouteTableAssociation1:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    RouteTableId: !Ref deadlinePublicRouteTable
    SubnetId: !Ref deadlinePublicSubnet1
deadlineInstanceAccessAccessRole:
  Type: 'AWS::IAM::Role'
  Properties:
    RoleName: !Join
      - '-'
      - - deadline
        - InstanceAccess
        - !Ref FleetId
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    Path: /
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
      - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
      - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
deadlineInstanceProfile:
  Type: 'AWS::IAM::InstanceProfile'
  Properties:
    Path: /
    Roles:
      - !Ref deadlineInstanceAccessAccessRole

```

```

deadlineLaunchTemplate:
  Type: 'AWS::EC2::LaunchTemplate'
  Properties:
    LaunchTemplateName: !Join
      - ''
      - - deadline-LT-
        - !Ref FleetId
    LaunchTemplateData:
      NetworkInterfaces:
        - DeviceIndex: 0
          AssociatePublicIpAddress: true
          Groups:
            - !Ref deadlineWorkerSecurityGroup
          DeleteOnTermination: true
      ImageId: !Ref AMIID
      InstanceInitiatedShutdownBehavior: terminate
      IamInstanceProfile:
        Arn: !GetAtt
          - deadlineInstanceProfile
          - Arn
      MetadataOptions:
        HttpTokens: required
        HttpEndpoint: enabled

deadlineAutoScalingGroup:
  Type: 'AWS::AutoScaling::AutoScalingGroup'
  Properties:
    AutoScalingGroupName: !Join
      - ''
      - - deadline-ASG-autoscalable-
        - !Ref FleetId
    MinSize: 0
    MaxSize: 10
    VPCZoneIdentifier:
      - !Ref deadlinePublicSubnet0
      - !Ref deadlinePublicSubnet1
    NewInstancesProtectedFromScaleIn: true
    MixedInstancesPolicy:
      InstancesDistribution:
        OnDemandBaseCapacity: 0
        OnDemandPercentageAboveBaseCapacity: 0
        SpotAllocationStrategy: capacity-optimized
        OnDemandAllocationStrategy: lowest-price
    LaunchTemplate:

```

```

LaunchTemplateSpecification:
  LaunchTemplateId: !Ref deadlineLaunchTemplate
  Version: !GetAtt
    - deadlineLaunchTemplate
    - LatestVersionNumber
Overrides:
  - InstanceType: m5.large
  - InstanceType: m5d.large
  - InstanceType: m5a.large
  - InstanceType: m5ad.large
  - InstanceType: m5n.large
  - InstanceType: m5dn.large
  - InstanceType: m4.large
  - InstanceType: m3.large
  - InstanceType: r5.large
  - InstanceType: r5d.large
  - InstanceType: r5a.large
  - InstanceType: r5ad.large
  - InstanceType: r5n.large
  - InstanceType: r5dn.large
  - InstanceType: r4.large
MetricsCollection:
  - Granularity: 1Minute
  Metrics:
    - GroupMinSize
    - GroupMaxSize
    - GroupDesiredCapacity
    - GroupInServiceInstances
    - GroupTotalInstances
    - GroupInServiceCapacity
    - GroupTotalCapacity

```

2. Buka AWS CloudFormation konsol di <https://console.aws.amazon.com/cloudformation>.

Gunakan AWS CloudFormation konsol untuk membuat tumpukan menggunakan instruksi untuk mengunggah file template yang Anda buat. Untuk informasi selengkapnya, lihat [Membuat tumpukan di AWS CloudFormation konsol](#) di Panduan AWS CloudFormation Pengguna.

Note

- Kredensial dari peran IAM yang dilampirkan ke EC2 instans Amazon pekerja Anda tersedia untuk semua proses yang berjalan pada pekerja tersebut, yang mencakup

pekerjaan. Pekerja harus memiliki hak istimewa paling sedikit untuk beroperasi:

`deadline:CreateWorker` dan `deadline:AssumeFleetRoleForWorker`.

- Agen pekerja memperoleh kredensial untuk peran antrian dan mengonfigurasinya untuk digunakan dengan menjalankan pekerjaan. Peran profil EC2 instans Amazon tidak boleh menyertakan izin yang diperlukan oleh pekerjaan Anda.

Skalakan otomatis EC2 armada Amazon Anda dengan fitur rekomendasi skala Deadline Cloud

Deadline Cloud memanfaatkan grup Amazon Auto EC2 Scaling (Auto Scaling) untuk menskalakan armada yang dikelola pelanggan EC2 Amazon (CMF) secara otomatis. Anda perlu mengonfigurasi mode armada serta menerapkan infrastruktur yang diperlukan di akun Anda untuk membuat skala otomatis armada Anda. Infrastruktur yang Anda gunakan akan berfungsi untuk semua armada, jadi Anda hanya perlu mengaturnya sekali.

Alur kerja dasarnya adalah: Anda mengonfigurasi mode armada Anda ke skala otomatis, lalu Deadline Cloud akan mengirimkan EventBridge acara untuk armada tersebut setiap kali ukuran armada yang direkomendasikan berubah (satu peristiwa berisi id armada, ukuran armada yang direkomendasikan, dan metadata lainnya). Anda akan memiliki EventBridge aturan untuk memfilter acara yang relevan dan meminta Lambda untuk mengkonsumsinya. Lambda akan berintegrasi dengan Amazon Auto EC2 AutoScalingGroup Scaling untuk menskalakan armada Amazon EC2 secara otomatis.

Atur mode armada ke **EVENT_BASED_AUTO_SCALING**

Konfigurasikan mode armada Anda ke **EVENT_BASED_AUTO_SCALING**. Anda dapat menggunakan konsol untuk melakukan ini, atau menggunakan AWS CLI untuk langsung memanggil `CreateFleet` atau `UpdateFleet` API. Setelah mode dikonfigurasi, Deadline Cloud mulai mengirimkan EventBridge peristiwa setiap kali ukuran armada yang direkomendasikan berubah.

- Contoh `UpdateFleet` perintah:

```
aws deadline update-fleet \  
  --farm-id FARM_ID \  
  --fleet-id FLEET_ID \  
  --configuration file://configuration.json
```

- Contoh `CreateFleet` perintah:

```
aws deadline create-fleet \  
  --farm-id FARM_ID \  
  --display-name "Fleet name" \  
  --max-worker-count 10 \  
  --configuration file://configuration.json
```

Berikut ini adalah contoh yang `configuration.json` digunakan dalam perintah CLI di atas (`--configuration file://configuration.json`).

- Untuk mengaktifkan Auto Scaling pada armada Anda, Anda harus mengatur mode ke. `EVENT_BASED_AUTO_SCALING`
- `workerCapabilities` ini adalah nilai default yang ditetapkan ke CMF saat Anda membuatnya. Anda dapat mengubah nilai-nilai ini jika Anda perlu meningkatkan sumber daya yang tersedia untuk CMF Anda.

Setelah Anda mengonfigurasi mode armada, Deadline Cloud mulai memancarkan acara rekomendasi ukuran armada untuk armada tersebut.

```
{  
  "customerManaged": {  
    "mode": "EVENT_BASED_AUTO_SCALING",  
    "workerCapabilities": {  
      "vCpuCount": {  
        "min": 1,  
        "max": 4  
      },  
      "memoryMiB": {  
        "min": 1024,  
        "max": 4096  
      },  
      "osFamily": "linux",  
      "cpuArchitectureType": "x86_64"  
    }  
  }  
}
```

Terapkan tumpukan Auto Scaling menggunakan template AWS CloudFormation

Anda dapat mengatur EventBridge aturan untuk memfilter peristiwa, Lambda untuk mengkonsumsi peristiwa dan mengontrol Auto Scaling, dan antrian SQS untuk menyimpan peristiwa yang belum diproses. Gunakan AWS CloudFormation template berikut untuk menyebarkan semuanya dalam tumpukan. Setelah Anda berhasil menyebarkan sumber daya, Anda dapat mengirimkan pekerjaan dan armada akan meningkat secara otomatis.

Resources:

AutoScalingLambda:

Type: 'AWS::Lambda::Function'

Properties:

Code:

ZipFile: |-

"""

This lambda is configured to handle "Fleet Size Recommendation Change" messages. It will handle all such events, and requires that the ASG is named based on the fleet id. It will scale up/down the fleet based on the recommended fleet size in the message.

Example EventBridge message:

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "Fleet Size Recommendation Change",
  "source": "aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "fleetId": "fleet-12345678900000000000000000000000",
    "oldFleetSize": 1,
    "newFleetSize": 5,
  }
}
```

"""

```
import json
import boto3
import logging
```

```

logger = logging.getLogger()
logger.setLevel(logging.INFO)

auto_scaling_client = boto3.client("autoscaling")

def lambda_handler(event, context):
    logger.info(event)
    event_detail = event["detail"]
    fleet_id = event_detail["fleetId"]
    desired_capacity = event_detail["newFleetSize"]

    asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
    auto_scaling_client.set_desired_capacity(
        AutoScalingGroupName=asg_name,
        DesiredCapacity=desired_capacity,
        HonorCooldown=False,
    )

    return {
        'statusCode': 200,
        'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')}
    }

Handler: index.lambda_handler
Role: !GetAtt
  - AutoScalingLambdaServiceRole
  - Arn
Runtime: python3.11
DependsOn:
  - AutoScalingLambdaServiceRoleDefaultPolicy
  - AutoScalingLambdaServiceRole
AutoScalingEventRule:
Type: 'AWS::Events::Rule'
Properties:
  EventPattern:
    source:
      - aws.deadline
    detail-type:
      - Fleet Size Recommendation Change
  State: ENABLED
  Targets:
    - Arn: !GetAtt
      - AutoScalingLambda
    - Arn

```

```

    DeadLetterConfig:
      Arn: !GetAtt
        - UnprocessedAutoScalingEventQueue
        - Arn
      Id: Target0
      RetryPolicy:
        MaximumRetryAttempts: 15
AutoScalingEventRuleTargetPermission:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !GetAtt
      - AutoScalingLambda
      - Arn
    Principal: events.amazonaws.com
    SourceArn: !GetAtt
      - AutoScalingEventRule
      - Arn
AutoScalingLambdaServiceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: 'sts:AssumeRole'
          Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
      Version: 2012-10-17
    ManagedPolicyArns:
      - !Join
        - ''
        - - 'arn:'
          - !Ref 'AWS::Partition'
          - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
AutoScalingLambdaServiceRoleDefaultPolicy:
  Type: 'AWS::IAM::Policy'
  Properties:
    PolicyDocument:
      Statement:
        - Action: 'autoscaling:SetDesiredCapacity'
          Effect: Allow
          Resource: '*'
      Version: 2012-10-17
    PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy

```

```

Roles:
  - !Ref AutoScalingLambdaServiceRole
UnprocessedAutoScalingEventQueue:
  Type: 'AWS::SQS::Queue'
  Properties:
    QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
UnprocessedAutoScalingEventQueuePolicy:
  Type: 'AWS::SQS::QueuePolicy'
  Properties:
    PolicyDocument:
      Statement:
        - Action: 'sqs:SendMessage'
          Condition:
            ArnEquals:
              'aws:SourceArn': !GetAtt
                - AutoScalingEventRule
                - Arn
          Effect: Allow
          Principal:
            Service: events.amazonaws.com
          Resource: !GetAtt
            - UnprocessedAutoScalingEventQueue
            - Arn
      Version: 2012-10-17
Queues:
  - !Ref UnprocessedAutoScalingEventQueue

```

Lakukan pemeriksaan kesehatan armada

Setelah membuat armada Anda, Anda harus membuat pemeriksaan kesehatan khusus untuk memastikan armada Anda tetap sehat dan bebas dari keadaan macet untuk membantu mencegah biaya yang tidak perlu. Lihat [Menerapkan pemeriksaan kesehatan armada Cloud Deadline](#) di GitHub. Ini dapat menurunkan risiko perubahan yang tidak disengaja pada Anda Amazon Machine Image, meluncurkan template, atau konfigurasi jaringan yang berjalan tidak terdeteksi.

Konfigurasikan dan gunakan armada yang dikelola layanan Deadline Cloud

Service-managed fleet (SMF) adalah kumpulan pekerja yang dikelola oleh Deadline Cloud. SMF menghilangkan kebutuhan untuk mengelola penskalaan armada untuk permintaan pemrosesan atau mengurangi ukuran armada setelah tugas selesai.

Ketika SMF dikaitkan dengan antrian menggunakan lingkungan antrian Conda default, Deadline Cloud mengonfigurasi pekerja di armada dengan paket perangkat lunak yang sesuai. Untuk aplikasi mitra yang didukung, lihat [Lingkungan antrian Conda default di Panduan Pengguna Cloud AWS Batas Waktu](#).

Dalam kebanyakan kasus, Anda tidak perlu mengubah SMF untuk memproses beban kerja Anda. Namun, beberapa situasi mungkin mengharuskan Anda membuat perubahan pada armada Anda. Ini termasuk:

- Menjalankan skrip yang memerlukan izin tinggi untuk menginstal perangkat lunak atau Docker wadah.

Topik

- [Jalankan skrip sebagai administrator untuk mengonfigurasi pekerja](#)
- [Hubungkan sumber daya VPC ke SMF Anda dengan titik akhir sumber daya VPC](#)

Jalankan skrip sebagai administrator untuk mengonfigurasi pekerja

Skrip konfigurasi host armada khusus memungkinkan Anda melakukan tugas administratif, seperti instalasi perangkat lunak, pada pekerja armada yang dikelola layanan Anda. Skrip ini berjalan dengan hak istimewa yang tinggi, memberi Anda fleksibilitas untuk mengonfigurasi pekerja Anda untuk sistem Anda.

Deadline Cloud menjalankan skrip setelah pekerja memasuki STARTING status dan sebelum menjalankan tugas apa pun.

Important

Skrip berjalan dengan izin tinggi, sudo pada Linux sistem dan 'Administrator' pada Windows sistem. Adalah tanggung jawab Anda untuk memastikan bahwa skrip tidak menimbulkan masalah keamanan apa pun.

Saat Anda menggunakan skrip admin, Anda bertanggung jawab untuk memantau kesehatan armada Anda.

Penggunaan umum untuk skrip meliputi:

- Menginstal perangkat lunak yang membutuhkan akses administrator
- Memasang Docker kontainer

Anda dapat membuat dan memperbarui skrip konfigurasi host menggunakan konsol atau menggunakan file AWS CLI.

Console

1. Pada halaman Detail armada, pilih tab Konfigurasi.
2. Di bidang Script, masukkan skrip untuk dijalankan dengan izin yang ditinggikan. Anda dapat memilih Impor untuk memuat skrip dari workstation Anda.
3. Tetapkan periode batas waktu dalam hitungan detik untuk menjalankan skrip. Default-nya adalah 300 detik (5 menit).
4. Pilih Simpan perubahan untuk menyimpan skrip.

Create with CLI

Gunakan AWS CLI perintah berikut untuk membuat armada dengan skrip konfigurasi host. Ganti *placeholder* teks dengan informasi Anda.

```
aws deadline-internal create-fleet \  
--farm-id farm-12345 \  
--display-name "fleet-name" \  
--max-worker-count 1 \  
--configuration '{  
  "serviceManagedEc2": {
```

```

    "instanceCapabilities": {
      "vCpuCount": {"min": 2},
      "memoryMiB": {"min": 4096},
      "osFamily": "linux",
      "cpuArchitectureType": "x86_64"
    },
    "instanceMarketOptions": {"type": "spot"}
  }
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
value}'

```

Update with CLI

Gunakan AWS CLI perintah berikut untuk memperbarui skrip konfigurasi host armada. Ganti *placeholder* teks dengan informasi Anda.

```

aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
value}'

```

Skrip berikut menunjukkan:

- Variabel lingkungan yang tersedia untuk skrip
- AWS Kredensi itu berfungsi di shell
- Bahwa skrip berjalan di shell yang ditinggikan

Linux

Gunakan skrip berikut untuk menunjukkan bahwa skrip berjalan dengan root hak istimewa:

```

# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity

```

Windows

Gunakan PowerShell skrip berikut untuk menunjukkan bahwa skrip berjalan dengan hak istimewa Administrator:

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
    $currentUser = New-Object
    Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
    $isAdmin =
    $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

    return $isAdmin
}

if (Test-AdminPrivileges) {
    Write-Host "The current PowerShell session is elevated (running as
Administrator)."
} else {
    Write-Host "The current PowerShell session is not elevated (not running as
Administrator)."
}
exit 0
```

Memecahkan masalah skrip konfigurasi host

Saat Anda menjalankan skrip konfigurasi host:

- Tentang kesuksesan: Pekerja menjalankan pekerjaan
- Pada kegagalan (kode keluar bukan nol atau crash):
 - Pekerja menutup

Armada secara otomatis meluncurkan pekerja baru menggunakan skrip konfigurasi host terbaru

Untuk memantau skrip:

1. Buka halaman armada di konsol Deadline Cloud.
2. Pilih Lihat pekerja untuk membuka monitor Deadline Cloud.

3. Lihat status pekerja di halaman monitor.

Catatan penting:

- Pekerja yang ditutup karena kesalahan tidak tersedia dalam daftar pekerja di monitor. Gunakan CloudWatch Log untuk melihat log pekerja di grup log berikut:

```
/aws/deadline/farm-XXXXX/fleet-YYYYY
```

Di dalam grup log itu ada aliran

```
worker-ZZZZZ
```

- CloudWatch Log menyimpan log pekerja sesuai dengan periode retensi yang dikonfigurasi.

Memantau eksekusi skrip konfigurasi host

Dengan skrip Admin untuk mengonfigurasi pekerja, Anda dapat mengendalikan penuh pekerja Deadline Cloud. Anda dapat menginstal paket perangkat lunak apa pun, mengkonfigurasi ulang parameter sistem operasi, atau memasang sistem file bersama. Dengan fitur canggih ini dan kemampuan Deadline Cloud untuk menskalakan ke ribuan pekerja, kini Anda dapat memantau kapan skrip konfigurasi berhasil dijalankan atau gagal. Penyebab kegagalan mungkin termasuk kesalahan skrip, perilaku bersisik, atau situasi tidak diketahui lainnya.

Kami merekomendasikan solusi berikut untuk memantau eksekusi skrip konfigurasi host.

CloudWatch Pemantauan log

Semua log konfigurasi host armada dialirkan ke grup CloudWatch log armada, dan secara khusus ke aliran CloudWatch log pekerja. Misalnya, `/aws/deadline/farm-123456789012/fleet-777788889999` adalah grup log untuk `pertanian123456789012`, `armada777788889999`.

Setiap pekerja menyediakan aliran log khusus, misalnyaworker-123456789012. Log konfigurasi host termasuk spanduk log seperti Running Host Configuration Script dan Selesai menjalankan Host Configuration Script, kode keluar: 0. Kode keluar skrip disertakan dalam spanduk yang sudah jadi dan dapat ditanyakan menggunakan CloudWatch alat.

CloudWatch Wawasan Log

CloudWatch Logs Insights menawarkan kemampuan canggih untuk menganalisis informasi log. Misalnya, kueri Log Insights berikut mengurai kode keluar konfigurasi host, diurutkan berdasarkan waktu:

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat [Menganalisis data CloudWatch log dengan Wawasan Log](#) di Panduan Pengguna CloudWatch Log Amazon.

Pencatatan terstruktur agen pekerja

Agen pekerja Deadline Cloud menerbitkan log JSON terstruktur ke CloudWatch Agen pekerja menawarkan beragam log terstruktur untuk menganalisis kesehatan pekerja. Untuk informasi selengkapnya, lihat [Deadline Login agen pekerja Cloud](#). GitHub

Atribut log terstruktur dibongkar ke bidang di Wawasan Log. Anda dapat menggunakan CloudWatch kemampuan ini untuk menghitung dan menganalisis kegagalan startup konfigurasi host. Misalnya, kueri count dan bin dapat digunakan untuk menentukan seberapa sering kegagalan terjadi:

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

CloudWatch filter metrik untuk metrik dan mengkhawatirkan

Anda dapat mengatur filter CloudWatch metrik untuk menghasilkan CloudWatch metrik dari log. Ini memungkinkan Anda membuat alarm dan dasbor untuk memantau eksekusi skrip konfigurasi host.

Untuk membuat filter metrik

1. Buka CloudWatch konsol.
2. Di panel navigasi, pilih Log, lalu Log grup.
3. Pilih grup log armada Anda.

4. Pilih Create metric filter (Buat filter metrik).
5. Tentukan pola filter Anda menggunakan salah satu dari berikut ini:
 - Untuk metrik sukses:

