aws

Panduan Pengguna

AWS AppConfig

AWS AppConfig: Panduan Pengguna

Table of Contents

Apakah AWS AppConfig itu?

AWS AppConfigflag fitur dan konfigurasi dinamis membantu pembangun perangkat lunak dengan cepat dan aman menyesuaikan perilaku aplikasi di lingkungan produksi tanpa penerapan kode penuh. AWS AppConfigmempercepat frekuensi rilis perangkat lunak, meningkatkan ketahanan aplikasi, dan membantu Anda mengatasi masalah yang muncul dengan lebih cepat. Dengan flag fitur, Anda dapat secara bertahap merilis kemampuan baru kepada pengguna dan mengukur dampak perubahan tersebut sebelum sepenuhnya menerapkan kemampuan baru ke semua pengguna. Dengan flag operasional dan konfigurasi dinamis, Anda dapat memperbarui daftar blokir, mengizinkan daftar, membatasi batas, mencatat verbositas, dan melakukan penyetelan operasional lainnya untuk merespons masalah di lingkungan produksi dengan cepat.

> **Note**
>
> AWS AppConfigadalah kemampuanAWS Systems Manager.

Tingkatkan efisiensi dan lepaskan perubahan lebih cepat

Menggunakan flag fitur dengan kemampuan baru mempercepat proses pelepasan perubahan pada lingkungan produksi. Alih-alih mengandalkan cabang pengembangan berumur panjang yang memerlukan penggabungan rumit sebelum rilis, flag fitur memungkinkan Anda menulis perangkat lunak menggunakan pengembangan berbasis batang. Bendera fitur memungkinkan Anda meluncurkan kode pra-rilis dengan aman dalam pipa CI/CD yang disembunyikan dari pengguna. Ketika Anda siap untuk merilis perubahan, Anda dapat memperbarui flag fitur tanpa menerapkan kode baru. Setelah peluncuran selesai, bendera masih dapat berfungsi sebagai sakelar blok untuk menonaktifkan fitur atau kemampuan baru tanpa perlu memutar kembali penerapan kode.

Hindari perubahan atau kegagalan yang tidak diinginkan dengan fitur keselamatan bawaan

AWS AppConfigmenawarkan fitur keamanan berikut untuk membantu Anda menghindari mengaktifkan flag fitur atau memperbarui data konfigurasi yang dapat menyebabkan kegagalan aplikasi.

- Validator: Validator memastikan bahwa data konfigurasi Anda benar secara sintaksis dan semantik sebelum menerapkan perubahan ke lingkungan produksi.
- Strategi penyebaran: Strategi penerapan memungkinkan Anda untuk secara perlahan melepaskan perubahan ke lingkungan produksi selama beberapa menit atau jam.

- Pemantauan dan rollback otomatis: AWS AppConfig terintegrasi dengan Amazon CloudWatch untuk memantau perubahan pada aplikasi Anda. Jika aplikasi Anda menjadi tidak sehat karena perubahan konfigurasi yang buruk dan perubahan itu memicu alarm masuk CloudWatch, secara AWS AppConfig otomatis memutar kembali perubahan untuk meminimalkan dampak pada pengguna aplikasi Anda.

Penerapan bendera fitur yang aman dan dapat diskalakan

AWS AppConfigterintegrasi dengan AWS Identity and Access Management (IAM) untuk menyediakan akses berbasis peran halus ke layanan. AWS AppConfigjuga terintegrasi dengan AWS Key Management Service (AWS KMS) untuk enkripsi dan AWS CloudTrail untuk audit. Sebelum dirilis ke pelanggan eksternal, semua kontrol AWS AppConfig keselamatan awalnya dikembangkan dengan dan divalidasi oleh pelanggan internal yang menggunakan layanan dalam skala besar.