```
{$.message = "*Worker Agent host configuration succeeded.*"}
```

- Untuk metrik kegagalan:

```
{$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with exit code*"}
```

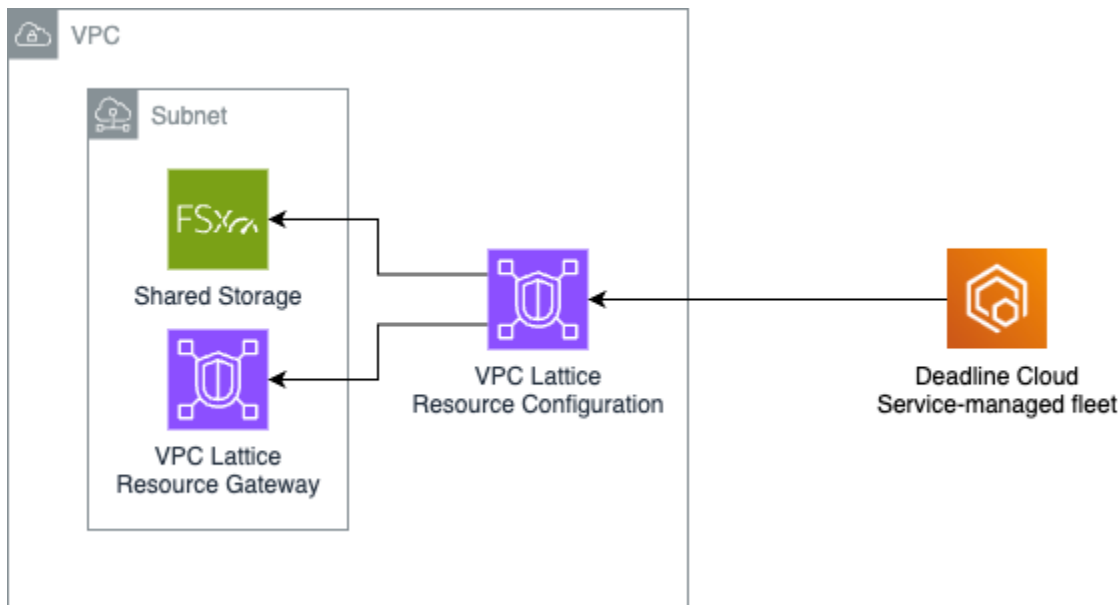
6. Pilih Berikutnya untuk membuat metrik dengan nilai berikut:
 - Namespace metrik: Namespace metrik Anda (misalnya,) **MyDeadlineFarm**
 - Nama metrik: Nama metrik yang Anda minta (misalnya, **host_config_failure**)
 - Nilai metrik: **1** (setiap contoh adalah hitungan 1)
 - Nilai default: Biarkan kosong
 - Satuan: **Count**

Setelah membuat filter metrik, Anda dapat mengonfigurasi CloudWatch alarm standar untuk mengambil tindakan pada tingkat kegagalan konfigurasi host yang meningkat, atau menambahkan metrik ke CloudWatch dasbor untuk day-to-day pengoperasian dan pemantauan.

Untuk detail selengkapnya, lihat [Filter dan sintaks pola](#) di Panduan Pengguna Amazon CloudWatch Logs.

Hubungkan sumber daya VPC ke SMF Anda dengan titik akhir sumber daya VPC

Dengan titik akhir sumber daya Amazon VPC untuk armada yang dikelola layanan Deadline Cloud (SMF), Anda dapat menghubungkan sumber daya VPC Anda seperti sistem file jaringan (NFS), server lisensi, dan database dengan pekerja Deadline Cloud Anda. Fitur ini memungkinkan Anda memanfaatkan platform Deadline Cloud yang dikelola sepenuhnya sambil berintegrasi dengan infrastruktur yang ada dalam VPC.



Cara kerja titik akhir sumber daya VPC

Titik akhir sumber daya VPC menggunakan VPC Lattice untuk membuat koneksi aman antara pekerja SMF Cloud Deadline Anda dan sumber daya di VPC Anda. Koneksinya searah, artinya pekerja dapat membuat koneksi ke sumber daya di VPC Anda dan mentransfer data bolak-balik, tetapi sumber daya di VPC Anda tidak dapat membuat koneksi ke pekerja.

Saat Anda menghubungkan sumber daya VPC ke armada yang dikelola layanan Deadline Cloud, pekerja Anda dapat mengakses sumber daya di VPC menggunakan nama domain pribadi. Selain itu, arus lalu lintas dari pekerja ke sumber daya VPC Anda melalui VPC Lattice, dan sumber daya di VPC Anda melihat lalu lintas yang berasal dari gateway sumber daya VPC Lattice.

Untuk mempelajari lebih lanjut, lihat panduan [pengguna VPC Lattice](#).

Prasyarat

Sebelum menghubungkan sumber daya VPC ke armada yang dikelola layanan Deadline Cloud, pastikan Anda memiliki hal-hal berikut:

- AWS Akun dengan VPC yang berisi sumber daya yang ingin Anda sambungkan.
- Izin IAM untuk membuat dan mengelola sumber daya VPC Lattice.
- Peternakan Cloud Deadline dengan setidaknya satu armada yang dikelola layanan.
- Sumber daya VPC yang ingin Anda buat dapat diakses (FSx, NFS, server lisensi, dll.).

Siapkan titik akhir sumber daya VPC

Untuk menyiapkan titik akhir sumber daya VPC, Anda perlu membuat sumber daya di [VPC Lattice](#) dan [AWS RAM](#), lalu menghubungkan sumber daya tersebut ke armada Anda di Deadline Cloud. Untuk menyiapkan titik akhir sumber daya VPC untuk SMF Anda, selesaikan langkah-langkah berikut.

1. Untuk membuat gateway sumber daya di VPC Lattice, lihat [Membuat gateway sumber daya](#) di panduan pengguna VPC Lattice.
2. Untuk membuat konfigurasi sumber daya di VPC Lattice, lihat [Membuat konfigurasi sumber daya](#) di panduan pengguna VPC Lattice.
3. Untuk berbagi sumber daya dengan armada Deadline Cloud Anda, buat pembagian sumber daya di AWS RAM. Lihat [Membuat berbagi sumber daya](#) untuk instruksi.

Saat membuat pembagian sumber daya, untuk Prinsipal, pilih Prinsipal layanan dari menu tarik-turun, lalu masukkan. **fleets.deadline.amazonaws.com**

4. Untuk menghubungkan konfigurasi sumber daya dengan armada Deadline Cloud Anda, selesaikan langkah-langkah berikut.
 - a. Jika Anda belum melakukannya, buka [konsol Deadline Cloud](#).
 - b. Di panel navigasi, pilih Peternakan, lalu pilih peternakan Anda.
 - c. Pilih tab Armada, lalu pilih armada Anda.
 - d. Pilih tab Konfigurasi.
 - e. Di bawah titik akhir sumber daya VPC, pilih Edit.
 - f. Pilih konfigurasi sumber daya yang Anda buat, lalu pilih Simpan perubahan.

Mengakses sumber daya VPC Anda

Setelah menghubungkan sumber daya VPC Anda ke armada Anda, pekerja dapat mengaksesnya menggunakan nama domain pribadi dalam format berikut:

```
<resource_config_id>.deadline.<region>.amazonaws.com
```

Domain ini bersifat pribadi dan hanya dapat diakses oleh pekerja (bukan dari internet atau workstation Anda).

Otentikasi dan keamanan

Untuk sumber daya yang memerlukan otentikasi, simpan kredensial dengan aman di AWS Secrets Manager, akses rahasia dari skrip konfigurasi host atau skrip pekerjaan Anda, dan terapkan izin sistem file yang sesuai untuk mengontrol akses. Pertimbangkan implikasi keamanan saat berbagi sumber daya di beberapa armada. Misalnya, jika dua armada terhubung ke penyimpanan bersama yang sama, pekerjaan yang berjalan pada satu armada mungkin dapat mengakses aset yang dibuat dari armada lainnya.

Pertimbangan teknis

Saat menggunakan titik akhir sumber daya VPC, pertimbangkan hal berikut:

- Koneksi hanya dapat dimulai dari pekerja ke sumber daya VPC, bukan dari sumber daya VPC ke pekerja.
- Setelah dibuat, koneksi tetap ada hingga diatur ulang, bahkan jika konfigurasi sumber daya terputus.
- Koneksi VPC Lattice menangani koneksi antara Availability Zones secara otomatis tanpa biaya tambahan. Gateway sumber daya Anda harus berbagi Availability Zone dengan resource VPC Anda, jadi sebaiknya konfigurasi gateway sumber daya untuk menjangkau semua Availability Zone.
- Lalu lintas melalui titik akhir sumber daya VPC menggunakan Network Address Translation (NAT), yang tidak kompatibel dengan semua kasus penggunaan. Misalnya, Microsoft Active Directory (AD) tidak dapat terhubung melalui NAT.

[Untuk informasi selengkapnya tentang kuota Kisi VPC, lihat Kuota untuk Kisi VPC.](#)

Pemecahan Masalah

Jika Anda mengalami masalah dengan titik akhir sumber daya VPC, periksa yang berikut ini.

- Jika Anda menerima pesan kesalahan seperti “mount.nfs: akses ditolak oleh server saat pemasangan,” Anda mungkin perlu memperbarui konfigurasi klien volume NFS Anda.
- Verifikasi pengaturan konfigurasi sumber daya Anda dengan menguji dari EC2 instans Amazon atau AWS CloudShell di VPC Anda.
- Uji koneksi Deadline Cloud Anda dengan pekerjaan CLI sederhana. Untuk informasi selengkapnya, lihat [contoh Deadline Cloud di GitHub](#).

- Periksa pengaturan pada grup keamanan gateway sumber daya jika Anda mengalami kegagalan koneksi.
- Aktifkan log akses VPC untuk memantau koneksi.

Menggunakan lisensi perangkat lunak dengan Deadline Cloud

Deadline Cloud menyediakan dua metode untuk menyediakan lisensi perangkat lunak untuk pekerjaan Anda:

- Lisensi berbasis penggunaan (UBL) — melacak dan menagih berdasarkan jumlah jam yang digunakan armada Anda untuk memproses pekerjaan. Tidak ada jumlah lisensi yang ditetapkan sehingga armada Anda dapat menskalakan sesuai kebutuhan. UBL adalah standar untuk armada yang dikelola layanan. Untuk armada yang dikelola pelanggan, Anda dapat menghubungkan titik akhir lisensi Deadline Cloud untuk UBL. UBL menyediakan lisensi untuk pekerja Deadline Cloud Anda untuk dirender, itu tidak memberikan lisensi untuk aplikasi DCC Anda.
- Bawa lisensi Anda sendiri (BYOL) — memungkinkan Anda untuk menggunakan lisensi perangkat lunak yang ada dengan armada yang dikelola layanan atau pelanggan Anda. Anda dapat menggunakan BYOL untuk terhubung ke server lisensi untuk perangkat lunak yang tidak didukung oleh lisensi berbasis penggunaan Deadline Cloud. Anda dapat menggunakan BYOL dengan armada yang dikelola layanan dengan menghubungkan ke server lisensi khusus.

Topik

- [Connect armada yang dikelola layanan ke server lisensi kustom](#)
- [Connect armada yang dikelola pelanggan ke titik akhir lisensi](#)

Connect armada yang dikelola layanan ke server lisensi kustom

Anda dapat membawa server lisensi Anda sendiri untuk digunakan dengan armada yang dikelola layanan Deadline Cloud. Untuk membawa lisensi Anda sendiri, Anda dapat mengonfigurasi server lisensi menggunakan lingkungan antrian di peternakan Anda. Untuk mengonfigurasi server lisensi Anda, Anda harus sudah menyiapkan pertanian dan antrian.

Bagaimana Anda terhubung ke server lisensi perangkat lunak tergantung pada konfigurasi armada Anda dan persyaratan vendor perangkat lunak. Biasanya, Anda mengakses server dengan salah satu dari dua cara:

- Langsung ke server lisensi. Pekerja Anda mendapatkan lisensi dari server lisensi vendor perangkat lunak menggunakan Internet. Semua pekerja Anda harus dapat terhubung ke server.

- Melalui proxy lisensi. Pekerja Anda terhubung ke server proxy di jaringan lokal Anda. Hanya server proxy yang diizinkan untuk terhubung ke server lisensi vendor melalui Internet.

Dengan petunjuk di bawah ini, Anda menggunakan Amazon EC2 Systems Manager (SSM) untuk meneruskan port dari instans pekerja ke server lisensi atau instance proxy Anda.

Topik

- [Langkah 1: Konfigurasi lingkungan antrian](#)
- [Langkah 2: \(Opsional\) Pengaturan instance proxy lisensi](#)
- [Langkah 3: pengaturan AWS CloudFormation template](#)

Langkah 1: Konfigurasi lingkungan antrian

Anda dapat mengonfigurasi lingkungan antrian dalam antrian untuk mengakses server lisensi Anda. Pertama, pastikan bahwa Anda memiliki AWS instance yang dikonfigurasi dengan akses server lisensi menggunakan salah satu metode berikut:

- Server lisensi — Instance meng-host server lisensi secara langsung.
- Proxy lisensi — Instance memiliki akses jaringan ke server lisensi, dan meneruskan port server lisensi ke server lisensi. Untuk detail tentang cara mengonfigurasi instance proxy lisensi, lihat [Langkah 2: \(Opsional\) Pengaturan instance proxy lisensi](#).

Untuk menambahkan izin yang diperlukan ke peran antrian

1. Dari [konsol Cloud Deadline](#), pilih Buka Dasbor.
2. Dari dasbor, pilih pertanian, lalu antrian yang ingin Anda konfigurasi.
3. Dari detail antrian > peran layanan, pilih peran.
4. Pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.
5. Pilih editor kebijakan JSON, lalu salin dan tempel teks berikut ke editor.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "s3:*",  
      "Resource": "*" }  
    ]  
}
```

```

    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": [
        "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
        "arn:aws:ec2:us-east-1:account_id:instance/instance_id"
      ]
    }
  ]
}

```

6. Sebelum menyimpan kebijakan baru, ganti nilai berikut dalam teks kebijakan:
 - Ganti `region` dengan AWS Wilayah tempat peternakan Anda berada
 - Ganti `instance_id` dengan ID instance untuk server lisensi atau instance proxy yang Anda gunakan
 - Ganti `account_id` dengan nomor AWS rekening yang berisi peternakan Anda
7. Pilih Berikutnya.
8. Untuk nama Policy, masukkan **LicenseForwarding**.
9. Pilih Buat kebijakan untuk menyimpan perubahan dan membuat kebijakan dengan izin yang diperlukan.

Untuk menambahkan lingkungan antrian baru ke antrian

1. Dari [konsol Deadline Cloud](#), pilih Go to Dashboard jika Anda belum melakukannya.
2. Dari dasbor, pilih pertanian, lalu antrian yang ingin Anda konfigurasi.
3. Pilih Lingkungan Antrian > Tindakan > Buat baru dengan YAMAL.
4. Salin dan tempel teks berikut ke editor skrip YAMAL.

Windows

```

specificationVersion: "environment-2023-09"
parameterDefinitions:
  - name: LicenseInstanceId

```

```

    type: STRING
    description: >
        The Instance ID of the license server/proxy instance
    default: ""
- name: LicenseInstanceRegion
  type: STRING
  description: >
    The region containing this farm
  default: ""
- name: LicensePorts
  type: STRING
  description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance.
    Example: "2700,2701,2702"
  default: ""
environment:
  name: BYOL License Forwarding
  variables:
    example_LICENSE: 2700@localhost
  script:
    actions:
      onEnter:
        command: powershell
        args: [ "{{Env.File.Enter}}" ]
      onExit:
        command: powershell
        args: [ "{{Env.File.Exit}}" ]
    embeddedFiles:
      - name: Enter
        filename: enter.ps1
        type: TEXT
        runnable: True
        data: |
          $ZIP_NAME="SessionManagerPlugin.zip"
          Invoke-WebRequest -Uri "https://s3.amazonaws.com/session-manager-
downloads/plugin/latest/windows/$ZIP_NAME" -OutFile $ZIP_NAME
          Expand-Archive -Path $ZIP_NAME
          Expand-Archive -Path .\SessionManagerPlugin\package.zip
          conda activate
          python {{Env.File.StartSession}} {{Session.WorkingDirectory}}\package
\bin\session-manager-plugin.exe
      - name: Exit
        filename: exit.ps1

```

```

type: TEXT
runnable: True
data: |
    Write-Output "Killing SSM Manager Plugin PIDs: $env:BYOL_SSM_PIDS"
    "$env:BYOL_SSM_PIDS".Split(",") | ForEach {
        Write-Output "Killing $_"
        Stop-Process -Id $_ -Force
    }
- name: StartSession
type: TEXT
data: |
    import boto3
    import json
    import subprocess
    import sys

    instance_id = "{{Param.LicenseInstanceId}}"
    region = "{{Param.LicenseInstanceRegion}}"
    license_ports_list = "{{Param.LicensePorts}}".split(",")

    ssm_client = boto3.client("ssm", region_name=region)
    pids = []

    for port in license_ports_list:
        session_response = ssm_client.start_session(
            Target=instance_id,
            DocumentName="AWS-StartPortForwardingSession",
            Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
            sys.argv[1],
            json.dumps(session_response),
            region,
            "StartSession",
            "",
            json.dumps({"Target": instance_id}),
            f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

```

```
print(f"openjd_env: BYOL_SSM_PIDS={' '.join(str(pid) for pid in
pids)}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
  - name: LicenseInstanceId
    type: STRING
    description: >
      The Instance ID of the license server/proxy instance
    default: ""
  - name: LicenseInstanceRegion
    type: STRING
    description: >
      The region containing this farm
    default: ""
  - name: LicensePorts
    type: STRING
    description: >
      Comma-separated list of ports to be forwarded to the license server/proxy
      instance.
      Example: "2700,2701,2702"
    default: ""
environment:
  name: BYOL License Forwarding
  variables:
    example_LICENSE: 2700@localhost
  script:
    actions:
      onEnter:
        command: bash
        args: [ "${Env.File.Enter}"]
      onExit:
        command: bash
        args: [ "${Env.File.Exit}" ]
  embeddedFiles:
    - name: Enter
      type: TEXT
      runnable: True
```

```

    data: |
        curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
--to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
{{Session.WorkingDirectory}}/session-manager-plugin
        chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
        conda activate
        python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
- name: Exit
  type: TEXT
  runnable: True
  data: |
    echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
    for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
- name: StartSession
  type: TEXT
  data: |
    import boto3
    import json
    import subprocess
    import sys

    instance_id = "{{Param.LicenseInstanceId}}"
    region = "{{Param.LicenseInstanceRegion}}"
    license_ports_list = "{{Param.LicensePorts}}".split(",")

    ssm_client = boto3.client("ssm", region_name=region)
    pids = []

    for port in license_ports_list:
        session_response = ssm_client.start_session(
            Target=instance_id,
            DocumentName="AWS-StartPortForwardingSession",
            Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
            sys.argv[1],
            json.dumps(session_response),
            region,
            "StartSession",
            "",
            json.dumps({"Target": instance_id}),

```

```

        f"https://ssm.{region}.amazonaws.com"
    ]

    process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
    pids.append(process.pid)
    print(f"SSM Port Forwarding Session started for port {port}")

    print(f"openjd_env: BYOL_SSM_PIDS='{','.join(str(pid) for pid in
pids)}'")

```

5. Sebelum menyimpan lingkungan antrian, buat perubahan berikut pada teks lingkungan sesuai kebutuhan:
 - Perbarui nilai default untuk parameter berikut untuk mencerminkan lingkungan Anda:
 - LicenseInstanceId - ID EC2 instans Amazon dari server lisensi atau instance proxy Anda
 - LicenseInstanceRegion— AWS Wilayah yang berisi peternakan Anda
 - LicensePorts— Daftar port yang dipisahkan koma untuk diteruskan ke server lisensi atau instance proxy (misalnya 2700,2701)
 - Tambahkan variabel lingkungan lisensi yang diperlukan ke bagian variabel. Variabel-variabel ini harus mengarahkan DCCs ke localhost pada port server lisensi. Misalnya, jika server lisensi Foundry Anda mendengarkan pada port 6101, Anda akan menambahkan variabel sebagai.
 foundry_LICENSE: 6101@localhost
6. (Opsional) Anda dapat membiarkan Prioritas disetel ke 0, atau Anda dapat mengubahnya untuk mengurutkan prioritas secara berbeda di antara beberapa lingkungan antrian.
7. Pilih Buat lingkungan antrian untuk menyelamatkan lingkungan baru.