Kasus penggunaan AWS AppConfig

Terlepas dari kenyataan bahwa konten konfigurasi aplikasi dapat sangat bervariasi dari aplikasi ke aplikasi, AWS AppConfig mendukung kasus penggunaan berikut, yang mencakup spektrum yang luas dari kebutuhan pelanggan:

- Bendera dan sakelar fitur — Lepaskan kemampuan baru dengan aman kepada pelanggan Anda di lingkungan yang terkendali. Putar kembali perubahan secara instan jika Anda mengalami masalah.
- Penyetelan aplikasi — Perkenalkan perubahan aplikasi dengan hati-hati saat menguji dampak perubahan tersebut dengan pengguna di lingkungan produksi.
- Izinkan daftar atau daftar blokir — Kontrol akses ke fitur premium atau langsung blokir pengguna tertentu tanpa menggunakan kode baru.
- Penyimpanan konfigurasi terpusat - Jaga agar data konfigurasi Anda tetap teratur dan konsisten di semua beban kerja Anda. Anda dapat menggunakan AWS AppConfig untuk menyebarkan data konfigurasi yang disimpan di penyimpanan konfigurasi yang AWS AppConfig dihosting,AWS Secrets Manager, Systems Manager Parameter Store, atau Amazon S3.

Manfaat menggunakan AWS AppConfig

AWS AppConfigmenawarkan manfaat berikut untuk organisasi Anda:

- Mengurangi down time yang tidak terduga untuk pelanggan Anda

AWS AppConfigmengurangi waktu henti aplikasi dengan memungkinkan Anda membuat aturan untuk memvalidasi konfigurasi Anda. Konfigurasi yang tidak valid tidak dapat diterapkan. AWS AppConfigmenyediakan dua opsi berikut untuk memvalidasi konfigurasi:

- Untuk validasi sintaksis, Anda dapat menggunakan skema JSON. AWS AppConfigmemvalidasi konfigurasi Anda dengan menggunakan skema JSON untuk memastikan bahwa perubahan konfigurasi mematuhi persyaratan aplikasi.
- Untuk validasi semantik, AWS AppConfig dapat memanggil AWS Lambda fungsi yang Anda miliki untuk memvalidasi data dalam konfigurasi Anda.

- Terapkan perubahan dengan cepat di seluruh set target

  AWS AppConfigmenyederhanakan administrasi aplikasi dalam skala besar dengan menerapkan perubahan konfigurasi dari lokasi pusat. AWS AppConfigmendukung konfigurasi yang disimpan di toko konfigurasi yang AWS AppConfig di-host, Systems Manager Parameter Store, dokumen Systems Manager (SSM), dan Amazon S3. Anda dapat menggunakan AWS AppConfig dengan aplikasi yang dihosting di instans EC2, wadahAWS Lambda, aplikasi seluler, atau perangkat IoT.

  Target tidak perlu dikonfigurasi dengan Agen SSM Systems Manager atau profil instans IAM yang diperlukan oleh kemampuan Systems Manager lainnya. Ini berarti bahwa AWS AppConfig bekerja dengan instance yang tidak dikelola.

- Perbarui aplikasi tanpa gangguan

  AWS AppConfigmenerapkan perubahan konfigurasi ke target Anda saat runtime tanpa proses build yang berat atau mengeluarkan target Anda dari layanan.

- Mengontrol penerapan perubahan di seluruh aplikasi Anda

  Saat menerapkan perubahan konfigurasi ke target Anda, AWS AppConfig memungkinkan Anda meminimalkan risiko dengan menggunakan strategi penerapan. Strategi penyebaran memungkinkan Anda untuk secara perlahan meluncurkan perubahan konfigurasi ke armada Anda. Jika Anda mengalami masalah selama penerapan, Anda dapat memutar kembali perubahan konfigurasi sebelum mencapai sebagian besar host Anda.

Cara kerja AWS AppConfig

Bagian ini memberikan deskripsi tingkat tinggi tentang cara AWS AppConfig kerja dan bagaimana Anda memulai.

1. Identifikasi nilai konfigurasi dalam kode yang ingin Anda kelola di cloud

   Sebelum Anda mulai membuat AWS AppConfig artefak, kami sarankan Anda mengidentifikasi data konfigurasi dalam kode Anda yang ingin Anda kelola secara dinamis menggunakan. AWS AppConfig Contoh yang baik termasuk flag fitur atau toggle, izinkan dan blokir daftar, logging verbositas, batas layanan, dan aturan pembatasan, untuk beberapa nama.