Dengan pengaturan lingkungan antrian, pekerjaan yang dikirimkan ke antrian ini akan mengambil lisensi dari server lisensi yang dikonfigurasi.

Langkah 2: (Opsional) Pengaturan instance proxy lisensi

Sebagai alternatif untuk menggunakan server lisensi, Anda dapat menggunakan proxy lisensi. Untuk membuat proxy lisensi, buat instans Amazon Linux 2023 baru yang memiliki akses jaringan ke server lisensi. Jika perlu, Anda dapat mengonfigurasi akses ini menggunakan koneksi VPN. Untuk informasi selengkapnya, lihat [Koneksi VPN](#) di Panduan Pengguna Amazon VPC.

Untuk menyiapkan instance proxy lisensi untuk Deadline Cloud, ikuti langkah-langkah dalam prosedur ini. Lakukan langkah-langkah konfigurasi berikut pada instance baru ini untuk mengaktifkan penerusan lalu lintas lisensi ke server lisensi Anda

1. Untuk menginstal HAProxy paket, masukkan

```
sudo yum install haproxy
```

2. Perbarui bagian server lisensi dengarkan dari file konfigurasi/etc/haproxy/haproxy.cfg dengan yang berikut ini:
 - a. Ganti LicensePort1 dan LicensePort2 dengan nomor port yang akan diteruskan ke server lisensi. Tambahkan atau hapus nilai yang dipisahkan koma untuk mengakomodasi jumlah port yang diperlukan.
 - b. Ganti LicenseServerHostdengan nama host atau alamat IP server lisensi.

```
global
    log          127.0.0.1 local2
    chroot       /var/lib/haproxy
    user         haproxy
    group        haproxy
    daemon

defaults
    timeout queue          1m
    timeout connect        10s
    timeout client         1m
    timeout server         1m
    timeout http-keep-alive 10s
    timeout check          10s

listen license-server
    bind *:LicensePort1,*:LicensePort2
    server license-server LicenseServerHost
```

3. Untuk mengaktifkan dan memulai HAProxy layanan, jalankan perintah berikut:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Setelah menyelesaikan langkah-langkah, permintaan lisensi yang dikirim ke localhost dari lingkungan antrian penerusan harus diteruskan ke server lisensi yang ditentukan.

Langkah 3: pengaturan AWS CloudFormation template

Anda dapat menggunakan AWS CloudFormation template untuk mengonfigurasi seluruh peternakan untuk menggunakan lisensi Anda sendiri.

1. Ubah template yang disediakan di langkah berikutnya untuk menambahkan variabel lingkungan lisensi yang diperlukan ke bagian variabel di bawah BYOLQueueLingkungan.
2. Gunakan AWS CloudFormation template berikut.

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create Deadline Cloud resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm

  QueuePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
```

```

ManagedPolicyName: BYOLQueuePolicy
PolicyDocument:
  Version: 2012-10-17
  Statement:
    - Effect: Allow
      Action:
        - s3:GetObject
        - s3:PutObject
        - s3:ListBucket
        - s3:GetBucketLocation
      Resource:
        - !Sub ${JobAttachmentBucket.Arn}
        - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
      Condition:
        StringEquals:
          aws:ResourceAccount: !Sub ${AWS::AccountId}
    - Effect: Allow
      Action: logs:GetLogEvents
      Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
    - Effect: Allow
      Action:
        - s3:ListBucket
        - s3:GetObject
      Resource:
        - "*"
      Condition:
        ArnLike:
          s3:DataAccessPointArn:
            - arn:aws:s3:*:*:accesspoint/deadline-software-*
        StringEquals:
          s3:AccessPointNetworkOrigin: VPC

BYOLSSMPolicy:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    ManagedPolicyName: BYOLSSMPolicy
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - ssm:StartSession
          Resource:

```

```

        - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
        - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}

WorkerPolicy:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    ManagedPolicyName: BYOLWorkerPolicy
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - logs:CreateLogStream
          Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          Condition:
            ForAnyValue:StringEquals:
              aws:CalledVia:
                - deadline.amazonaws.com
        - Effect: Allow
          Action:
            - logs:PutLogEvents
            - logs:GetLogEvents
          Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*

QueueRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: BYOLQueueRole
    ManagedPolicyArns:
      - !Ref QueuePolicy
      - !Ref BYOLSSMPolicy
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - sts:AssumeRole
          Principal:

```

```
    Service:
      - credentials.deadline.amazonaws.com
      - deadline.amazonaws.com
    Condition:
      StringEquals:
        aws:SourceAccount: !Sub ${AWS::AccountId}
      ArnEquals:
        aws:SourceArn: !Ref Farm

WorkerRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: BYOLWorkerRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
      - !Ref WorkerPolicy
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - sts:AssumeRole
          Principal:
            Service: credentials.deadline.amazonaws.com

Queue:
  Type: AWS::Deadline::Queue
  Properties:
    DisplayName: BYOLQueue
    FarmId: !GetAtt Farm.FarmId
    RoleArn: !GetAtt QueueRole.Arn
    JobRunAsUser:
      Posix:
        Group: ""
        User: ""
      RunAs: WORKER_AGENT_USER
    JobAttachmentSettings:
      RootPrefix: job-attachments
      S3BucketName: !Ref JobAttachmentBucket

Fleet:
  Type: AWS::Deadline::Fleet
  Properties:
```

```

DisplayName: BYOLFleet
FarmId: !GetAtt Farm.FarmId
MinWorkerCount: 1
MaxWorkerCount: 2
Configuration:
  ServiceManagedEc2:
    InstanceCapabilities:
      VCpuCount:
        Min: 4
        Max: 16
      MemoryMiB:
        Min: 4096
        Max: 16384
      OsFamily: LINUX
      CpuArchitectureType: x86_64
    InstanceMarketOptions:
      Type: on-demand
  RoleArn: !GetAtt WorkerRole.Arn

```

QFA:

```

Type: AWS::Deadline::QueueFleetAssociation
Properties:
  FarmId: !GetAtt Farm.FarmId
  FleetId: !GetAtt Fleet.FleetId
  QueueId: !GetAtt Queue.QueueId

```

CondaQueueEnvironment:

```

Type: AWS::Deadline::QueueEnvironment
Properties:
  FarmId: !GetAtt Farm.FarmId
  Priority: 5
  QueueId: !GetAtt Queue.QueueId
  TemplateType: YAML
  Template: |
    specificationVersion: 'environment-2023-09'
    parameterDefinitions:
      - name: CondaPackages
        type: STRING
        description: >
          This is a space-separated list of Conda package match specifications to
install for the job.
          E.g. "blender=3.6" for a job that renders frames in Blender 3.6.

```

See <https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/pkg-specs.html#package-match-specifications>

```
default: ""
userInterface:
  control: LINE_EDIT
  label: Conda Packages
- name: CondaChannels
  type: STRING
  description: >
```

This is a space-separated list of Conda channels from which to install packages. Deadline Cloud SMF packages are

installed from the "deadline-cloud" channel that is configured by Deadline Cloud.

Add "conda-forge" to get packages from the [https://conda-forge.org/](https://conda-forge.org/community) community, and "defaults" to get packages from Anaconda Inc (make sure your usage complies with <https://www.anaconda.com/terms-of-use>).

```
default: "deadline-cloud"
userInterface:
  control: LINE_EDIT
  label: Conda Channels
environment:
  name: Conda
  script:
    actions:
      onEnter:
        command: "conda-queue-env-enter"
        args: ["${Session.WorkingDirectory}"/".env", "--packages",
"${Param.CondaPackages}", "--channels", "${Param.CondaChannels}"]
      onExit:
        command: "conda-queue-env-exit"
```

BYOLQueueEnvironment:

Type: AWS::Deadline::QueueEnvironment

Properties:

FarmId: !GetAtt Farm.FarmId

Priority: 10

QueueId: !GetAtt Queue.QueueId

TemplateType: YAML

Template: !Sub |

specificationVersion: "environment-2023-09"

parameterDefinitions:

- name: LicenseInstanceId

```

    type: STRING
    description: >
      The Instance ID of the license server/proxy instance
    default: "${LicenseInstanceId}"
  - name: LicenseInstanceRegion
    type: STRING
    description: >
      The region containing this farm
    default: "${AWS::Region}"
  - name: LicensePorts
    type: STRING
    description: >
      Comma-separated list of ports to be forwarded to the license server/
      proxy instance.
      Example: "2700,2701,2702"
    default: "${LicensePorts}"
  environment:
    name: BYOL License Forwarding
    variables:
      example_LICENSE: 2700@localhost
    script:
      actions:
        onEnter:
          command: bash
          args: [ "${Env.File.Enter}"]
        onExit:
          command: bash
          args: [ "${Env.File.Exit}"]
      embeddedFiles:
        - name: Enter
          type: TEXT
          runnable: True
          data: |
            curl https://s3.amazonaws.com/session-manager-downloads/
            plugin/latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio
            -iv --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
            {{Session.WorkingDirectory}}/session-manager-plugin
            chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
            conda activate
            python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
            session-manager-plugin
        - name: Exit
          type: TEXT
          runnable: True

```

```

data: |
    echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
    for pid in ${!BYOL_SSM_PIDS//,/ }; do kill $pid; done
- name: StartSession
  type: TEXT
  data: |
    import boto3
    import json
    import subprocess
    import sys

    instance_id = "{{Param.LicenseInstanceId}}"
    region = "{{Param.LicenseInstanceRegion}}"
    license_ports_list = "{{Param.LicensePorts}}".split(",")

    ssm_client = boto3.client("ssm", region_name=region)
    pids = []

    for port in license_ports_list:
        session_response = ssm_client.start_session(
            Target=instance_id,
            DocumentName="AWS-StartPortForwardingSession",
            Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
            sys.argv[1],
            json.dumps(session_response),
            region,
            "StartSession",
            "",
            json.dumps({"Target": instance_id}),
            f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

    print(f"openjd_env: BYOL_SSM_PIDS='{','.join(str(pid) for pid in
pids)}'")

```

3. Saat menerapkan AWS CloudFormation template, berikan parameter berikut:
 - Memperbarui LicenseInstanceID dengan ID EC2 Instans Amazon dari server lisensi atau instans proxy Anda
 - Perbarui LicensePorts dengan daftar port yang dipisahkan koma untuk diteruskan ke server lisensi atau instance proxy (misalnya 2700,2701)
4. Terapkan template untuk menyiapkan peternakan Anda dengan membawa kemampuan lisensi Anda sendiri.

Connect armada yang dikelola pelanggan ke titik akhir lisensi

Server lisensi berbasis penggunaan AWS Deadline Cloud menyediakan lisensi sesuai permintaan untuk produk pihak ketiga tertentu. Dengan lisensi berbasis penggunaan, Anda dapat membayar sesuai keinginan Anda. Anda hanya dikenakan biaya untuk waktu yang Anda gunakan. Lisensi berbasis penggunaan menyediakan lisensi untuk dirender oleh pekerja Cloud Deadline Anda, tidak memberikan lisensi untuk aplikasi DCC Anda.

Server lisensi berbasis penggunaan Deadline Cloud dapat digunakan dengan jenis armada apa pun selama pekerja Deadline Cloud dapat berkomunikasi dengan server lisensi. Ini secara otomatis diatur dalam armada yang dikelola layanan. Pengaturan ini hanya diperlukan untuk armada yang dikelola pelanggan.

Untuk membuat server lisensi, Anda memerlukan grup keamanan untuk VPC pertanian Anda yang memungkinkan lalu lintas untuk lisensi pihak ketiga.

Topik

- [Langkah 1: Buat grup keamanan](#)
- [Langkah 2: Siapkan titik akhir lisensi](#)
- [Langkah 3: Hubungkan aplikasi rendering ke titik akhir](#)
- [Langkah 4: Hapus titik akhir lisensi](#)

Langkah 1: Buat grup keamanan

Gunakan [Konsol VPC Amazon](#) untuk membuat grup keamanan untuk VPC pertanian Anda. Konfigurasi grup keamanan untuk mengizinkan aturan masuk berikut:

- Autodesk Maya dan Arnold - 2701 - 2702, TCP,, IPv4 IPv6

- Autodesk 3ds Maks - 2704, TCP,, IPv4 IPv6
- Bioskop 4D — 7057, TCP,, IPv4 IPv6
- KeyShot — 2703, TCP,, IPv4 IPv6
- Pengecoran Nuke — 6101, TCP,, IPv4 IPv6
- Pergeseran Merah — 7054, TCP,, IPv4 IPv6
- SidFX Houdini, Mantra, dan Karma — 1715 - 1717, TCP,, IPv4 IPv6
- V-Ray — 30304, TCP, IPV4

Sumber untuk setiap aturan masuk adalah kelompok keamanan pekerja armada.

Untuk informasi selengkapnya tentang membuat grup keamanan, lihat [Membuat grup keamanan](#) di panduan pengguna Amazon Virtual Private Cloud.

Langkah 2: Siapkan titik akhir lisensi

Titik akhir lisensi menyediakan akses ke server lisensi untuk produk pihak ketiga. Permintaan lisensi dikirim ke titik akhir lisensi. Titik akhir merutekan mereka ke server lisensi yang sesuai. Server lisensi melacak batas penggunaan dan hak. Membuat titik akhir lisensi di Deadline Cloud menyediakan titik akhir AWS PrivateLink antarmuka di VPC Anda. Titik akhir ini ditagih sesuai dengan harga standar AWS PrivateLink . Untuk informasi selengkapnya, lihat [harga AWS PrivateLink](#).

Dengan izin yang sesuai, Anda dapat membuat titik akhir lisensi Anda. Untuk kebijakan yang diperlukan untuk membuat titik akhir lisensi, lihat [Kebijakan untuk mengizinkan pembuatan titik akhir lisensi](#).

[Anda dapat membuat titik akhir lisensi dari dasbor di konsol Deadline Cloud.](#)

1. Dari panel navigasi kiri, pilih titik akhir Lisensi, lalu pilih Buat titik akhir lisensi.
2. Dari halaman Create license endpoint, lengkapi yang berikut ini:
 - Pilih VPC.
 - Pilih subnet yang berisi pekerja Deadline Cloud Anda. Anda dapat memilih hingga 10 subnet.
 - Pilih grup keamanan yang Anda buat di langkah 1. Anda dapat memilih hingga 10 grup keamanan untuk skenario yang lebih rumit.
 - (Opsional) Pilih Tambahkan tag baru dan tambahkan satu atau beberapa tag. Anda dapat menambahkan hingga 50 tanda.

3. Pilih Buat titik akhir lisensi. Ketika titik akhir lisensi dibuat, itu akan ditampilkan pada halaman titik akhir lisensi.
4. Dari bagian produk terukur, pilih Tambahkan produk, lalu pilih produk yang ingin Anda tambahkan ke titik akhir lisensi Anda. Pilih Tambahkan.

Untuk menghapus produk dari titik akhir lisensi, di bagian produk terukur, pilih produk lalu pilih Hapus. Dalam konfirmasi, pilih Hapus lagi.

Langkah 3: Hubungkan aplikasi rendering ke titik akhir

Setelah titik akhir lisensi diatur, aplikasi menggunakannya sama seperti mereka menggunakan server lisensi pihak ketiga. Anda biasanya mengonfigurasi server lisensi untuk aplikasi dengan menetapkan variabel lingkungan atau pengaturan sistem lainnya, seperti kunci registri Microsoft Windows, ke port dan alamat server lisensi.

Untuk mendapatkan nama DNS titik akhir lisensi, pilih titik akhir lisensi di konsol dan kemudian pilih ikon salin di bagian Nama DNS.

Contoh konfigurasi

Example Autodesk Maya dan Arnold

Note

Anda dapat menggunakan Autodesk Maya dan Arnold bersama-sama atau secara terpisah. Gunakan port 2702 untuk Autodesk Maya dan port 2701 untuk Arnold.

Untuk Autodesk Maya, atur variabel lingkungan ADKFLEX_LICENSE_FILE ke:

```
2702@VPC_Endpoint_DNS_Name
```

Untuk Arnold, atur variabel lingkungan ADKFLEX_LICENSE_FILE ke:

```
2701@VPC_Endpoint_DNS_Name
```

Untuk Autodesk Maya dan Arnold, atur variabel ADKFLEX_LICENSE_FILE lingkungan ke:

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

Note

Untuk Windows pekerja, gunakan titik koma (;) alih-alih titik dua (:) untuk memisahkan titik akhir.

Example — Autodesk 3ds Maks

Atur variabel lingkungan ADSKFLEX_LICENSE_FILE ke:

```
2704@VPC_Endpoint_DNS_Name
```

Example — Bioskop 4D

Atur variabel lingkungan g_licenseServerRLM ke:

```
VPC_Endpoint_DNS_Name:7057
```

Setelah Anda membuat variabel lingkungan, Anda harus dapat membuat gambar menggunakan baris perintah yang mirip dengan yang ini:

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^  
"C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^  
-oimage "C:\Users\Administrator\User\MyOutputImage.png"
```

Example – KeyShot

Atur variabel lingkungan LUXION_LICENSE_FILE ke:

```
2703@VPC_Endpoint_DNS_Name
```

Setelah Anda menginstal KeyShot dan menjalankan `pip install deadline-cloud-for-keyshot` Anda dapat menguji lisensi bekerja menggunakan perintah berikut. Skrip memvalidasi pengaturan Anda tetapi tidak merender apa pun.

```
"C:\Program Files\KeyShot12\bin\keyshot_headless.exe" ^
```

```
-floating_feature keyshot2 ^  
-floating_license_server 2703@VPC_Endpoint_DNS_Name ^  
-script "C:\Program Files\Python311\Lib\site-packages\deadline\keyshot_adaptor  
\KeyShotClient\keyshot_handler.py"
```