   Jika data konfigurasi Anda sudah ada di cloud, Anda dapat memanfaatkan fitur AWS AppConfig validasi, penerapan, dan ekstensi untuk lebih merampingkan manajemen data konfigurasi.

2. Buat namespace aplikasi

   Untuk membuat namespace, Anda membuat AWS AppConfig artefak yang disebut aplikasi. Aplikasi hanyalah konstruksi organisasi seperti folder.

3. Ciptakan lingkungan

   Untuk setiap AWS AppConfig aplikasi, Anda menentukan satu atau lebih lingkungan. Lingkungan adalah pengelompokan logis target, seperti aplikasi dalam `Beta` atau `Production` lingkungan, AWS Lambda fungsi, atau wadah. Anda juga dapat menentukan lingkungan untuk subkomponen aplikasi, seperti`Web`,`Mobile`, dan`Back-end`.

   Anda dapat mengonfigurasi CloudWatch alarm Amazon untuk setiap lingkungan. Sistem memantau alarm selama deployment konfigurasi. Jika alarm dipicu, sistem memutar kembali konfigurasi.

4. Buat profil konfigurasi

   Profil konfigurasi mencakup, antara lain, URI yang memungkinkan AWS AppConfig untuk menemukan data konfigurasi Anda di lokasi yang disimpan dan jenis profil. AWS AppConfigmendukung dua jenis profil konfigurasi: bendera fitur dan konfigurasi bentuk bebas. Profil konfigurasi bendera fitur menyimpan data mereka di toko konfigurasi yang AWS AppConfig dihosting dan URI sederhana`hosted`. Untuk profil konfigurasi bentuk bebas, Anda dapat menyimpan data Anda di penyimpanan konfigurasi yang AWS AppConfig dihosting atau AWS layanan apa pun yang terintegrasiAWS AppConfig, seperti yang dijelaskan dalam. Membuat profil konfigurasi formulir gratis di AWS AppConfig

   Profil konfigurasi juga dapat menyertakan validator opsional untuk memastikan data konfigurasi Anda benar secara sintaksis dan semantik. AWS AppConfigmelakukan pemeriksaan menggunakan validator saat Anda memulai penerapan. Jika ada kesalahan yang terdeteksi, penerapan akan kembali ke data konfigurasi sebelumnya.

5. Menyebarkan data konfigurasi

   Saat Anda membuat penerapan baru, Anda menentukan yang berikut ini:

   - ID aplikasi
   - ID profil konfigurasi
   - Versi konfigurasi
   - ID lingkungan tempat Anda ingin menyebarkan data konfigurasi
   - ID strategi penerapan yang menentukan seberapa cepat Anda ingin perubahan diterapkan

   Saat Anda memanggil tindakan StartDeploymentAPI, AWS AppConfig lakukan tugas-tugas berikut:

   1. Mengambil data konfigurasi dari penyimpanan data yang mendasarinya dengan menggunakan URI lokasi di profil konfigurasi.
   2. Memverifikasi data konfigurasi secara sintaksis dan semantik benar dengan menggunakan validator yang Anda tentukan saat Anda membuat profil konfigurasi.
   3. Cache salinan data sehingga siap untuk diambil oleh aplikasi Anda. Salinan cache ini disebut data yang digunakan.

6. Ambil konfigurasi

   Anda dapat mengonfigurasi AWS AppConfig Agen sebagai host lokal dan memiliki polling agen AWS AppConfig untuk pembaruan konfigurasi. Agen memanggil tindakan StartConfigurationSessiondan GetLatestConfigurationAPI dan menyimpan data konfigurasi Anda secara lokal. Untuk mengambil data, aplikasi Anda membuat panggilan HTTP ke server localhost. AWS AppConfig Agen mendukung beberapa kasus penggunaan, seperti yang dijelaskan dalamMetode pengambilan yang disederhanakan.

   Jika AWS AppConfig Agen tidak didukung untuk kasus penggunaan Anda, Anda dapat mengonfigurasi aplikasi Anda untuk melakukan polling AWS AppConfig untuk pembaruan konfigurasi dengan langsung memanggil tindakan StartConfigurationSessiondan GetLatestConfigurationAPI.

Memulai dengan AWS AppConfig

Sumber daya berikut dapat membantu Anda bekerja secara langsungAWS AppConfig.

Lihat lebih banyak AWS video di Amazon Web Services YouTube Channel.

Blog berikut dapat membantu Anda mempelajari lebih lanjut tentang AWS AppConfig dan kemampuannya:

- Menggunakan bendera AWS AppConfig fitur
- Praktik Terbaik untuk memvalidasi Bendera AWS AppConfig Fitur dan Data Konfigurasi

SDK

Untuk informasi tentang SDK AWS AppConfig khusus bahasa, lihat sumber daya berikut:

- AWS Command Line Interface
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK untuk JavaScript
- AWSSDK for PHP V3
- AWSSDK untuk Python
- AWSSDK for Ruby V3

Harga untuk AWS AppConfig

Harga untuk AWS AppConfig pay-as-you-go didasarkan pada data konfigurasi dan pengambilan bendera fitur. Kami merekomendasikan menggunakan AWS AppConfig Agen untuk membantu mengoptimalkan biaya. Untuk informasi selengkapnya, lihat AWS Systems Manager Harga.

Kuota AWS AppConfig

Informasi tentang AWS AppConfig titik akhir dan kuota layanan bersama dengan kuota Systems Manager lainnya ada di. Referensi Umum Amazon Web

> **Note**
>
> Untuk informasi tentang kuota untuk layanan yang menyimpan AWS AppConfig konfigurasi, lihat. Tentang kuota dan batasan penyimpanan konfigurasi

Menyiapkan AWS AppConfig

Jika Anda belum melakukannya, daftar Akun AWS dan buat pengguna administratif.

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

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

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

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

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

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

Masuk sebagai pengguna dengan akses administratif

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

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

Tetapkan akses ke pengguna tambahan

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

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

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

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

Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDK, alat, dan AWS API, lihat autentikasi Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat.
IAM	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensil jangka panjang untuk menandatangani permintaan terprogram	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
	ke AWS CLI, AWS SDK, atau API. AWS	• Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensi jangka panjang di Panduan Referensi AWS SDK dan Alat. • Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

(Opsional) Konfigurasikan izin untuk rollback berdasarkan alarm CloudWatch

Anda dapat mengonfigurasi AWS AppConfig untuk memutar kembali ke versi konfigurasi sebelumnya sebagai respons terhadap satu atau beberapa CloudWatch alarm Amazon. Saat mengonfigurasi penerapan untuk merespons CloudWatch alarm, Anda menentukan peran AWS Identity and Access Management (IAM). AWS AppConfig membutuhkan peran ini sehingga dapat memantau CloudWatch alarm.

> **Note**
>
> Peran IAM harus milik akun saat ini. Secara default, hanya AWS AppConfig dapat memantau alarm yang dimiliki oleh akun saat ini. Jika Anda ingin mengonfigurasi AWS AppConfig untuk memutar kembali penerapan sebagai respons terhadap metrik dari akun yang berbeda, Anda

harus mengonfigurasi alarm lintas akun. Untuk informasi selengkapnya, lihat CloudWatch Konsol lintas wilayah lintas akun di CloudWatch Panduan Pengguna Amazon.

Gunakan prosedur berikut untuk membuat peran IAM yang memungkinkan AWS AppConfig untuk melakukan rollback berdasarkan alarm. CloudWatch Bagian ini mencakup prosedur berikut.