Respons harus berisi yang berikut ini tanpa pesan kesalahan:

```
Connecting to floating license server
```

Example — Pengecoran Nuke

Atur variabel lingkungan foundry_LICENSE ke:

```
6101@VPC_Endpoint_DNS_Name
```

Untuk menguji bahwa lisensi berfungsi dengan baik, Anda dapat menjalankan Nuke di terminal:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example — Pergeseran Merah

Atur variabel lingkungan redshift_LICENSE ke:

```
7054@VPC_Endpoint_DNS_Name
```

Setelah Anda membuat variabel lingkungan, Anda harus dapat membuat gambar menggunakan baris perintah yang mirip dengan yang ini:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^  
C:\demo\proxy\RS_Proxy_Demo.rs ^  
-oip C:\demo\proxy\images
```

Example — SidFX Houdini, Mantra, dan Karma

Jalankan perintah berikut:

```
/opt/hfs19.5.640/bin/hserver -S  
"http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://  
VPC_Endpoint_DNS_Name:1717;"
```

Untuk menguji bahwa lisensi berfungsi dengan baik, Anda dapat merender adegan Houdini melalui perintah ini:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example – V-Ray

Atur variabel lingkungan VRAY_AUTH_CLIENT_SETTINGS ke:

```
licset://VPC_Endpoint_DNS_Name:30304
```

Atur variabel lingkungan VRAY_AUTH_CLIENT_FILE_PATH ke:

```
/null
```

Untuk menguji bahwa lisensi berfungsi dengan benar, Anda dapat merender gambar V-Ray menggunakan perintah yang mirip dengan yang ini:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Langkah 4: Hapus titik akhir lisensi

Saat menghapus armada yang dikelola pelanggan, ingatlah untuk menghapus titik akhir lisensi Anda. Jika Anda tidak menghapus titik akhir lisensi, Anda akan terus dikenakan biaya AWS PrivateLink tetap

[Anda dapat menghapus titik akhir lisensi dari dasbor di konsol Deadline Cloud.](#)

1. Dari panel navigasi kiri, pilih titik akhir Lisensi.
2. Pilih titik akhir yang ingin Anda hapus dan pilih hapus, lalu pilih hapus lagi untuk mengonfirmasi.

AWS Batas Waktu Pemantauan Cloud

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja AWS Deadline Cloud (Deadline Cloud) dan solusi Anda AWS . Kumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Sebelum Anda mulai memantau Deadline Cloud, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapakah yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

AWS dan Deadline Cloud menyediakan alat yang dapat Anda gunakan untuk memantau sumber daya Anda dan menanggapi potensi insiden. Beberapa alat ini melakukan pemantauan untuk Anda, beberapa alat memerlukan intervensi manual. Anda harus mengotomatiskan tugas pemantauan sebanyak mungkin.

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat [Panduan CloudWatch Pengguna Amazon](#).

Deadline Cloud memiliki tiga CloudWatch metrik.

- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari EC2 instans Amazon CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat [Panduan Pengguna Amazon CloudWatch Logs](#).

- Amazon EventBridge dapat digunakan untuk 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 mana yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Untuk informasi selengkapnya, lihat [Panduan EventBridge Pengguna Amazon](#).
- AWS CloudTrail menangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat [Panduan Pengguna AWS CloudTrail](#).

Topik

- [Pencatatan panggilan Deadline Cloud API menggunakan AWS CloudTrail](#)
- [Pemantauan CloudWatch dengan](#)
- [Mengelola peristiwa Deadline Cloud menggunakan Amazon EventBridge](#)

Pencatatan panggilan Deadline Cloud API menggunakan AWS CloudTrail

Deadline Cloud terintegrasi dengan [AWS CloudTrail](#), layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk Deadline Cloud sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari Deadline Cloud konsol dan panggilan kode ke operasi Deadline Cloud API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat Deadline Cloud, alamat IP dari mana permintaan 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 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 Menyimpan 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](#).

Deadline Cloud peristiwa data di CloudTrail

[Peristiwa data](#) memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya (misalnya, membaca atau menulis ke objek Amazon S3). Ini juga dikenal sebagai operasi bidang data. Peristiwa data seringkali merupakan aktivitas volume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data. Riwayat CloudTrail peristiwa tidak merekam peristiwa data.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#).

Anda dapat mencatat peristiwa data untuk jenis Deadline Cloud sumber daya menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Untuk informasi selengkapnya tentang cara mencatat peristiwa data, lihat [Mencatat peristiwa data dengan AWS Management Console](#) dan [Logging peristiwa data dengan AWS Command Line Interface](#) di Panduan AWS CloudTrail Pengguna.

Tabel berikut mencantumkan jenis Deadline Cloud sumber daya yang dapat Anda log peristiwa data. Kolom tipe peristiwa data (konsol) menunjukkan nilai yang akan dipilih dari daftar tipe peristiwa Data di CloudTrail konsol. Kolom nilai resources.type menunjukkan **resources.type** nilai, yang akan Anda tentukan saat mengonfigurasi penyeleksi acara lanjutan menggunakan or. AWS CLI CloudTrail APIs CloudTrailKolom Data yang APIs dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk CloudTrail
Armada Batas Waktu	AWS::Deadline::Fleet	• SearchWorkers
Antrian Batas Waktu	AWS::Deadline::Fleet	• SearchJobs
Batas waktu Pekerja	AWS::Deadline::Worker	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk CloudTrail
Deadline Job	AWS::Deadline::Job	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada `eventNameReadOnly`, dan `resources.ARN` bidang untuk mencatat hanya peristiwa yang penting bagi Anda. Untuk informasi selengkapnya tentang bidang ini, lihat [AdvancedFieldSelector](#) di Referensi API AWS CloudTrail .

Deadline Cloud acara manajemen di CloudTrail

[Acara manajemen](#) memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di Akun AWS. Ini juga dikenal sebagai operasi pesawat kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

AWS Deadline Cloud mencatat operasi bidang Deadline Cloud kontrol berikut ke CloudTrail sebagai peristiwa manajemen.

- [associate-member-to-farm](#)
- [associate-member-to-fleet](#)
- [associate-member-to-job](#)
- [associate-member-to-queue](#)
- [assume-fleet-role-for-baca](#)
- [assume-fleet-role-for-pekerja](#)
- [assume-queue-role-for-baca](#)
- [assume-queue-role-for-pengguna](#)
- [assume-queue-role-for-pekerja](#)
- [buat-anggaran](#)
- [buat-pertanian](#)
- [membuat-armada](#)
- [create-license-endpoint](#)
- [buat-batas](#)
- [buat-monitor](#)
- [buat-antrian](#)
- [create-queue-environment](#)
- [create-queue-fleet-association](#)
- [create-queue-limit-association](#)
- [create-storage-profile](#)
- [menciptakan-pekerja](#)
- [hapus-anggaran](#)
- [hapus-pertanian](#)

- [hapus-armada](#)
- [delete-license-endpoint](#)
- [hapus-batas](#)
- [delete-metered-product](#)
- [hapus-monitor](#)
- [hapus-antrian](#)
- [delete-queue-environment](#)
- [delete-queue-fleet-association](#)
- [delete-queue-limit-association](#)
- [delete-storage-profile](#)
- [hapus-pekerja](#)
- [disassociate-member-from-farm](#)
- [disassociate-member-from-fleet](#)
- [disassociate-member-from-job](#)
- [disassociate-member-from-queue](#)
- [get-application-version](#)
- [dapatkan-anggaran](#)
- [dapatkan-pertanian](#)
- [get-feature-map](#)
- [dapatkan-armada](#)
- [get-license-endpoint](#)
- [dapatkan-batas](#)
- [dapatkan-monitor](#)
- [get-antrian](#)
- [get-queue-environment](#)
- [get-queue-fleet-association](#)
- [get-queue-limit-association](#)
- [get-sessions-statistics-aggregation](#)
- [get-storage-profile](#)
- [get-storage-profile-for-antrian](#)

- [list-available-metered-products](#)
- [daftar-anggaran](#)
- [list-farm-members](#)
- [daftar-peternakan](#)
- [list-fleet-members](#)
- [daftar-armada](#)
- [list-job-members](#)
- [list-license-endpoints](#)
- [daftar-batas](#)
- [list-metered-products](#)
- [daftar-monitor](#)
- [list-queue-environments](#)
- [list-queue-fleet-associations](#)
- [list-queue-limit-associations](#)
- [list-queue-members](#)
- [daftar-antrian](#)
- [list-storage-profiles](#)
- [list-storage-profiles-for-antrian](#)
- [list-tags-for-resource](#)
- [put-metered-product](#)
- [start-sessions-statistics-aggregation](#)
- [tag-sumber daya](#)
- [untag-sumber daya](#)
- [pembaruan-anggaran](#)
- [perbaruan-pertanian](#)
- [perbaruan-armada](#)
- [batas pembaruan-](#)
- [pembaruan-monitor](#)
- [antrian pembaruan-](#)
- [update-queue-environment](#)

- [update-queue-fleet-association](#)
- [update-queue-limit-association](#)
- [update-storage-profile](#)
- [pembaruan-pekerja](#)

Deadline Cloud contoh acara

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 CreateFarm operasi.

```
{
  "eventVersion": "0",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
    "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-Session",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE-accessKeyId",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EXAMPLE-PrincipalID",
        "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
        "accountId": "111122223333",
        "userName": "EXAMPLE-UserName"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-03-08T23:25:49Z"
      }
    }
  },
  "eventTime": "2021-03-08T23:25:49Z",
  "eventSource": "deadline.amazonaws.com",
  "eventName": "CreateFarm",
```

```
{
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "EXAMPLE-userAgent",
  "requestParameters": {
    "displayName": "example-farm",
    "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
    "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
    "description": "example-description",
    "tags": {
      "purpose_1": "e2e"
      "purpose_2": "tag_test"
    }
  },
  "responseElements": {
    "farmId": "EXAMPLE-farmID"
  },
  "requestID": "EXAMPLE-requestID",
  "eventID": "EXAMPLE-eventID",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
  "eventCategory": "Management",
}
```

Contoh JSON menunjukkan Wilayah AWS, alamat IP, dan "requestParameters" lainnya seperti "" dan displayName "kmsKeyArn" yang dapat membantu Anda mengidentifikasi acara.

Untuk informasi tentang konten CloudTrail rekaman, lihat [konten CloudTrail rekaman](#) di Panduan AWS CloudTrail Pengguna.

Pemantauan CloudWatch dengan

Amazon CloudWatch (CloudWatch) mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. Anda dapat membuka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/> untuk melihat dan memfilter metrik Deadline Cloud.

Statistik ini disimpan selama 15 bulan sehingga Anda dapat mengakses informasi historis untuk mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau

mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat [Panduan CloudWatch Pengguna Amazon](#).

Deadline Cloud memiliki dua jenis log — log tugas dan log pekerja. Log tugas adalah saat Anda menjalankan log eksekusi sebagai skrip atau saat DCC berjalan. Log tugas mungkin menampilkan peristiwa seperti pemuatan aset, rendering ubin, atau tekstur yang tidak ditemukan.

Log pekerja menunjukkan proses agen pekerja. Ini mungkin termasuk hal-hal seperti ketika agen pekerja memulai, mendaftarkan dirinya sendiri, melaporkan kemajuan, memuat konfigurasi, atau menyelesaikan tugas.

Namespace untuk log ini adalah. `/aws/deadline/*`

Untuk Deadline Cloud, pekerja mengunggah log ini ke CloudWatch Log. Secara default, log tidak pernah kedaluwarsa. Jika suatu pekerjaan menghasilkan volume data yang tinggi, Anda dapat dikenakan biaya tambahan. Untuk informasi selengkapnya, lihat [CloudWatch harga Amazon](#).

Anda dapat menyesuaikan kebijakan penyimpanan untuk setiap grup log. Retensi yang lebih pendek menghilangkan log lama dan dapat membantu mengurangi biaya penyimpanan. Untuk menyimpan log, Anda dapat mengarsipkannya ke Amazon Simple Storage Service sebelum menghapus log. Untuk informasi selengkapnya, lihat [Mengekspor data log ke Amazon S3 menggunakan konsol di panduan CloudWatch pengguna Amazon](#).

Note

CloudWatch pembacaan log dibatasi oleh AWS. Jika Anda berencana untuk bergabung dengan banyak artis, kami sarankan Anda menghubungi dukungan AWS pelanggan dan meminta kenaikan `GetLogEvents` kuota. CloudWatch Selain itu, kami sarankan Anda menutup portal tailing log saat Anda tidak men-debug.

Untuk informasi selengkapnya, lihat [Kuota CloudWatch log](#) di panduan CloudWatch pengguna Amazon.

CloudWatch metrik

Deadline Cloud mengirimkan metrik ke Amazon. CloudWatch Anda dapat menggunakan, API AWS Management Console AWS CLI, atau API untuk membuat daftar metrik yang dikirimkan oleh Deadline Cloud. CloudWatch Secara default, setiap titik data mencakup 1 menit yang mengikuti

waktu mulai aktivitas. Untuk informasi tentang cara melihat metrik yang tersedia menggunakan AWS Management Console atau metrik AWS CLI, lihat [Melihat metrik yang tersedia](#) di CloudWatch Panduan Pengguna Amazon.

Metrik armada yang dikelola pelanggan

AWS/DeadlineCloudNamespace berisi metrik berikut untuk armada yang dikelola pelanggan Anda:

Metrik	Deskripsi	Unit
RecommendedFleetSize	Jumlah pekerja yang direkomendasikan Deadline Cloud yang Anda gunakan untuk memproses pekerjaan. Anda dapat menggunakan metrik ini untuk memperluas atau mengontrak jumlah pekerja dari armada Anda.	Hitungan
UnhealthyWorkerCount	Jumlah pekerja yang ditugaskan untuk memproses pekerjaan yang tidak sehat.	Hitungan

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik armada yang dikelola pelanggan:

Dimensi	Deskripsi
FarmId	Dimensi ini menyaring data yang Anda minta ke peternakan yang ditentukan.
FleetId	Dimensi ini menyaring data yang Anda minta ke armada pekerja yang ditentukan.

Metrik batas sumber daya

AWS/DeadlineCloudNamespace berisi metrik berikut untuk batas sumber daya:

Metrik	Deskripsi	Unit
CurrentCount	Jumlah sumber daya yang dimodelkan oleh batas ini digunakan.	Hitungan
MaxCount	Jumlah maksimum sumber daya yang dimodelkan oleh batas ini. Jika Anda menyetel maxCount nilai ke -1 menggunakan API, Deadline Cloud tidak memancarkan metrik. MaxCount	Hitungan

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik batas bersamaan:

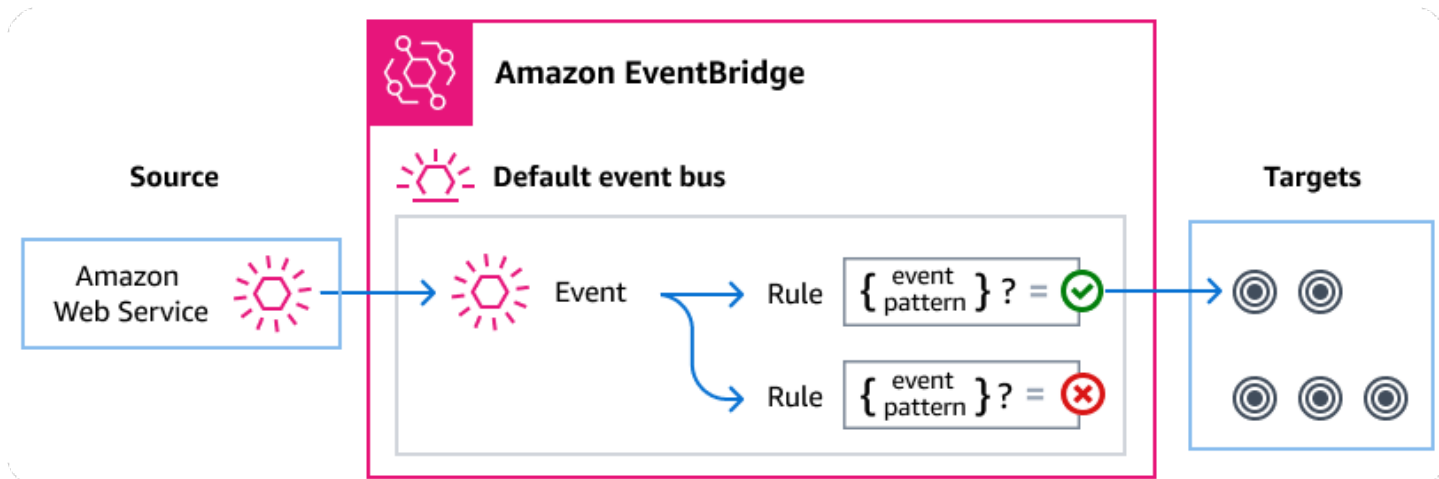
Dimensi	Deskripsi
FarmId	Dimensi ini menyaring data yang Anda minta ke peternakan yang ditentukan.
LimitId	Dimensi ini menyaring data yang Anda minta ke batas yang ditentukan.

Mengelola peristiwa Deadline Cloud menggunakan Amazon EventBridge

Amazon EventBridge adalah layanan tanpa server yang menggunakan peristiwa untuk menghubungkan komponen aplikasi bersama-sama, sehingga memudahkan Anda untuk membangun aplikasi berbasis peristiwa yang dapat diskalakan. Arsitektur berbasis peristiwa adalah gaya membangun sistem perangkat lunak yang digabungkan secara longgar yang bekerja sama dengan memancarkan dan menanggapi peristiwa. Peristiwa mewakili perubahan dalam sumber daya atau lingkungan.

Begini cara kerjanya:

Seperti banyak AWS layanan, Deadline Cloud menghasilkan dan mengirim acara ke bus acara EventBridge default. (Bus acara default secara otomatis disediakan di setiap AWS akun.) Bus acara adalah router yang menerima acara dan mengirimkannya ke nol atau lebih tujuan, atau target. Aturan yang Anda tentukan untuk bus acara mengevaluasi peristiwa saat mereka tiba. Setiap aturan memeriksa apakah suatu peristiwa cocok dengan pola acara aturan. Jika acara tidak cocok, bus acara mengirimkan acara ke target yang ditentukan.



Topik

- [Acara Batas Waktu Cloud](#)
- [Menyampaikan event Deadline Cloud menggunakan aturan EventBridge](#)
- [Referensi detail acara Batas waktu Cloud](#)

Acara Batas Waktu Cloud

Deadline Cloud mengirimkan peristiwa berikut ke bus EventBridge acara default secara otomatis. Peristiwa yang cocok dengan pola acara aturan dikirimkan ke target yang ditentukan [berdasarkan upaya terbaik](#). Acara mungkin dikirim keluar dari pesanan.

Untuk informasi selengkapnya, lihat [EventBridge peristiwa](#) di Panduan Amazon EventBridge Pengguna.

Jenis detail acara	Deskripsi
Ambang Batas Anggaran Tercapai	Dikirim ketika antrian mencapai persentase dari anggaran yang ditetapkan.

Jenis detail acara	Deskripsi
Perubahan Status Siklus Hidup Job	Dikirim ketika ada perubahan pada status siklus hidup suatu pekerjaan.
Perubahan Status Job Run	Dikirim ketika status keseluruhan tugas dalam pekerjaan berubah.
Langkah Perubahan Status Siklus Hidup	Dikirim ketika ada perubahan status siklus hidup dari langkah dalam pekerjaan.
Langkah Jalankan Perubahan Status	Dikirim ketika status keseluruhan tugas dalam satu langkah berubah.
Perubahan Status Jalankan Tugas	Dikirim saat status tugas berubah.

Menyampaikan event Deadline Cloud menggunakan aturan EventBridge

Agar bus acara EventBridge default mengirim peristiwa Deadline Cloud ke target, Anda harus membuat aturan. Setiap aturan berisi pola acara, yang EventBridge cocok dengan setiap acara yang diterima di bus acara. Jika data peristiwa cocok dengan pola peristiwa yang ditentukan, EventBridge mengirimkan peristiwa itu ke target aturan.

Untuk petunjuk komprehensif tentang cara membuat aturan bus acara, lihat [Membuat aturan yang bereaksi terhadap peristiwa](#) di Panduan EventBridge Pengguna.

Membuat pola acara yang cocok dengan peristiwa Deadline Cloud

Setiap pola acara adalah objek JSON yang berisi:

- `sourceAtribut` yang mengidentifikasi layanan yang mengirim acara. Untuk acara Deadline Cloud, sumbernya adalah `aws.deadline`.
- (Opsional): `detail-type` Atribut yang berisi array jenis acara yang cocok.
- (Opsional): `detail` Atribut yang berisi data acara lain yang cocok.

Misalnya, pola peristiwa berikut cocok dengan semua peristiwa Perubahan Rekomendasi Ukuran Armada untuk yang ditentukan `farmId` untuk Deadline Cloud:

[illegible]

Untuk informasi selengkapnya tentang penulisan pola acara, lihat [Pola acara](#) di Panduan EventBridge Pengguna.

Referensi detail acara Batas waktu Cloud

Semua peristiwa dari AWS layanan memiliki seperangkat bidang umum yang berisi metadata tentang acara tersebut, seperti AWS layanan yang merupakan sumber acara, waktu acara dibuat, akun dan wilayah tempat acara berlangsung, dan lainnya. Untuk definisi bidang umum ini, lihat [Referensi struktur acara](#) di Panduan Amazon EventBridge Pengguna.

Selain itu, setiap acara memiliki detail bidang yang berisi data khusus untuk peristiwa tertentu. Referensi di bawah ini mendefinisikan bidang detail untuk berbagai peristiwa Deadline Cloud.

Saat menggunakan EventBridge untuk memilih dan mengelola peristiwa Deadline Cloud, penting untuk mengingat hal berikut:

- `sourceBidang` untuk semua acara dari Deadline Cloud diatur ke `aws.deadline.`
- `detail-typeBidang` menentukan jenis acara.

Misalnya, Fleet Size Recommendation Change.

- detailBidang berisi data yang spesifik untuk peristiwa tertentu.

Untuk informasi tentang membuat pola peristiwa yang memungkinkan aturan agar sesuai dengan peristiwa Deadline Cloud, lihat [Pola acara](#) di Amazon EventBridge Panduan Pengguna.

Untuk informasi selengkapnya tentang peristiwa dan cara EventBridge memprosesnya, lihat [Amazon EventBridge peristiwa](#) di Panduan Amazon EventBridge Pengguna.

Topik

- Ambang Batas Anggaran Mencapai acara
- Acara Perubahan Rekomendasi Ukuran Armada

- [Acara Perubahan Status Siklus Hidup Job](#)
- [Acara Perubahan Status Job Run](#)
- [Langkah Siklus Hidup Status Ubah acara](#)
- [Langkah Jalankan Status Ubah acara](#)
- [Acara Ubah Status Jalankan Tugas](#)

Ambang Batas Anggaran Mencapai acara

Anda dapat menggunakan acara Budget Threshold Received untuk memantau persentase anggaran yang telah digunakan. Deadline Cloud mengirimkan peristiwa ketika persentase yang digunakan melewati ambang batas berikut:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

Frekuensi Deadline Cloud mengirimkan peristiwa Budget Threshold Received meningkat saat anggaran mendekati batasnya. Ini memungkinkan Anda untuk memantau anggaran dengan cermat saat mendekati batasnya dan mengambil tindakan untuk mencegah pengeluaran berlebihan. Anda juga dapat menetapkan ambang anggaran Anda sendiri. Deadline Cloud mengirimkan peristiwa saat penggunaan melewati ambang batas kustom Anda.

Jika Anda mengubah jumlah anggaran, waktu berikutnya Deadline Cloud mengirimkan acara Ambang Batas Anggaran Tercapai, hal itu didasarkan pada persentase anggaran saat ini yang telah digunakan. Misalnya, jika Anda menambahkan \$50 ke anggaran \$100 yang telah mencapai batasnya, acara Ambang Batas Anggaran berikutnya menunjukkan bahwa anggaran berada pada 75 persen.

Di bawah ini adalah bidang detail untuk Budget Threshold Reached acara tersebut.

detail-typeBidang source dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Budget Threshold Reached",
  "source": "aws.deadline",
  "account": "111122223333",
```

```

    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
      "farmId": "farm-12345678900000000000000000000000",
      "budgetId": "budget-12345678900000000000000000000000",
      "thresholdInPercent": 0
    }
  }
}