1. Langkah 1: Buat kebijakan izin untuk rollback berdasarkan alarm CloudWatch
2. Langkah 2: Buat peran IAM untuk rollback berdasarkan alarm CloudWatch
3. Langkah 3: Tambahkan hubungan kepercayaan

Langkah 1: Buat kebijakan izin untuk rollback berdasarkan alarm CloudWatch

Gunakan prosedur berikut untuk membuat kebijakan IAM yang memberikan AWS AppConfig izin untuk memanggil tindakan `DescribeAlarms` API.

Untuk membuat kebijakan izin IAM untuk rollback berdasarkan alarm CloudWatch

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan dan kemudian pilih Buat kebijakan.
3. Di halaman Buat kebijakan, pilih tab JSON.
4. Ganti konten default pada tab JSON dengan kebijakan izin berikut, lalu pilih Berikutnya: Tag.

> **Note**
>
> Untuk mengembalikan informasi tentang alarm CloudWatch komposit, operasi DescribeAlarmsAPI harus diberi * izin, seperti yang ditunjukkan di sini. Anda tidak dapat mengembalikan informasi tentang alarm komposit jika `DescribeAlarms` memiliki cakupan yang lebih sempit.

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

```
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:DescribeAlarms"
                ],
                "Resource": "*"
            }
        ]
    }
```

5. Masukkan tag untuk peran ini, lalu pilih Berikutnya: Tinjau.
6. Pada halaman Ulasan, masukkan **`SSMCloudWatchAlarmDiscoveryPolicy`** di bidang Nama.
7. Pilih Buat kebijakan. Sistem mengembalikan Anda ke halaman Kebijakan.

Langkah 2: Buat peran IAM untuk rollback berdasarkan alarm CloudWatch

Gunakan prosedur berikut untuk membuat peran IAM dan tetapkan kebijakan yang Anda buat di prosedur sebelumnya.

Untuk membuat peran IAM untuk rollback berdasarkan alarm CloudWatch

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Di bawah Pilih jenis entitas terpercaya, pilih AWS layanan.
4. Segera di bawah Pilih layanan yang akan menggunakan peran ini, pilih EC2: Mengizinkan instans EC2 memanggil AWS layanan atas nama Anda, lalu pilih Berikutnya: Izin.
5. Pada halaman Kebijakan izin terlampir, cari CloudWatchAlarmDiscoveryPolicySSM.
6. Pilih kebijakan ini dan kemudian pilih Berikutnya: Tag.
7. Masukkan tag untuk peran ini, lalu pilih Berikutnya: Tinjau.
8. Pada halaman Buat peran, masukkan **`SSMCloudWatchAlarmDiscoveryRole`** di bidang Nama peran, lalu pilih Buat peran.
9. Pada halaman Peran, pilih peran yang baru saja Anda buat. Halaman Ringkasan terbuka.

Langkah 3: Tambahkan hubungan kepercayaan

Gunakan prosedur berikut untuk mengonfigurasi peran yang baru saja Anda buat agar dipercaya AWS AppConfig.

Untuk menambah hubungan kepercayaan untuk AWS AppConfig

1. Di halaman Ringkasan untuk peran yang baru saja Anda buat, pilih tab Trust Relationships, lalu pilih Edit Trust Relationship.
2. Edit kebijakan untuk menyertakan hanya "`appconfig.amazonaws.com`“, seperti yang ditunjukkan pada contoh berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Pilih Perbarui Kebijakan Kepercayaan.

Membuat flag fitur dan data konfigurasi formulir gratis di AWS AppConfig

Topik di bagian ini membantu Anda menyelesaikan tugas-tugas berikut AWS AppConfig. Tugas-tugas ini membuat artefak penting untuk menyebarkan data konfigurasi.

1. Buat namespace aplikasi

   Untuk membuat namespace aplikasi, Anda membuat AWS AppConfig artefak yang disebut aplikasi. Aplikasi hanyalah konstruksi organisasi seperti folder.

2. Buat lingkungan

   Untuk setiap AWS AppConfig aplikasi, Anda menentukan satu atau lebih lingkungan. Lingkungan adalah kelompok penyebaran AWS AppConfig target yang logis, seperti aplikasi di `Production` lingkungan `Beta` atau. Anda juga dapat menentukan lingkungan untuk subkomponen aplikasi, sepertiAWS Lambda functions,,Containers, WebMobile, danBack-end.

   Anda dapat mengonfigurasi CloudWatch alarm Amazon untuk setiap lingkungan untuk secara otomatis mengembalikan perubahan konfigurasi yang bermasalah. Sistem memantau alarm selama deployment konfigurasi. Jika alarm dipicu, sistem memutar kembali konfigurasi.

3. Buat profil konfigurasi

   Profil konfigurasi mencakup, antara lain, URI yang memungkinkan AWS AppConfig untuk menemukan data konfigurasi Anda di lokasi yang disimpan dan jenis profil. AWS AppConfig mendukung dua jenis profil konfigurasi: bendera fitur dan konfigurasi bentuk bebas. Profil konfigurasi bendera fitur menyimpan data mereka di toko konfigurasi yang AWS AppConfig dihosting dan URI sederhanahosted. Untuk profil konfigurasi bentuk bebas, Anda dapat menyimpan data Anda di penyimpanan konfigurasi yang AWS AppConfig dihosting atau kemampuan atau AWS layanan Systems Manager lain yang terintegrasi AWS AppConfig, seperti yang dijelaskan dalam. Membuat profil konfigurasi formulir gratis di AWS AppConfig

   Profil konfigurasi juga dapat menyertakan validator opsional untuk memastikan data konfigurasi Anda benar secara sintaksis dan semantik. AWS AppConfig melakukan pemeriksaan menggunakan validator saat Anda memulai penerapan. Jika ada kesalahan yang terdeteksi, penerapan berhenti sebelum membuat perubahan apa pun pada target konfigurasi.

> **Note**
>
> Kecuali Anda memiliki kebutuhan khusus untuk menyimpan rahasia di AWS Secrets Manager atau mengelola data di Amazon Simple Storage Service (Amazon S3), sebaiknya hosting data konfigurasi Anda di AWS AppConfig toko konfigurasi yang dihosting karena menawarkan fitur dan penyempurnaan terbanyak.

Topik

Contoh konfigurasi

Gunakan AWS AppConfig, kemampuan AWS Systems Manager, untuk membuat, mengelola, dan dengan cepat menyebarkan konfigurasi aplikasi. Konfigurasi adalah kumpulan pengaturan yang memengaruhi perilaku aplikasi Anda. Berikut ini adalah beberapa contoh.

Konfigurasi bendera fitur

Konfigurasi tanda fitur berikut memungkinkan atau menonaktifkan pembayaran seluler dan pembayaran default berdasarkan per wilayah.

JSON

```
{
  "allow_mobile_payments": {
    "enabled": false
  },
  "default_payments_per_region": {
    "enabled": true
  }
}
```

YAML

```
---
allow_mobile_payments:
  enabled: false
default_payments_per_region:
  enabled: true
```

Konfigurasi operasional

Konfigurasi bentuk bebas berikut memberlakukan batasan tentang cara aplikasi memproses permintaan.

JSON

```
{
  "throttle-limits": {
    "enabled": "true",
    "throttles": [
      {
        "simultaneous_connections": 12
      },
      {
        "tps_maximum": 5000
      }
    ],
    "limit-background-tasks": [
      true
    ]
  }
}
```

YAML

```
---
throttle-limits:
  enabled: 'true'
  throttles:
  - simultaneous_connections: 12
  - tps_maximum: 5000
  limit-background-tasks:
```

```
  - true
```

Konfigurasi daftar kontrol akses

Konfigurasi bentuk bebas daftar kontrol akses berikut menentukan pengguna atau grup mana yang dapat mengakses aplikasi.