```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Budget Threshold Reached`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara `Deadline Cloud`, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

farmId

Pengidentifikasi pertanian yang berisi pekerjaan.

budgetId

Pengidentifikasi anggaran yang telah mencapai ambang batas.

thresholdInPercent

Persentase anggaran yang telah digunakan.

Acara Perubahan Rekomendasi Ukuran Armada

Saat mengonfigurasi armada untuk menggunakan penskalaan otomatis berbasis peristiwa, `Deadline Cloud` mengirimkan acara yang dapat Anda gunakan untuk mengelola armada Anda. Masing-

masing acara ini berisi informasi tentang ukuran saat ini dan ukuran armada yang diminta. Untuk contoh penggunaan EventBridge acara dan contoh fungsi Lambda untuk menangani acara, lihat Menskalakan [otomatis EC2 armada Amazon Anda dengan fitur rekomendasi skala Deadline Cloud](#).

Acara perubahan rekomendasi ukuran armada dikirim ketika hal berikut terjadi:

- Ketika ukuran armada yang direkomendasikan berubah dan `oldFleetSize` berbeda dari `newFleetSize`.
- Ketika layanan mendeteksi bahwa ukuran armada sebenarnya tidak sesuai dengan ukuran armada yang direkomendasikan. Anda bisa mendapatkan ukuran armada sebenarnya dari `WorkerCount` dalam respons `GetFleet` operasi. Hal ini dapat terjadi ketika EC2 instans Amazon aktif gagal mendaftar sebagai pekerja Deadline Cloud.

Di bawah ini adalah bidang detail untuk `Fleet Size Recommendation Change` acara tersebut.

`detail-type` Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Fleet Size Recommendation Change",
  "source": "aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "aa-example-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "fleetId": "fleet-12345678900000000000000000000000",
    "oldFleetSize": 1,
    "newFleetSize": 5,
  }
}
```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Fleet Size Recommendation Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi pertanian yang berisi pekerjaan.

`fleetId`

Pengidentifikasi armada yang membutuhkan perubahan ukuran.

`oldFleetSize`

Ukuran armada saat ini.

`newFleetSize`

Ukuran baru yang direkomendasikan untuk armada.

Acara Perubahan Status Siklus Hidup Job

Saat Anda membuat atau memperbarui pekerjaan, Deadline Cloud menetapkan status siklus hidup untuk menampilkan status tindakan yang dimulai pengguna terbaru.

Peristiwa perubahan status siklus hidup pekerjaan dikirim untuk setiap perubahan status siklus hidup, termasuk saat pekerjaan dibuat.

Di bawah ini adalah bidang detail untuk `Job Lifecycle Status Change` acara tersebut.

`detail-type` Bidang `source` dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{  
  "version": "0",
```

```
{
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Job Lifecycle Status Change",
  "source": "aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "aa-example-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "queueId": "queue-12345678900000000000000000000000",
    "jobId": "job-12345678900000000000000000000000",
    "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
    "lifecycleStatus": "UPDATE_SUCCEEDED"
  }
}
```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Job Lifecycle Status Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

farmId

Pengidentifikasi pertanian yang berisi pekerjaan.

queueId

Pengidentifikasi antrian yang berisi pekerjaan.

jobId

Pengidentifikasi pekerjaan.

previousLifecycleStatus

Siklus hidup menyatakan bahwa pekerjaan akan pergi. Bidang ini tidak termasuk saat Anda pertama kali mengirimkan pekerjaan.

lifecycleStatus

Siklus hidup menyatakan bahwa pekerjaan sedang masuk.

Acara Perubahan Status Job Run

Pekerjaan terdiri dari banyak tugas. Setiap tugas memiliki status. Status semua tugas digabungkan untuk memberikan status keseluruhan untuk suatu pekerjaan. Untuk informasi selengkapnya, lihat [Status pekerjaan di Deadline Cloud](#) di Panduan Pengguna Cloud AWS Deadline.

Peristiwa perubahan status job run dikirim saat:

- [taskRunStatus](#) Bidang gabungan berubah.
- Pekerjaan tersebut diminta kembali, kecuali pekerjaan tersebut dalam keadaan READY.

Peristiwa perubahan status job run TIDAK dikirim saat:

- Pekerjaan pertama kali dibuat. Untuk memantau pembuatan lowongan kerja, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.
- [taskRunStatusCounts](#) Bidang pekerjaan berubah tetapi status tugas tugas kerja gabungan tidak berubah.

Di bawah ini adalah bidang detail untuk Job Run Status Change acara tersebut.

detail-typeBidang source dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Job Run Status Change",
  "source": "aws.deadline",
  "account": "111122223333",
```

```

    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
      "farmId": "farm-12345678900000000000000000000000",
      "queueId": "queue-12345678900000000000000000000000",
      "jobId": "job-12345678900000000000000000000000",
      "previousTaskRunStatus": "RUNNING",
      "taskRunStatus": "SUCCEEDED",
      "taskRunStatusCounts": {
        "PENDING": 0,
        "READY": 0,
        "RUNNING": 0,
        "ASSIGNED": 0,
        "STARTING": 0,
        "SCHEDULED": 0,
        "INTERRUPTING": 0,
        "SUSPENDED": 0,
        "CANCELED": 0,
        "FAILED": 0,
        "SUCCEEDED": 20,
        "NOT_COMPATIBLE": 0
      }
    }
  }
}

```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Job Run Status Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara `Deadline Cloud`, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi pertanian yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`previousTaskRunStatus`

Task run menyatakan bahwa pekerjaan akan pergi.

`taskRunStatus`

Task run menyatakan bahwa pekerjaan sedang masuk.

`taskRunStatusCounts`

Jumlah tugas pekerjaan di setiap negara bagian.

Langkah Siklus Hidup Status Ubah acara

Saat Anda membuat atau memperbarui peristiwa, Deadline Cloud akan menetapkan status siklus hidup pekerjaan untuk menjelaskan status tindakan yang dimulai pengguna terbaru.

Peristiwa perubahan status siklus hidup langkah dikirim saat:

- Pembaruan langkah dimulai (UPDATE_IN_PROGRESS).
- Pembaruan langkah berhasil diselesaikan (UPDATE_SUCCEEDED).
- Pembaruan langkah gagal (UPDATE_FAILED).

Peristiwa tidak dikirim saat langkah pertama kali dibuat. Untuk memantau pembuatan langkah, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.

Di bawah ini adalah bidang detail untuk Step Lifecycle Status Change acara tersebut.

`detail-type` Bidang source dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Step Lifecycle Status Change",
  "source": "aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "aa-example-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "queueId": "queue-12345678900000000000000000000000",
    "jobId": "job-12345678900000000000000000000000",
    "stepId": "step-12345678900000000000000000000000",
    "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
    "lifecycleStatus": "UPDATE_SUCCEEDED"
  }
}
```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Step Lifecycle Status Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

farmId

Pengidentifikasi pertanian yang berisi pekerjaan.

queueId

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`stepId`

Pengidentifikasi langkah pekerjaan saat ini.

`previousLifecycleStatus`

Siklus hidup menyatakan bahwa langkah akan pergi.

`lifecycleStatus`

Siklus hidup menyatakan bahwa langkah tersebut masuk.

Langkah Jalankan Status Ubah acara

Setiap langkah dalam suatu pekerjaan terdiri dari banyak tugas. Setiap tugas memiliki status. Status tugas digabungkan untuk memberikan status keseluruhan untuk langkah dan pekerjaan.

Peristiwa perubahan status step run dikirim saat:

- [taskRunStatus](#) Perubahan gabungan.
- Langkah tersebut diminta ulang, kecuali langkah itu dalam keadaan READY.

Sebuah acara tidak dikirim ketika:

- Langkah ini pertama kali dibuat. Untuk memantau pembuatan langkah, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.
- Langkah [taskRunStatusCounts](#) berubah tetapi status run tugas langkah gabungan tidak berubah.

Di bawah ini adalah bidang detail untuk Step Run Status Change acara tersebut.

`detail-type` Bidang source dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```

    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
      "farmId": "farm-12345678900000000000000000000000",
      "queueId": "queue-12345678900000000000000000000000",
      "jobId": "job-12345678900000000000000000000000",
      "stepId": "step-12345678900000000000000000000000",
      "previousTaskRunStatus": "RUNNING",
      "taskRunStatus": "SUCCEEDED",
      "taskRunStatusCounts": {
        "PENDING": 0,
        "READY": 0,
        "RUNNING": 0,
        "ASSIGNED": 0,
        "STARTING": 0,
        "SCHEDULED": 0,
        "INTERRUPTING": 0,
        "SUSPENDED": 0,
        "CANCELED": 0,
        "FAILED": 0,
        "SUCCEEDED": 20,
        "NOT_COMPATIBLE": 0
      }
    }
  }
}

```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Step Run Status Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah `aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

`farmId`

Pengidentifikasi pertanian yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`stepId`

Pengidentifikasi langkah pekerjaan saat ini.

`previousTaskRunStatus`

Lari menyatakan bahwa langkahnya akan pergi.

`taskRunStatus`

Jalankan menyatakan bahwa langkah sedang masuk.

`taskRunStatusCounts`

Jumlah tugas langkah di setiap negara bagian.

Acara Ubah Status Jalankan Tugas

[runStatus](#) Bidang diperbarui saat tugas berjalan. Sebuah acara dikirim ketika:

- Status run tugas berubah.
- Tugas tersebut diminta kembali, kecuali tugas dalam keadaan READY.

Sebuah acara tidak dikirim ketika:

- Tugas pertama kali dibuat. Untuk memantau pembuatan tugas, pantau peristiwa Perubahan Status Siklus Hidup Job untuk perubahan.

Di bawah ini adalah bidang detail untuk Task Run Status Change acara tersebut.

detail-typeBidang source dan disertakan di bawah ini karena berisi nilai spesifik untuk peristiwa Deadline Cloud. Untuk definisi bidang metadata lain yang disertakan dalam semua peristiwa, lihat [Referensi struktur acara](#) di Amazon EventBridge Panduan Pengguna.

```
{
  "version": "0",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "detail-type": "Task Run Status Change",
  "source": "aws.aws.deadline",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "aa-example-1",
  "resources": [],
  "detail": {
    "farmId": "farm-12345678900000000000000000000000",
    "queueId": "queue-12345678900000000000000000000000",
    "jobId": "job-12345678900000000000000000000000",
    "stepId": "step-12345678900000000000000000000000",
    "taskId": "task-12345678900000000000000000000000-0",
    "previousRunStatus": "RUNNING",
    "runStatus": "SUCCEEDED"
  }
}
```

detail-type

Mengidentifikasi jenis acara.

Untuk acara ini, nilai ini adalah `Fleet Size Recommendation Change`.

source

Mengidentifikasi layanan yang menghasilkan peristiwa. Untuk acara Deadline Cloud, nilai ini adalah `aws.aws.deadline`.

detail

Objek JSON yang berisi informasi tentang peristiwa. Layanan yang menghasilkan acara menentukan konten bidang ini.

Untuk acara ini, data ini meliputi:

farmId

Pengidentifikasi pertanian yang berisi pekerjaan.

`queueId`

Pengidentifikasi antrian yang berisi pekerjaan.

`jobId`

Pengidentifikasi pekerjaan.

`stepId`

Pengidentifikasi langkah pekerjaan saat ini.

`taskId`

Pengidentifikasi tugas yang sedang berjalan.

`previousRunStatus`

Jalankan menyatakan bahwa tugas akan pergi.

`runStatus`

Status run yang dimasukkan tugas.

Keamanan di Deadline Cloud

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](#) menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program AWS Kepatuhan Program AWS Kepatuhan](#) . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Deadline Cloud, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS 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 Deadline Cloud. Topik berikut menunjukkan cara mengonfigurasi Deadline Cloud untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan Deadline Cloud sumber daya Anda.

Topik

- [Perlindungan data di Deadline Cloud](#)
- [Identity and Access Management di Deadline Cloud](#)
- [Validasi kepatuhan untuk Deadline Cloud](#)
- [Ketahanan di Deadline Cloud](#)
- [Keamanan infrastruktur di Deadline Cloud](#)
- [Analisis konfigurasi dan kerentanan di Deadline Cloud](#)
- [Pencegahan "confused deputy" lintas layanan](#)
- [Akses AWS Deadline Cloud menggunakan endpoint antarmuka \(\)AWS PrivateLink](#)

- [Praktik terbaik keamanan untuk Deadline Cloud](#)

Perlindungan data di Deadline Cloud

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS Deadline Cloud. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti

bidang Nama. Ini termasuk saat Anda bekerja dengan Deadline Cloud atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Data yang dimasukkan ke dalam bidang nama dalam templat Deadline Cloud pekerjaan juga dapat dimasukkan dalam log penagihan atau diagnostik dan tidak boleh berisi informasi rahasia atau sensitif.

Topik

- [Enkripsi diam](#)
- [Enkripsi bergerak](#)
- [Manajemen kunci](#)
- [Privasi lalu lintas antar jaringan](#)
- [Menyisih](#)

Enkripsi diam

AWS Deadline Cloud melindungi data sensitif dengan mengenkripsinya saat istirahat menggunakan kunci enkripsi yang disimpan di [AWS Key Management Service \(AWS KMS\)](#). Enkripsi saat istirahat tersedia di semua Wilayah AWS tempat Deadline Cloud yang tersedia.

Mengenkripsi data berarti data sensitif yang disimpan pada disk tidak dapat dibaca oleh pengguna atau aplikasi tanpa kunci yang valid. Hanya pihak dengan kunci terkelola yang valid yang dapat mendekripsi data.

Untuk informasi tentang cara Deadline Cloud penggunaan AWS KMS untuk mengenkripsi data saat istirahat, lihat. [Manajemen kunci](#)

Enkripsi bergerak

Untuk data dalam perjalanan, AWS Deadline Cloud gunakan Transport Layer Security (TLS) 1.2 atau 1.3 untuk mengenkripsi data yang dikirim antara layanan dan pekerja. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3. Selain itu, jika Anda menggunakan virtual private cloud (VPC), Anda

dapat menggunakannya AWS PrivateLink untuk membuat koneksi pribadi antara VPC dan VPC Anda. Deadline Cloud

Manajemen kunci

Saat membuat peternakan baru, Anda dapat memilih salah satu kunci berikut untuk mengenkripsi data pertanian Anda:

- AWS kunci KMS yang dimiliki — Jenis enkripsi default jika Anda tidak menentukan kunci saat membuat peternakan. Kunci KMS dimiliki oleh AWS Deadline Cloud. Anda tidak dapat melihat, mengelola, atau menggunakan kunci AWS yang dimiliki. Namun, Anda tidak perlu mengambil tindakan apa pun untuk melindungi kunci yang mengenkripsi data Anda. Untuk informasi selengkapnya, lihat [kunci yang AWS dimiliki](#) di panduan AWS Key Management Service pengembang.
- Kunci KMS yang dikelola pelanggan — Anda menentukan kunci yang dikelola pelanggan saat membuat peternakan. Semua konten di dalam peternakan dienkripsi dengan kunci KMS. Kunci disimpan di akun Anda dan dibuat, dimiliki, dan dikelola oleh Anda dan AWS KMS dikenakan biaya. Anda memiliki kontrol penuh atas tombol KMS. Anda dapat melakukan tugas-tugas seperti:
 - Menetapkan dan memelihara kebijakan utama
 - Menetapkan dan memelihara kebijakan dan hibah IAM
 - Mengaktifkan dan menonaktifkan kebijakan utama
 - Menambahkan tanda
 - Membuat alias kunci

Anda tidak dapat memutar kunci milik pelanggan secara manual yang digunakan dengan Deadline Cloud peternakan. Rotasi otomatis tombol didukung.

Untuk informasi selengkapnya, lihat [Kunci milik pelanggan](#) di Panduan AWS Key Management Service Pengembang.

Untuk membuat kunci terkelola pelanggan, ikuti langkah-langkah untuk [Membuat kunci terkelola pelanggan simetris](#) di Panduan AWS Key Management Service Pengembang.

Bagaimana Deadline Cloud menggunakan AWS KMS hibah

Deadline Cloud membutuhkan [hibah](#) untuk menggunakan kunci yang dikelola pelanggan Anda. Saat Anda membuat peternakan yang dienkripsi dengan kunci yang dikelola pelanggan, Deadline Cloud

buat hibah atas nama Anda dengan mengirimkan [CreateGrant](#) permintaan untuk mendapatkan akses AWS KMS ke kunci KMS yang Anda tentukan.

Deadline Cloud menggunakan beberapa hibah. Setiap hibah digunakan oleh bagian yang berbeda Deadline Cloud yang perlu mengenkripsi atau mendekripsi data Anda. Deadline Cloud juga menggunakan hibah untuk memungkinkan akses ke AWS layanan lain yang digunakan untuk menyimpan data atas nama Anda, seperti Amazon Simple Storage Service, Amazon Elastic Block Store, atau OpenSearch.

Hibah yang memungkinkan Deadline Cloud untuk mengelola mesin dalam armada yang dikelola layanan mencakup nomor Deadline Cloud akun dan peran dalam `GranteePrincipal` alih-alih prinsip layanan. Meskipun tidak khas, ini diperlukan untuk mengenkripsi volume Amazon EBS untuk pekerja dalam armada yang dikelola layanan menggunakan kunci KMS terkelola pelanggan yang ditentukan untuk pertanian.

Kebijakan kunci yang dikelola pelanggan

Kebijakan utama mengontrol akses ke kunci yang dikelola pelanggan Anda. Setiap kunci harus memiliki persis satu kebijakan kunci yang berisi pernyataan yang menentukan siapa yang dapat menggunakan kunci dan bagaimana mereka dapat menggunakannya. Saat membuat kunci terkelola pelanggan, Anda dapat menentukan kebijakan kunci. Untuk informasi selengkapnya, lihat [Mengelola akses ke kunci terkelola pelanggan](#) di Panduan AWS Key Management Service Pengembang.

Kebijakan IAM minimal untuk CreateFarm

Untuk menggunakan kunci terkelola pelanggan Anda untuk membuat farm menggunakan konsol atau operasi [CreateFarm](#) API, operasi AWS KMS API berikut harus diizinkan:

- [kms:CreateGrant](#)— Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses konsol ke AWS KMS kunci tertentu. Untuk informasi selengkapnya, lihat [Menggunakan hibah](#) di panduan AWS Key Management Service pengembang.
- [kms:Decrypt](#)— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- [kms:DescribeKey](#)— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.
- [kms:GenerateDataKey](#)— Memungkinkan Deadline Cloud untuk mengenkripsi data menggunakan kunci data yang unik.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi. `CreateFarm`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DeadlineCreateGrants",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/1234567890abcdef0",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.us-west-2.amazonaws.com"
        }
      }
    }
  ]
}
```

Kebijakan IAM minimal untuk operasi hanya-baca

Untuk menggunakan kunci yang dikelola pelanggan Anda untuk Deadline Cloud operasi hanya-baca, seperti mendapatkan informasi tentang peternakan, antrian, dan armada. Operasi AWS KMS API berikut harus diizinkan:

- [kms:Decrypt](#)— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- [kms:DescribeKey](#)— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi hanya-baca.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DeadlineReadOnly",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.us-west-2.amazonaws.com"
        }
      }
    }
  ]
}
```

Kebijakan IAM minimal untuk operasi baca-tulis

Untuk menggunakan kunci terkelola pelanggan Anda untuk Deadline Cloud operasi baca-tulis, seperti membuat dan memperbarui peternakan, antrian, dan armada. Operasi AWS KMS API berikut harus diizinkan:

- [kms:Decrypt](#)— Memungkinkan Deadline Cloud untuk mendekripsi data di peternakan.
- [kms:DescribeKey](#)— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Deadline Cloud memvalidasi kunci.
- [kms:GenerateDataKey](#)— Memungkinkan Deadline Cloud untuk mengenkripsi data menggunakan kunci data yang unik.

Pernyataan kebijakan berikut memberikan izin yang diperlukan untuk operasi. CreateFarm

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DeadlineReadWrite",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.us-west-2.amazonaws.com"
        }
      }
    }
  ]
}
```

Memantau kunci enkripsi Anda

Saat menggunakan kunci terkelola AWS KMS pelanggan dengan Deadline Cloud peternakan, Anda dapat menggunakan [AWS CloudTrail](#) atau [Amazon CloudWatch Logs](#) untuk melacak permintaan yang Deadline Cloud dikirim AWS KMS.

CloudTrail acara untuk hibah

Contoh CloudTrail peristiwa berikut terjadi ketika hibah dibuat, biasanya ketika Anda memanggil `CreateFarm`, `CreateMonitor`, atau `CreateFleet` operasi.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:SampleUser01",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/SampleUser01",
```

```

    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE",
        "arn": "arn:aws::iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2024-04-23T02:05:26Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "deadline.amazonaws.com"
  },
  "eventTime": "2024-04-23T02:05:35Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "deadline.amazonaws.com",
  "userAgent": "deadline.amazonaws.com",
  "requestParameters": {
    "operations": [
      "CreateGrant",
      "Decrypt",
      "DescribeKey",
      "Encrypt",
      "GenerateDataKey"
    ],
    "constraints": {
      "encryptionContextSubset": {
        "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
        "aws:deadline:accountId": "111122223333"
      }
    },
    "granteePrincipal": "deadline.amazonaws.com",
    "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "retiringPrincipal": "deadline.amazonaws.com"
  },
  "responseElements": {

```

```

    "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
    "keyId": "arn:aws::kms:us-west-2:111122223333: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-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}

```

CloudTrail acara untuk dekripsi

Contoh CloudTrail peristiwa berikut terjadi ketika mendekripsi nilai menggunakan kunci KMS yang dikelola pelanggan.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:SampleUser01",
    "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE",
        "arn": "arn:aws::iam::111122223333:role/SampleRole",
        "accountId": "111122223333",
        "userName": "SampleRole"
      },
      "webIdFederationData": {},
    }
  },
  "webIdFederationData": {},
}

```

```

        "attributes": {
            "creationDate": "2024-04-23T18:46:51Z",
            "mfaAuthenticated": "false"
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEEaaqNOTREALaGTESTONLY  
+p/5H+EuKd4Q=="
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-  
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}

```

CloudTrail acara untuk enkripsi

Contoh CloudTrail peristiwa berikut terjadi ketika mengenkripsi nilai menggunakan kunci KMS yang dikelola pelanggan.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:SampleUser01",
    "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE",
        "arn": "arn:aws::iam::111122223333:role/SampleRole",
        "accountId": "111122223333",
        "userName": "SampleRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2024-04-23T18:46:51Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "deadline.amazonaws.com"
  },
  "eventTime": "2024-04-23T18:52:40Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "deadline.amazonaws.com",
  "userAgent": "deadline.amazonaws.com",
  "requestParameters": {
    "numberOfBytes": 32,
    "encryptionContext": {
      "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
      "aws:deadline:accountId": "111122223333",
      "aws-crypto-public-key": "AotL+SAMPLEVALUEi0MEXAMPLEaaqNOTREALaGTESTONLY+p/5H+EuKd4Q=="
    }
  },
}
```

```

    "keyId": "arn:aws::kms:us-west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}

```

Menghapus kunci KMS yang dikelola pelanggan

Menghapus kunci KMS yang dikelola pelanggan di AWS Key Management Service (AWS KMS) bersifat merusak dan berpotensi berbahaya. Ini secara permanen menghapus materi kunci dan semua metadata yang terkait dengan kunci. Setelah kunci KMS yang dikelola pelanggan dihapus, Anda tidak dapat lagi mendekripsi data yang dienkripsi oleh kunci itu. Ini berarti bahwa data menjadi tidak dapat dipulihkan.

Inilah sebabnya mengapa AWS KMS memberi pelanggan masa tunggu hingga 30 hari sebelum menghapus kunci KMS. Masa tunggu default adalah 30 hari.

Tentang masa tunggu

Karena menghapus kunci KMS yang dikelola pelanggan merusak dan berpotensi berbahaya, kami mengharuskan Anda menetapkan masa tunggu 7-30 hari. Masa tunggu default adalah 30 hari.

Namun, masa tunggu sebenarnya mungkin hingga 24 jam lebih lama dari periode yang Anda jadwalkan. Untuk mendapatkan tanggal dan waktu aktual ketika kunci akan dihapus, gunakan [DescribeKey](#) operasi. Anda juga dapat melihat tanggal penghapusan kunci yang dijadwalkan di [AWS KMS konsol](#) pada halaman detail kunci, di bagian Konfigurasi umum. Perhatikan zona waktu.

Selama masa tunggu, status dan status kunci yang dikelola pelanggan adalah Penghapusan tertunda.

- [Kunci KMS yang dikelola pelanggan yang tertunda penghapusan tidak dapat digunakan dalam operasi kriptografi apa pun.](#)
- AWS KMS tidak [memutar kunci dukungan kunci](#) KMS yang dikelola pelanggan yang sedang menunggu penghapusan.

Untuk informasi selengkapnya tentang menghapus kunci KMS yang dikelola pelanggan, lihat [Menghapus kunci master pelanggan](#) di Panduan Pengembang AWS Key Management Service .

Privasi lalu lintas antar jaringan

AWS Deadline Cloud mendukung Amazon Virtual Private Cloud (Amazon VPC) untuk mengamankan koneksi. Amazon VPC menyediakan fitur yang dapat Anda gunakan untuk meningkatkan dan memantau keamanan virtual private cloud (VPC) Anda.

Anda dapat menyiapkan armada yang dikelola pelanggan (CMF) dengan instans Amazon Elastic Compute Cloud (Amazon EC2) yang berjalan di dalam VPC. Dengan menerapkan titik akhir VPC Amazon untuk AWS PrivateLink digunakan, lalu lintas antar pekerja di CMF Anda dan Deadline Cloud titik akhir tetap berada dalam VPC Anda. Selanjutnya, Anda dapat mengonfigurasi VPC Anda untuk membatasi akses internet ke instans Anda.

Dalam armada yang dikelola layanan, pekerja tidak dapat dijangkau dari internet, tetapi mereka memiliki akses internet dan terhubung ke layanan melalui internet. Deadline Cloud

Menyisih

AWS Deadline Cloud mengumpulkan informasi operasional tertentu untuk membantu kami mengembangkan dan meningkatkan Deadline Cloud. Data yang dikumpulkan mencakup hal-hal seperti ID AWS akun dan ID pengguna Anda, sehingga kami dapat mengidentifikasi Anda dengan benar jika Anda memiliki masalah dengan Deadline Cloud. Kami juga mengumpulkan informasi Deadline Cloud spesifik, seperti Resource IDs (FarmId atau QueueID bila berlaku), nama produk (misalnya,, JobAttachments WorkerAgent, dan lainnya) dan versi produk.

Anda dapat memilih untuk memilih keluar dari pengumpulan data ini menggunakan konfigurasi aplikasi. Setiap komputer yang berinteraksi dengan Deadline Cloud, baik workstation klien dan pekerja armada, perlu memilih keluar secara terpisah.

Deadline Cloud monitor - desktop

Deadline Cloud monitor - desktop mengumpulkan informasi operasional, seperti ketika crash terjadi dan ketika aplikasi dibuka, untuk membantu kami mengetahui kapan Anda mengalami masalah dengan aplikasi. Untuk memilih keluar dari pengumpulan informasi operasional ini, buka halaman pengaturan dan hapus Aktifkan pengumpulan data untuk mengukur kinerja Deadline Cloud Monitor.

Setelah Anda memilih keluar, monitor desktop tidak lagi mengirimkan data operasional. Setiap data yang dikumpulkan sebelumnya disimpan dan masih dapat digunakan untuk meningkatkan layanan. Untuk informasi selengkapnya, lihat [FAQ Privasi Data](#).

AWS Deadline Cloud CLI dan Alat

AWS Deadline Cloud CLI, pengirim, dan agen pekerja semuanya mengumpulkan informasi operasional seperti kapan crash terjadi dan kapan pekerjaan dikirimkan untuk membantu kami mengetahui kapan Anda mengalami masalah dengan aplikasi ini. Untuk memilih keluar dari pengumpulan informasi operasional ini, gunakan salah satu metode berikut:

- Di terminal, masukkan **deadline config set telemetry.opt_out true**.

Ini akan memilih keluar dari CLI, pengirim, dan agen pekerja saat berjalan sebagai pengguna saat ini.

- Saat menginstal agen Deadline Cloud pekerja, tambahkan argumen baris **--telemetry-opt-out** perintah. Misalnya, **./install.sh --farm-id \$FARM_ID --fleet-id \$FLEET_ID --telemetry-opt-out**.
- Sebelum menjalankan agen pekerja, CLI, atau submitter, tetapkan variabel lingkungan: **DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true**

Setelah Anda memilih keluar, Deadline Cloud alat tidak lagi mengirim data operasional. Setiap data yang dikumpulkan sebelumnya disimpan dan masih dapat digunakan untuk meningkatkan layanan. Untuk informasi selengkapnya, lihat [FAQ Privasi Data](#).

Identity and Access Management di Deadline Cloud

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 diberi wewenang (memiliki izin) untuk menggunakan sumber daya Deadline Cloud. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana Deadline Cloud bekerja dengan IAM](#)
- [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)
- [AWS kebijakan terkelola untuk Deadline Cloud](#)
- [Pemecahan Masalah AWS Batas Waktu Identitas dan akses Cloud](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Deadline Cloud.

Pengguna layanan — Jika Anda menggunakan layanan Deadline Cloud untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Deadline Cloud 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 Deadline Cloud, lihat [Pemecahan Masalah AWS Batas Waktu Identitas dan akses Cloud](#).

Administrator layanan — Jika Anda bertanggung jawab atas sumber daya Deadline Cloud di perusahaan Anda, Anda mungkin memiliki akses penuh ke Deadline Cloud. Tugas Anda adalah menentukan fitur dan sumber daya Deadline Cloud 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 Deadline Cloud, lihat. [Bagaimana Deadline Cloud bekerja dengan IAM](#)

Administrator IAM - Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Deadline Cloud. Untuk melihat contoh Kebijakan berbasis identitas Cloud Batas waktu yang dapat Anda gunakan di IAM, lihat. [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensial 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 kredensial 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 kredensial 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 kredensi 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 kredensial 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 instans yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensial 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 [Ikhtisar Access Control List \(ACL\)](#) 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 Deadline Cloud bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Deadline Cloud, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Deadline Cloud.

Fitur IAM yang dapat Anda gunakan dengan AWS Deadline Cloud

Fitur IAM	Dukungan Batas Waktu Cloud
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya

Fitur IAM	Dukungan Batas Waktu Cloud
ACLs	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS kerja Deadline Cloud dan lainnya dengan sebagian besar fitur IAM, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

Kebijakan berbasis identitas untuk Deadline Cloud

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 Deadline Cloud

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)

Kebijakan berbasis sumber daya dalam Deadline Cloud

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 Deadline Cloud

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 Cloud Deadline, lihat [Tindakan yang ditentukan oleh AWS Deadline Cloud](#) di Referensi Otorisasi Layanan.

Tindakan kebijakan di Deadline Cloud menggunakan awalan berikut sebelum tindakan:

```
awsdeadlinecloud
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [  
    "awsdeadlinecloud:action1",  
    "awsdeadlinecloud:action2"  
]
```

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)

Sumber daya kebijakan untuk Deadline Cloud

Mendukung sumber daya 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 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 Cloud Deadline dan jenisnya ARNs, lihat Sumber Daya yang [ditentukan oleh AWS Deadline Cloud](#) di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat [Tindakan yang ditentukan oleh AWS Deadline Cloud](#).

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)

Kunci kondisi kebijakan untuk Deadline Cloud

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.

Untuk melihat daftar kunci kondisi Deadline Cloud, lihat Kunci kondisi [untuk AWS Deadline Cloud](#) di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat [Tindakan yang ditentukan oleh AWS Deadline Cloud](#).

Untuk melihat contoh kebijakan berbasis identitas Deadline Cloud, lihat. [Contoh kebijakan berbasis identitas untuk Deadline Cloud](#)

ACLs di Deadline Cloud

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.

ABAC dengan Deadline Cloud

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.

Menggunakan kredensial sementara dengan Deadline Cloud

Mendukung kredensial sementara: Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensial 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 kredensi 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 kredensial 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 kredensial sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensial sementara tersebut untuk mengakses AWS. AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat [Kredensial keamanan sementara di IAM](#).

Teruskan sesi akses untuk Deadline Cloud

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 Deadline Cloud

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 Deadline Cloud. Edit peran layanan hanya jika Deadline Cloud memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Deadline Cloud

Mendukung peran terkait layanan: Tidak

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 pembuatan atau manajemen peran terkait layanan, lihat [Layanan AWS yang berfungsi dengan IAM](#). Cari layanan dalam tabel yang memiliki Yes di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk Deadline Cloud

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Deadline Cloud. 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 Deadline Cloud, termasuk format ARNs untuk setiap jenis sumber daya, lihat [Kunci tindakan, sumber daya, dan kondisi untuk AWS Deadline Cloud](#) di Referensi Otorisasi Layanan.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol Cloud Deadline](#)

- [Kebijakan untuk mengirimkan pekerjaan ke antrian](#)
- [Kebijakan untuk mengizinkan pembuatan titik akhir lisensi](#)
- [Kebijakan untuk memungkinkan pemantauan antrian pertanian tertentu](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Deadline Cloud 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 Akun AWS Anda. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan terkelola 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.

Menggunakan konsol Cloud Deadline

Untuk mengakses konsol AWS Deadline Cloud, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Cloud Deadline di Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Deadline Cloud, lampirkan juga Deadline Cloud *ConsoleAccess* atau kebijakan *ReadOnly* AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat [Menambah izin untuk pengguna](#) dalam Panduan Pengguna IAM.

Kebijakan untuk mengirimkan pekerjaan ke antrian

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin untuk mengirimkan pekerjaan ke antrian tertentu di peternakan tertentu.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SubmitJobsFarmAndQueue",
```

```

        "Effect": "Allow",
        "Action": ":createJob",
        "Resource": "arn:aws::REGION:ACCOUNT_ID:farm/FARM_A/queue/QUEUE_B/
job/*"
    }
  ]
}

```

Kebijakan untuk mengizinkan pembuatan titik akhir lisensi

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin yang diperlukan untuk membuat dan mengelola titik akhir lisensi. Gunakan kebijakan ini untuk membuat titik akhir lisensi untuk VPC yang terkait dengan farm Anda.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "CreateLicenseEndpoint",
    "Effect": "Allow",
    "Action": [
      ":CreateLicenseEndpoint",
      ":DeleteLicenseEndpoint",
      ":GetLicenseEndpoint",
      ":ListLicenseEndpoints",
      ":PutMeteredProduct",
      ":DeleteMeteredProduct",
      ":ListMeteredProducts",
      ":ListAvailableMeteredProducts",
      "ec2:CreateVpcEndpoint",
      "ec2:DescribeVpcEndpoints",
      "ec2>DeleteVpcEndpoints"
    ],
    "Resource": [
      "arn:aws::*:ACCOUNT_ID:*",
      "arn:aws:ec2::*:ACCOUNT_ID:vpc-endpoint/*"
    ]
  }]
}

```

Kebijakan untuk memungkinkan pemantauan antrian pertanian tertentu

Dalam contoh ini, Anda membuat kebijakan cakupan bawah yang memberikan izin untuk memantau pekerjaan dalam antrian tertentu untuk peternakan tertentu.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "MonitorJobsFarmAndQueue",
    "Effect": "Allow",
    "Action": [
      ":SearchJobs",
      ":ListJobs",
      ":GetJob",
      ":SearchSteps",
      ":ListSteps",
      ":ListStepConsumers",
      ":ListStepDependencies",
      ":GetStep",
      ":SearchTasks",
      ":ListTasks",
      ":GetTask",
      ":ListSessions",
      ":GetSession",
      ":ListSessionActions",
      ":GetSessionAction"
    ],
    "Resource": [
      "arn:aws::REGION:123456789012:farm/FARM_A/queue/QUEUE_B",
      "arn:aws::REGION:123456789012:farm/FARM_A/queue/QUEUE_B/*"
    ]
  }]
}
```

AWS kebijakan terkelola untuk Deadline Cloud

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan [kebijakan yang dikelola pelanggan](#) yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat [Kebijakan terkelola AWS](#) dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AWSDeadlineCloud-FleetWorker

Anda dapat melampirkan AWSDeadlineCloud-FleetWorker kebijakan ke identitas AWS Identity and Access Management (IAM) Anda.