JSON

```
{
  "allow-list": {
    "enabled": "true",
    "cohorts": [
      {
        "internal_employees": true
      },
      {
        "beta_group": false
      },
      {
        "recent_new_customers": false
      },
      {
        "user_name": "Jane_Doe"
      },
      {
        "user_name": "John_Doe"
      }
    ]
  }
}
```

YAML

```
---
allow-list:
  enabled: 'true'
  cohorts:
  - internal_employees: true
  - beta_group: false
  - recent_new_customers: false
  - user_name: Jane_Doe
```

```
  - user_name: Ashok_Kumar
```

Tentang peran IAM profil konfigurasi

Anda dapat membuat peran IAM yang menyediakan akses ke data konfigurasi dengan menggunakan AWS AppConfig. Atau Anda dapat membuat peran IAM sendiri. Jika Anda membuat peran menggunakan AWS AppConfig, sistem akan membuat peran dan menentukan salah satu kebijakan izin berikut, tergantung pada jenis sumber konfigurasi yang Anda pilih.

Sumber konfigurasi adalah rahasia Secrets Manager

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
             ],
            "Resource": [
                "arn:aws:secretsmanager:Wilayah AWS:account_ID:secret:secret_name-
a1b2c3"
            ]
        }
    ]
}
```

Sumber konfigurasi adalah parameter Parameter Store

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
                "arn:aws:ssm:Wilayah AWS:account_ID:parameter/parameter_name"
            ]
        }
```

```
    ]
    }
```

Sumber konfigurasi adalah dokumen SSM

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument"
            ],
            "Resource": [
                "arn:aws:ssm:Wilayah AWS:account_ID:document/document_name"
            ]
        }
    ]
}
```

Jika Anda membuat peran dengan menggunakan AWS AppConfig, sistem juga menciptakan hubungan kepercayaan berikut untuk peran tersebut.

```
{

  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Membuat namespace untuk aplikasi Anda di AWS AppConfig

Prosedur di bagian ini membantu Anda membuat AWS AppConfig artefak yang disebut aplikasi. Aplikasi hanyalah konstruksi organisasi seperti folder yang mengidentifikasi namespace aplikasi Anda. Konstruksi organisasi ini memiliki hubungan dengan beberapa unit kode yang dapat dieksekusi. Misalnya, Anda dapat membuat aplikasi yang dipanggil MyMobileApp untuk mengatur dan mengelola data konfigurasi untuk aplikasi seluler yang diinstal oleh pengguna Anda. Anda harus membuat artefak ini sebelum dapat digunakan AWS AppConfig untuk menyebarkan dan mengambil bendera fitur atau data konfigurasi formulir gratis.

> **Note**
>
> Anda dapat menggunakan AWS CloudFormation untuk membuat AWS AppConfig artefak, termasuk aplikasi, lingkungan, profil konfigurasi, penerapan, strategi penerapan, dan versi konfigurasi yang dihosting. Untuk informasi selengkapnya, lihat referensi jenis AWS AppConfig sumber daya di Panduan AWS CloudFormation Pengguna.

Topik

Membuat AWS AppConfig aplikasi (konsol)

Gunakan prosedur berikut untuk membuat AWS AppConfig aplikasi dengan menggunakan AWS Systems Manager konsol.

Untuk membuat aplikasi

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih Buat aplikasi.
3. Untuk Nama, masukkan nama untuk aplikasi.
4. Untuk Deskripsi, masukkan informasi tentang aplikasi.
5. (Opsional) Di bagian Ekstensi, pilih ekstensi dari daftar. Untuk informasi selengkapnya, lihat Tentang AWS AppConfig ekstensi.

6. (Opsional) Di tag bagian, masukkan kunci dan nilai opsional. Anda dapat menentukan maksimum 50 tag untuk sumber daya.
7. Pilih Create application (Buat aplikasi).

AWS AppConfig membuat aplikasi dan kemudian menampilkan tab Lingkungan. Lanjut ke Membuat lingkungan untuk aplikasi Anda di AWS AppConfig.

Membuat AWS AppConfig aplikasi (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS CLI (di Linux atau Windows) atau AWS Tools for PowerShell untuk membuat AWS AppConfig aplikasi.