Kebijakan ini memberi pekerja di armada ini izin yang diperlukan untuk terhubung dan menerima tugas dari layanan.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan kepala sekolah untuk mengelola pekerja dalam armada.

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-FleetWorker](#) di panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-WorkerHost

Anda dapat melampirkan kebijakan AWSDeadlineCloud-WorkerHost ke identitas IAM Anda.

Kebijakan ini memberikan izin yang diperlukan untuk awalnya terhubung ke layanan. Ini dapat digunakan sebagai profil instans Amazon Elastic Compute Cloud (Amazon EC2).

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk membuat pekerja, mengambil peran armada untuk pekerja, dan menerapkan tag untuk pekerja

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-WorkerHost](#) di panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessFarms

Anda dapat melampirkan kebijakan AWSDeadlineCloud-UserAccessFarms ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data pertanian berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis EC2 instans Amazon.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-UserAccessFarms](#) di panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessFleets

Anda dapat melampirkan kebijakan AWSDeadlineCloud-UserAccessFleets ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data armada berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis EC2 instans Amazon.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-UserAccessFleets](#) di panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessJobs

Anda dapat melampirkan kebijakan AWSDeadlineCloud-UserAccessJobs ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data pekerjaan berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis EC2 instans Amazon.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-UserAccessJobs](#) di panduan referensi Kebijakan Terkelola AWS.

AWS kebijakan terkelola: AWSDeadlineCloud-UserAccessQueues

Anda dapat melampirkan kebijakan AWSDeadlineCloud-UserAccessQueues ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna untuk mengakses data antrian berdasarkan peternakan tempat mereka menjadi anggota dan tingkat keanggotaan mereka.

Detail izin

Kebijakan ini mencakup izin berikut:

- `deadline`— Memungkinkan pengguna untuk mengakses data pertanian.
- `ec2`— Memungkinkan pengguna untuk melihat detail tentang jenis EC2 instans Amazon.
- `identitystore`— Memungkinkan pengguna untuk melihat nama pengguna dan grup.

Untuk daftar JSON tentang detail kebijakan, lihat [AWSDeadlineCloud-UserAccessQueues](#) di panduan referensi Kebijakan Terkelola AWS.

Pembaruan Cloud batas waktu ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Deadline Cloud sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen Cloud Tenggat Waktu.

Perubahan	Deskripsi	Tanggal
AWSDeadlineCloud-WorkerHost — Ubah	Deadline Cloud menambahkan tindakan baru <code>deadline:TagResource</code> dan <code>deadline:ListTagsForResource</code> memungkinkan Anda menambahkan dan melihat tag yang terkait dengan pekerja di armada Anda.	30 Mei 2025
AWSDeadlineCloud-UserAccessFarms — Ubah AWSDeadlineCloud-UserAccessJobs — Ubah AWSDeadlineCloud-UserAccessQueues — Ubah	Deadline Cloud menambahkan tindakan baru <code>deadline:GetJobTemplate</code> dan <code>deadline:ListJobParameterDefinitions</code> memungkinkan Anda	Oktober 7, 2024

Perubahan	Deskripsi	Tanggal
	mengirimkan kembali pekerjaan.	
Deadline Cloud mulai melacak perubahan	Deadline Cloud mulai melacak perubahan pada kebijakan AWS terkelolanya.	April 2, 2024

Pemecahan Masalah AWS Batas Waktu Identitas dan akses Cloud

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Deadline Cloud dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di Deadline Cloud](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Deadline Cloud saya](#)

Saya tidak berwenang untuk melakukan tindakan di Deadline Cloud

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 `awsdeadlinecloud:GetWidget` rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awsdeadlinecloud: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 `awsdeadlinecloud: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 yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Deadline Cloud.

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 Deadline Cloud. 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 Deadline Cloud 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 Deadline Cloud mendukung fitur-fitur ini, lihat [Bagaimana Deadline Cloud 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](#).

Validasi kepatuhan untuk Deadline Cloud

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 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 Manager](#)Ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Ketahanan di Deadline Cloud

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat [Infrastruktur AWS Global](#).

AWS Deadline Cloud tidak mencadangkan data yang disimpan di bucket S3 lampiran pekerjaan Anda. [Anda dapat mengaktifkan pencadangan data lampiran pekerjaan Anda menggunakan mekanisme pencadangan Amazon S3 standar apa pun, seperti Pembuatan Versi S3 atau AWS Backup](#)

Keamanan infrastruktur di Deadline Cloud

Sebagai layanan terkelola, AWS Deadline Cloud 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 Deadline Cloud 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.

Deadline Cloud tidak mendukung penggunaan kebijakan titik akhir AWS PrivateLink virtual private cloud (VPC). Ini menggunakan kebijakan AWS PrivateLink default, yang memberikan akses penuh ke titik akhir. Untuk informasi selengkapnya, lihat [Kebijakan titik akhir default](#) di panduan AWS PrivateLink pengguna.

Analisis konfigurasi dan kerentanan di Deadline Cloud

AWS menangani tugas-tugas keamanan dasar seperti sistem operasi tamu (OS) dan patching database, konfigurasi firewall, dan pemulihan bencana. Prosedur ini telah ditinjau dan disertifikasi oleh pihak ketiga yang sesuai. Untuk detail selengkapnya, lihat sumber daya berikut:

- [Model Tanggung Jawab Bersama](#)
- [Amazon Web Services: Gambaran Umum Proses Keamanan](#) (whitepaper)

AWS Deadline Cloud mengelola tugas pada armada yang dikelola layanan atau yang dikelola pelanggan:

- Untuk armada yang dikelola layanan, Deadline Cloud mengelola sistem operasi tamu.
- Untuk armada yang dikelola pelanggan, Anda bertanggung jawab untuk mengelola sistem operasi.

Untuk informasi tambahan tentang konfigurasi dan analisis kerentanan untuk AWS Deadline Cloud, lihat

- [Praktik terbaik keamanan untuk Deadline Cloud](#)

Pencegahan "confused deputy" lintas layanan

Masalah "confused deputy" adalah masalah keamanan saat entitas yang tidak memiliki izin untuk melakukan suatu tindakan dapat memaksa entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan tersebut. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data untuk semua layanan dengan principal layanan yang telah diberi akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi [aws:SourceAccount](#) global [aws:SourceArn](#) dan dalam kebijakan sumber daya untuk membatasi izin yang AWS Deadline Cloud memberikan layanan lain ke sumber daya. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan Nama Sumber Daya Amazon (ARN) lengkap dari sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci konteks kondisi global `aws:SourceArn` dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:awsdeadlinecloud:*:123456789012:*`.

Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan Deadline Cloud untuk mencegah masalah wakil yang membingungkan.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": ".amazonaws.com"
    },
    "Action": ":*ActionName",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws::*:123456789012:"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Akses AWS Deadline Cloud menggunakan endpoint antarmuka ()AWS PrivateLink

Anda dapat menggunakan AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan AWS Deadline Cloud. Anda dapat mengakses Deadline Cloud seolah-olah itu ada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses Deadline Cloud.

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka, yang didukung oleh AWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditakdirkan. Deadline Cloud

Deadline Cloud juga memiliki titik akhir dual-stack yang tersedia. Titik akhir dual-stack mendukung permintaan over dan. IPv6 IPv4

Untuk informasi selengkapnya, lihat [Mengakses Layanan AWS melalui AWS PrivateLink](#) di Panduan AWS PrivateLink .

Pertimbangan untuk Deadline Cloud

Sebelum Anda menyiapkan titik akhir antarmuka Deadline Cloud, lihat [Mengakses layanan AWS menggunakan titik akhir VPC antarmuka](#) dalam Panduan.AWS PrivateLink

Deadline Cloud mendukung panggilan ke semua tindakan API-nya melalui titik akhir antarmuka.

Secara default, akses penuh ke Deadline Cloud diizinkan melalui titik akhir antarmuka. Atau, Anda dapat mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengontrol lalu lintas Deadline Cloud melalui titik akhir antarmuka.

Deadline Cloud juga mendukung kebijakan titik akhir VPC. Untuk informasi selengkapnya, lihat [Mengontrol akses ke titik akhir VPC menggunakan kebijakan titik akhir](#) di Panduan.AWS PrivateLink

Deadline Cloud titik akhir

Deadline Cloud menggunakan empat titik akhir untuk akses ke layanan menggunakan AWS PrivateLink - dua untuk IPv4 dan dua untuk IPv6.

Pekerja menggunakan `scheduling.deadline.region.amazonaws.com` endpoint untuk mendapatkan tugas dari antrian, melaporkan kemajuan ke Deadline Cloud, dan mengirim output tugas kembali. Jika Anda menggunakan armada yang dikelola pelanggan, titik akhir penjadwalan adalah satu-satunya titik akhir yang perlu Anda buat kecuali Anda menggunakan operasi manajemen. Misalnya, jika pekerjaan menciptakan lebih banyak pekerjaan, Anda perlu mengaktifkan titik akhir manajemen untuk memanggil `CreateJob` operasi.

Deadline Cloud Monitor menggunakan `management.deadline.region.amazonaws.com` untuk mengelola sumber daya di peternakan Anda, seperti membuat dan memodifikasi antrian dan armada atau mendapatkan daftar pekerjaan, langkah, dan tugas.

Deadline Cloud juga membutuhkan titik akhir untuk titik akhir AWS layanan berikut:

- Deadline Cloud digunakan AWS STS untuk mengautentikasi pekerja sehingga mereka dapat mengakses aset pekerjaan. Untuk informasi selengkapnya AWS STS, lihat [Kredensial keamanan sementara di IAM di Panduan](#) Pengguna.AWS Identity and Access Management

- Jika Anda menyiapkan armada yang dikelola pelanggan di subnet tanpa koneksi internet, Anda harus membuat titik akhir VPC untuk CloudWatch Amazon Logs agar pekerja dapat menulis log. Untuk informasi selengkapnya, lihat [Memantau dengan CloudWatch](#).
- Jika Anda menggunakan lampiran pekerjaan, Anda harus membuat titik akhir VPC untuk Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) sehingga pekerja dapat mengakses lampiran. Untuk informasi selengkapnya, lihat [Lampiran Job di Deadline Cloud](#).

Buat titik akhir untuk Deadline Cloud

Anda dapat membuat titik akhir antarmuka untuk Deadline Cloud menggunakan konsol VPC Amazon atau () AWS Command Line Interface .AWS CLI Untuk informasi selengkapnya, lihat [Membuat titik akhir antarmuka](#) di AWS PrivateLink Panduan.

Buat endpoint manajemen dan penjadwalan untuk Deadline Cloud menggunakan nama layanan berikut. Ganti *region* dengan Wilayah AWS tempat yang Anda gunakan. Deadline Cloud

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud mendukung titik akhir dual-stack.

Jika Anda mengaktifkan DNS pribadi untuk titik akhir antarmuka, Anda dapat membuat permintaan API untuk Deadline Cloud menggunakan nama DNS Regional default. Misalnya, `scheduling.deadline.us-east-1.amazonaws.com` untuk operasi pekerja, atau `management.deadline.us-east-1.amazonaws.com` untuk semua operasi lainnya.

Anda juga harus membuat endpoint untuk AWS STS menggunakan nama layanan berikut:

```
com.amazonaws.region.sts
```

Jika armada yang dikelola pelanggan Anda berada di subnet tanpa koneksi internet, Anda harus membuat titik akhir CloudWatch Log menggunakan nama layanan berikut:

```
com.amazonaws.region.logs
```

Jika Anda menggunakan lampiran pekerjaan untuk mentransfer file, Anda harus membuat titik akhir Amazon S3 menggunakan nama layanan berikut:

```
com.amazonaws.region.s3
```

Praktik terbaik keamanan untuk Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Note

Untuk informasi selengkapnya tentang pentingnya banyak topik keamanan, lihat [Model Tanggung Jawab Bersama](#).

Perlindungan data

Untuk tujuan perlindungan data, kami menyarankan Anda untuk melindungi Akun AWS kredensial dan menyiapkan akun individual dengan AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon Simple Storage Service (Amazon S3).
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi sensitif apapun, seperti nomor rekening pelanggan Anda, ke dalam kolom isian teks bebas seperti kolom Nama. Ini termasuk saat Anda bekerja dengan AWS Deadline Cloud atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke Deadline Cloud atau layanan lain mungkin diambil untuk dimasukkan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

AWS Identity and Access Management izin

Kelola akses ke AWS sumber daya menggunakan pengguna, peran AWS Identity and Access Management (IAM), dan dengan memberikan hak istimewa paling sedikit kepada pengguna. Menetapkan kebijakan dan prosedur manajemen kredensial untuk membuat, mendistribusikan, memutar, dan mencabut AWS kredensial akses. Untuk informasi selengkapnya, lihat [Praktik Terbaik IAM](#) dalam Panduan Pengguna IAM.

Jalankan pekerjaan sebagai pengguna dan grup

Saat menggunakan fungsionalitas antrian di Deadline Cloud, ini adalah praktik terbaik untuk menentukan pengguna sistem operasi (OS) dan grup utamanya sehingga pengguna OS memiliki izin hak istimewa paling sedikit untuk pekerjaan antrian.

Saat Anda menentukan “Jalankan sebagai pengguna” (dan grup), proses apa pun untuk pekerjaan yang dikirimkan ke antrian akan dijalankan menggunakan pengguna OS tersebut dan akan mewarisi izin OS terkait pengguna tersebut.

Konfigurasi armada dan antrian bergabung untuk membangun postur keamanan. Di sisi antrian, peran “Job run as user” dan IAM dapat ditentukan untuk menggunakan OS dan AWS izin untuk pekerjaan antrian. Armada mendefinisikan infrastruktur (host pekerja, jaringan, penyimpanan bersama yang dipasang) yang, ketika dikaitkan dengan antrian tertentu, menjalankan pekerjaan dalam antrian. Data yang tersedia pada host pekerja perlu diakses oleh pekerjaan dari satu atau lebih antrian terkait. Menentukan pengguna atau grup membantu melindungi data dalam pekerjaan dari antrian lain, perangkat lunak lain yang diinstal, atau pengguna lain dengan akses ke host pekerja. Ketika antrian tanpa pengguna, itu berjalan sebagai pengguna agen yang dapat meniru (sudo) setiap pengguna antrian. Dengan cara ini, antrian tanpa pengguna dapat meningkatkan hak istimewa ke antrian lain.

Jaringan

Untuk mencegah lalu lintas dicegat atau dialihkan, penting untuk mengamankan bagaimana dan di mana lalu lintas jaringan Anda diarahkan.

Kami menyarankan Anda mengamankan lingkungan jaringan Anda dengan cara berikut:

- Amankan tabel rute subnet Amazon Virtual Private Cloud (Amazon VPC) untuk mengontrol bagaimana lalu lintas lapisan IP dirutekan.
- Jika Anda menggunakan Amazon Route 53 (Route 53) sebagai penyedia DNS di penyiapan farm atau workstation Anda, amankan akses ke API Route 53.
- Jika Anda tersambung ke Deadline Cloud di luar AWS seperti menggunakan workstation lokal atau pusat data lainnya, amankan infrastruktur jaringan lokal apa pun. Ini termasuk server DNS dan tabel rute pada router, switch, dan perangkat jaringan lainnya.

Pekerjaan dan data pekerjaan

Tenggat waktu pekerjaan Cloud berjalan dalam sesi di host pekerja. Setiap sesi menjalankan satu atau lebih proses pada host pekerja, yang umumnya mengharuskan Anda memasukkan data untuk menghasilkan output.

Untuk mengamankan data ini, Anda dapat mengonfigurasi pengguna sistem operasi dengan antrian. Agen pekerja menggunakan pengguna OS antrian untuk menjalankan sub-proses sesi. Sub-proses ini mewarisi izin pengguna OS antrian.

Kami menyarankan Anda mengikuti praktik terbaik untuk mengamankan akses ke data akses sub-proses ini. Untuk informasi selengkapnya, lihat [Model tanggung jawab bersama](#).

Struktur pertanian

Anda dapat mengatur armada Deadline Cloud dan antrian banyak cara. Namun, ada implikasi keamanan dengan pengaturan tertentu.

Sebuah peternakan memiliki salah satu batas paling aman karena tidak dapat berbagi sumber daya Deadline Cloud dengan peternakan lain, termasuk armada, antrian, dan profil penyimpanan. Namun, Anda dapat berbagi AWS sumber daya eksternal di dalam peternakan, yang membahayakan batas keamanan.

Anda juga dapat menetapkan batas keamanan antara antrian dalam peternakan yang sama menggunakan konfigurasi yang sesuai.

Ikuti praktik terbaik ini untuk membuat antrian aman di peternakan yang sama:

- Kaitkan armada hanya dengan antrian dalam batas keamanan yang sama. Perhatikan hal berikut:
 - Setelah pekerjaan berjalan di host pekerja, data mungkin tetap tertinggal, seperti di direktori sementara atau direktori home pengguna antrian.
 - Pengguna OS yang sama menjalankan semua pekerjaan pada host pekerja armada milik layanan, terlepas dari antrian mana Anda mengirimkan pekerjaan.
 - Pekerjaan mungkin membiarkan proses berjalan pada host pekerja, sehingga memungkinkan pekerjaan dari antrian lain untuk mengamati proses berjalan lainnya.
- Pastikan hanya antrian dalam batas keamanan yang sama yang berbagi bucket Amazon S3 untuk lampiran pekerjaan.
- Pastikan bahwa hanya antrian dalam batas keamanan yang sama berbagi pengguna OS.
- Amankan AWS sumber daya lain yang terintegrasi ke dalam pertanian hingga batas.

Antrian lampiran pekerjaan

Lampiran Job dikaitkan dengan antrian, yang menggunakan bucket Amazon S3 Anda.

- Lampiran Job menulis dan membaca dari awalan root di bucket Amazon S3. Anda menentukan awalan root ini dalam panggilan `CreateQueue` API.
- Bucket memiliki kode yang sesuai `QueueRole`, yang menentukan peran yang memberi pengguna antrian akses ke awalan bucket dan root. Saat membuat antrian, Anda menentukan Nama Sumber Daya `QueueRole` Amazon (ARN) di samping bucket lampiran pekerjaan dan awalan root.
- Panggilan resmi ke `AssumeQueueRoleForRead`, `AssumeQueueRoleForUser`, dan operasi `AssumeQueueRoleForWorker` API mengembalikan satu set kredensial keamanan sementara untuk `QueueRole`.

Jika Anda membuat antrian dan menggunakan kembali bucket Amazon S3 dan awalan root, ada risiko informasi diungkapkan kepada pihak yang tidak berwenang. Misalnya, `QueueA` dan `QueueB` berbagi bucket dan awalan root yang sama. Dalam alur kerja yang aman, `ArtisTa` memiliki akses ke `QueueA` tetapi tidak `QueueB`. Namun, ketika beberapa antrian berbagi bucket, `ArtisTA` dapat

mengakses data dalam data QueueB karena menggunakan bucket dan awalan root yang sama dengan QueueA.

Konsol mengatur antrian yang aman secara default. Pastikan antrian memiliki kombinasi yang berbeda antara bucket Amazon S3 dan awalan root kecuali mereka merupakan bagian dari batas keamanan umum.

Untuk mengisolasi antrian Anda, Anda harus mengonfigurasi Queue Role untuk hanya mengizinkan akses antrian ke bucket dan awalan root. Dalam contoh berikut, ganti masing-masing *placeholder* dengan informasi spesifik sumber daya Anda.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",
        "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "ACCOUNT_ID"
        }
      }
    },
    {
      "Action": [
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-east-1:ACCOUNT_ID:log-group:/aws/
deadline/FARM_ID/*"
    }
  ]
}
```

```

    }
  ]
}

```

Anda juga harus menetapkan kebijakan kepercayaan tentang peran tersebut. Dalam contoh berikut, ganti *placeholder* teks dengan informasi spesifik sumber daya Anda.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sts:AssumeRole"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": "deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-
east-1:ACCOUNT_ID:farm/FARM_ID"
        }
      }
    },
    {
      "Action": [
        "sts:AssumeRole"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "ACCOUNT_ID"
        }
      }
    }
  ]
}

```

```

    },
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:deadline:us-east-1:ACCOUNT_ID:farm/FARM_ID"
    }
  }
]
}

```

Bucket Amazon S3 perangkat lunak khusus

Anda dapat menambahkan pernyataan berikut ke perangkat lunak khusus Queue Role untuk mengakses perangkat lunak khusus di bucket Amazon S3 Anda. Dalam contoh berikut, ganti *SOFTWARE_BUCKET_NAME* dengan nama bucket S3 Anda.

```

"Statement": [
  {
    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
      "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
    ]
  }
]

```

Untuk informasi selengkapnya tentang praktik terbaik keamanan Amazon S3, lihat Praktik [terbaik keamanan untuk Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Tuan rumah pekerja

Host pekerja aman untuk membantu memastikan bahwa setiap pengguna hanya dapat melakukan operasi untuk peran yang ditetapkan.

Kami merekomendasikan praktik terbaik berikut untuk mengamankan host pekerja:

- Menggunakan skrip konfigurasi host dapat mengubah keamanan dan operasi pekerja. Konfigurasi yang salah dapat menyebabkan pekerja menjadi tidak stabil atau berhenti bekerja. Adalah tanggung jawab Anda untuk men-debug kegagalan tersebut.
- Jangan gunakan `jobRunAsUser` nilai yang sama dengan beberapa antrian kecuali pekerjaan yang dikirimkan ke antrian tersebut berada dalam batas keamanan yang sama.
- Jangan atur antrian `jobRunAsUser` ke nama pengguna OS yang dijalankan oleh agen pekerja.
- Berikan izin OS dengan hak istimewa paling sedikit kepada pengguna antrian yang diperlukan untuk beban kerja antrian yang dimaksud. Pastikan bahwa mereka tidak memiliki izin menulis sistem file untuk bekerja file program agen atau perangkat lunak bersama lainnya.
- Pastikan hanya pengguna root yang aktif Linux dan Administrator memiliki akun sendiri Windows dan dapat memodifikasi file program agen pekerja.
- Pada host Linux pekerja, pertimbangkan untuk mengonfigurasi umask penggantian `/etc/sudoers` yang memungkinkan pengguna agen pekerja meluncurkan proses sebagai pengguna antrian. Konfigurasi ini membantu memastikan pengguna lain tidak dapat mengakses file yang ditulis ke antrian.
- Berikan individu tepercaya akses paling tidak memiliki hak istimewa ke host pekerja.
- Batasi izin untuk mengganti file konfigurasi DNS lokal (`/etc/hosts` aktif dan `aktifWindows`), Linux dan untuk merutekan tabel `C:\Windows\system32\etc\hosts` di workstation dan sistem operasi host pekerja.
- Batasi izin untuk konfigurasi DNS pada workstation dan sistem operasi host pekerja.
- Secara teratur menambal sistem operasi dan semua perangkat lunak yang diinstal. Pendekatan ini mencakup perangkat lunak yang khusus digunakan dengan Deadline Cloud seperti submitter, adaptor, agen pekerja, OpenJD paket, dan lain-lain.
- Gunakan kata sandi yang kuat untuk Windows antrian `jobRunAsUser`.
- Putar kata sandi untuk antrian `jobRunAsUser` Anda secara teratur.
- Pastikan akses hak istimewa paling sedikit ke rahasia Windows kata sandi dan hapus rahasia yang tidak digunakan.
- Jangan berikan `jobRunAsUser` izin antrian perintah jadwal untuk dijalankan di masa mendatang:
 - Pada Linux, tolak akses akun ini ke `cron` dan `at`.
 - On Windows, tolak akses akun ini ke penjadwal Windows tugas.

 Note

Untuk informasi selengkapnya tentang pentingnya menambal sistem operasi dan perangkat lunak yang diinstal secara teratur, lihat [Model Tanggung Jawab Bersama](#).

Skrip konfigurasi host

- Menggunakan skrip konfigurasi host dapat mengubah keamanan dan operasi pekerja. Konfigurasi yang salah dapat menyebabkan pekerja menjadi tidak stabil atau berhenti bekerja. Adalah tanggung jawab Anda untuk men-debug kegagalan tersebut.

Workstation

Sangat penting untuk mengamankan workstation dengan akses ke Deadline Cloud. Pendekatan ini membantu memastikan bahwa pekerjaan apa pun yang Anda kirimkan ke Deadline Cloud tidak dapat menjalankan beban kerja sewenang-wenang yang ditagih ke Anda. Akun AWS

Kami merekomendasikan praktik terbaik berikut untuk mengamankan workstation artis. Untuk informasi selengkapnya, lihat [Model Tanggung Jawab Bersama](#).

- Amankan semua kredensial tetap yang menyediakan akses ke AWS, termasuk Deadline Cloud. Untuk informasi lebih lanjut, lihat [Mengelola access key untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
- Hanya instal perangkat lunak tepercaya dan aman.
- Mengharuskan pengguna berfederasi dengan penyedia identitas untuk mengakses AWS dengan kredensi sementara.
- Gunakan izin aman pada file program submitter Deadline Cloud untuk mencegah gangguan.
- Berikan individu tepercaya akses paling tidak istimewa ke workstation artis.
- Hanya gunakan pengirim dan adaptor yang Anda dapatkan melalui Deadline Cloud Monitor.
- Batasi izin ke DNS lokal mengganti file konfigurasi (/etc/hosts on Linux dan macOS, dan C : \Windows\system32\etc\hosts on Windows), dan untuk merutekan tabel pada workstation dan sistem operasi host pekerja.
- Batasi izin /etc/resolve.conf pada workstation dan sistem operasi host pekerja.

- Secara teratur menambal sistem operasi dan semua perangkat lunak yang diinstal. Pendekatan ini mencakup perangkat lunak yang khusus digunakan dengan Deadline Cloud seperti submitter, adaptor, agen pekerja, OpenJD paket, dan lain-lain.

Verifikasi keaslian perangkat lunak yang diunduh

Verifikasi keaslian perangkat lunak Anda setelah mengunduh penginstal untuk melindungi dari gangguan file. Prosedur ini berfungsi untuk keduanya Windows dan Linux sistem.

Windows

Untuk memverifikasi keaslian file yang Anda unduh, selesaikan langkah-langkah berikut.

1. Dalam perintah berikut, ganti *file* dengan file yang ingin Anda verifikasi. Misalnya, **C:\PATH\TO\MY\DeadlineCloudSubmitter-windows-x64-installer.exe** . Juga, ganti *signtool-sdk-version* dengan versi SignTool SDK yang diinstal. Misalnya, **10.0.22000.0**.

```
"C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
```

2. Misalnya, Anda dapat memverifikasi file installer submitter Deadline Cloud dengan menjalankan perintah berikut:

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-windows-x64-installer.exe
```

Linux

Untuk memverifikasi keaslian file yang Anda unduh, gunakan alat baris gpg perintah.

1. Impor OpenPGP kunci dengan menjalankan perintah berikut:

```
gpg --import --armor <<EOF
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q7606fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
1e4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFekjopA3pjsTBP6lW/mb1bDBDEwwwtH0x91V7A03FJ9T7Uzu/qSh
```

```

q0/Uydkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZx0LZk/fvpYPMYEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2015DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3Ncz0ozPoVJt+fw8CB1VIX00J715
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3i0J5gBYUNXqHSxUdv8kt76/KVmQa1B
Ak1+MPKpMq+1hw++S3G/1XqwWadNQbRRw7dSZHymQVXvPp1nsqc3hV7K10M+6s6g
1g4mvFY41f6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbnUgQ2xvdWQgPGF3cy1kZWFKbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNP0a3bzzvBQJl+hkLAXsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNP0a3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUE06MTt9AykF/jw+CQg2UzFtEy0bHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
10uxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIR1Qyctq8gnR9JvYXX
42ASqLq5+0XKo4qh81b1XKYqtc176BbbSNFjWnzIQgKDgNiHFZCdc0VgqDhw015r
NICbqqwwNLj/Fr2kecYx180Ktp10j00w5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSR15DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageY158vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQ02Fx7fd2QYJheIPPASHHcfJ0+xgWCoF45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ410u/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF

```

2. Tentukan apakah akan mempercayai OpenPGP kuncinya. Beberapa faktor yang perlu dipertimbangkan ketika memutuskan apakah akan mempercayai kunci di atas termasuk yang berikut:
 - Koneksi internet yang Anda gunakan untuk mendapatkan kunci GPG dari situs web ini aman.
 - Perangkat tempat Anda mengakses situs web ini aman.
 - AWS telah mengambil langkah-langkah untuk mengamankan hosting kunci OpenPGP publik di situs web ini.
3. Jika Anda memutuskan untuk mempercayai OpenPGP kunci, edit kunci untuk dipercaya dengan gpg mirip dengan contoh berikut:

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF
```

```
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
```

```
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.
```

```
pub 4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA  
                        trust: unknown      validity: unknown  
[ unknown] (1). AWS Deadline Cloud example@example.com
```

```
gpg> trust  
pub 4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA  
                        trust: unknown      validity: unknown  
[ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
```

Please decide how far you trust this user to correctly verify other users' keys

(by looking at passports, checking fingerprints from different sources, etc.)

```
1 = I don't know or won't say  
2 = I do NOT trust  
3 = I trust marginally  
4 = I trust fully  
5 = I trust ultimately  
m = back to the main menu
```

Your decision? 5

Do you really want to set this key to ultimate trust? (y/N) y

```
pub 4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA  
                        trust: ultimate      validity: unknown  
[ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com  
Please note that the shown key validity is not necessarily correct  
unless you restart the program.
```

```
gpg> quit
```

4. Verifikasi penginstal pengirim Cloud Deadline

Untuk memverifikasi installer submitter Deadline Cloud, selesaikan langkah-langkah berikut:

- a. Kembali ke halaman Unduhan [konsol](#) Cloud Deadline dan unduh file tanda tangan untuk penginstal pengirim Deadline Cloud.
- b. Verifikasi tanda tangan penginstal submitter Deadline Cloud dengan menjalankan:

```
gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
DeadlineCloudSubmitter-linux-x64-installer.run
```

5. Verifikasi monitor Cloud Deadline

Note

Anda dapat memverifikasi unduhan monitor Deadline Cloud menggunakan file tanda tangan atau metode khusus platform. Untuk metode khusus platform, lihat Linux (Debian) tab, tab Linux (RPM), atau Linux (AppImage) tab berdasarkan jenis file yang Anda unduh.

Untuk memverifikasi aplikasi desktop monitor Deadline Cloud dengan file tanda tangan, selesaikan langkah-langkah berikut:

- a. Kembali ke halaman Unduhan [konsol](#) Cloud Deadline dan unduh file.sig yang sesuai, lalu jalankan

Untuk.deb:

```
gpg --verify ./deadline-cloud-monitor_<APP_VERSION>_amd64.deb.sig ./
deadline-cloud-monitor_<APP_VERSION>_amd64.deb
```

Untuk.rpm:

```
gpg --verify ./deadline-cloud-monitor_<APP_VERSION>_x86_64.deb.sig ./
deadline-cloud-monitor_<APP_VERSION>_x86_64.rpm
```

Untuk. AppImage:

```
gpg --verify ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage.sig ./
deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

- b. Konfirmasikan bahwa output terlihat mirip dengan yang berikut:

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC
```

```
gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Jika output berisi frasa `Good signature from "AWS Deadline Cloud"`, itu berarti tanda tangan telah berhasil diverifikasi dan Anda dapat menjalankan skrip instalasi monitor Deadline Cloud.

Linux (AppImage)

Untuk memverifikasi paket yang menggunakan file Linux. AppImage biner, pertama selesaikan langkah 1-3 di Linux tab, lalu selesaikan langkah-langkah berikut.

1. Dari AppImageUpdate [halaman](#) di GitHub, unduh `validate-x86_64.AppImage`.
2. Setelah mengunduh file, untuk menambahkan izin eksekusi, jalankan perintah berikut.

```
chmod a+x ./validate-x86_64.AppImage
```

3. Untuk menambahkan izin eksekusi, jalankan perintah berikut.

```
chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

4. Untuk memverifikasi tanda tangan monitor Deadline Cloud, jalankan perintah berikut.

```
./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

Jika output berisi frasa `Validation successful`, itu berarti tanda tangan telah berhasil diverifikasi dan Anda dapat menjalankan skrip instalasi monitor Deadline Cloud dengan aman.

Linux (Debian)

Untuk memverifikasi paket yang menggunakan Linux biner .deb, pertama-tama selesaikan langkah 1-3 di tab. Linux

`dpkg` adalah alat manajemen paket inti di sebagian besar distribusi debian berbasis Linux. Anda dapat memverifikasi file .deb dengan alat ini.

1. Dari halaman Unduhan [Konsol](#) Cloud Deadline, unduh file Deadline Cloud monitor .deb.
2. Ganti `<APP_VERSION>` dengan versi file .deb yang ingin Anda verifikasi.

```
dpkg-sig --verify deadline-cloud-monitor_<APP_VERSION>_amd64.deb
```

3. Outputnya akan mirip dengan:

```
ProcessingLinux deadline-cloud-monitor_<APP_VERSION>_amd64.deb...  
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. Untuk memverifikasi file.deb, konfirmasikan bahwa GOODSIG ada dalam output.

Linux (RPM)

Untuk memverifikasi paket yang menggunakan Linux biner.rpm, pertama-tama selesaikan langkah 1-3 di Linux tab.

1. Dari halaman Unduhan [Konsol](#) Cloud Deadline, unduh file monitor Deadline Cloud .rpm.
2. Ganti <APP_VERSION> dengan versi file.rpm untuk memverifikasi.

```
gpg --export --armor "Deadline Cloud" > key.pub  
sudo rpm --import key.pub  
rpm -K deadline-cloud-monitor-<APP_VERSION>-1.x86_64.rpm
```

3. Outputnya akan mirip dengan:

```
deadline-cloud-monitor-deadline-cloud-  
monitor-<APP_VERSION>-1.x86_64.rpm-1.x86_64.rpm: digests signatures OK
```

4. Untuk memverifikasi file.rpm, konfirmasikan yang digests signatures OK ada di output.

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengembang Cloud AWS Deadline.

Perubahan	Deskripsi	Tanggal
Menambahkan skrip konfigurasi host pekerja	Menambahkan dokumentasi untuk menggunakan skrip konfigurasi host pekerja yang berjalan dengan hak istimewa yang ditinggikan. Untuk informasi selengkapnya, lihat Menjalankan skrip sebagai administrator untuk mengonfigurasi pekerja .	12 Mei 2025
Bagian keamanan yang diperbarui tentang penggunaan AWS PrivateLink	Memperbarui instruksi untuk mengakses Deadline Cloud menggunakan AWS PrivateLink dan titik akhir dual-stack baru. Untuk informasi selengkapnya, lihat Akses Batas Waktu Cloud menggunakan titik akhir antarmuka .	Maret 17, 2025
Informasi kredensi armada yang dikelola pelanggan yang diperbarui	Memperbarui instruksi untuk membuat kredensi untuk armada yang dikelola pelanggan untuk memberikan informasi lebih lanjut tentang mengamankan armada Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi kredensi AWS .	Februari 10, 2025

Konten yang direorganisasi dari panduan pengguna

Memindahkan konten yang berfokus pada pengembangan dari panduan pengguna ke panduan pengembang:

Januari 6, 2025

- Memindahkan instruksi untuk membuat armada yang dikelola pelanggan dari panduan pengguna ke chapter armada baru yang [dikelola Pelanggan](#).
- Membuat chapter baru [Menggunakan lisensi perangkat lunak](#) dengan informasi tentang lisensi berbasis penggunaan dan menggunakan lisensi Anda sendiri dengan armada yang dikelola layanan dan pelanggan.
- Memindahkan detail tentang pemantauan dengan CloudTrail CloudWatch, dan EventBridge dari panduan pengguna ke chapter [Monitoring](#).

Buat paket conda

Menambahkan informasi tentang cara membuat paket conda untuk aplikasi. Untuk informasi selengkapnya, lihat [Membuat paket conda](#).

Agustus 29, 2024

Panduan baru

Ini adalah rilis awal dari Deadline Cloud Developer Guide.

Juli 26, 2024

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.