Untuk membuat aplikasi langkah demi langkah

1. Buka AWS CLI.
2. Jalankan perintah berikut untuk membuat aplikasi.

   Linux

   ```
   aws appconfig create-application \
     --name A_name_for_the_application \
     --description A_description_of_the_application \
     --tags User_defined_key_value_pair_metadata_for_the_application
   ```

   Windows

   ```
   aws appconfig create-application ^
     --name A_name_for_the_application ^
     --description A_description_of_the_application ^
     --tags User_defined_key_value_pair_metadata_for_the_application
   ```

   PowerShell

   ```
   New-APPCApplication `
     -Name Name_for_the_application `
     -Description Description_of_the_application `
     -Tag Hashtable_type_user_defined_key_value_pair_metadata_for_the_application
   ```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "Id": "Application ID",
   "Name": "Application name",
   "Description": "Description of the application"
}
```

Windows

```
{
   "Id": "Application ID",
   "Name": "Application name",
   "Description": "Description of the application"
}
```

PowerShell

```
ContentLength    : Runtime of the command
Description      : Description of the application
HttpStatusCode   : HTTP Status of the runtime
Id               : Application ID
Name             : Application name
ResponseMetadata : Runtime Metadata
```

Membuat lingkungan untuk aplikasi Anda di AWS AppConfig

Untuk setiap AWS AppConfig aplikasi, Anda menentukan satu atau lebih lingkungan. Lingkungan adalah kelompok penyebaran logis AppConfig target, seperti aplikasi dalam `Beta` atau `Production` lingkungan, AWS Lambda fungsi, atau kontainer. Anda juga dapat menentukan lingkungan untuk subkomponen aplikasi, seperti`Web`,`Mobile`, dan`Back-end`. Anda dapat mengonfigurasi CloudWatch alarm Amazon untuk setiap lingkungan. Sistem memantau alarm selama deployment konfigurasi. Jika alarm dipicu, sistem memutar kembali konfigurasi.

Sebelum Anda Memulai

Jika Anda ingin mengaktifkan AWS AppConfig untuk memutar kembali konfigurasi sebagai respons terhadap CloudWatch alarm, maka Anda harus mengonfigurasi peran AWS Identity and Access Management (IAM) dengan izin AWS AppConfig untuk mengaktifkan respons alarm. CloudWatch Anda memilih peran ini dalam prosedur berikut. Untuk informasi selengkapnya, lihat (Opsional) Konfigurasikan izin untuk rollback berdasarkan alarm CloudWatch .

Topik

- Menciptakan AWS AppConfig lingkungan (konsol)
- Menciptakan AWS AppConfig lingkungan (baris perintah)

Menciptakan AWS AppConfig lingkungan (konsol)

Gunakan prosedur berikut untuk membuat AWS AppConfig lingkungan dengan menggunakan AWS Systems Manager konsol.

Untuk membuat lingkungan

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih nama aplikasi untuk membuka halaman detail.
3. Pilih tab Lingkungan, lalu pilih Buat lingkungan.
4. Untuk Nama, masukkan nama untuk lingkungan.
5. Untuk Deskripsi, masukkan informasi tentang lingkungan.
6. (Opsional) Di bagian Monitor, pilih bidang peran IAM, lalu pilih peran IAM dengan izin untuk memutar kembali konfigurasi jika alarm dipicu.
7. Dalam daftar CloudWatch alarm, pilih satu atau beberapa alarm untuk dipantau. AWS AppConfig memutar kembali penerapan konfigurasi Anda jika salah satu alarm ini masuk ke status alarm.
8. (Opsional) Di bagian Ekstensi asosiasi, pilih ekstensi dari daftar. Untuk informasi selengkapnya, lihat Tentang AWS AppConfig ekstensi.
9. (Opsional) Di tag bagian, masukkan kunci dan nilai opsional. Anda dapat menentukan maksimum 50 tag untuk sumber daya.
10. Pilih Buat lingkungan.

AWS AppConfig menciptakan lingkungan dan kemudian menampilkan halaman detail Lingkungan. Lanjut ke Membuat profil konfigurasi di AWS AppConfig.

Menciptakan AWS AppConfig lingkungan (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS CLI (di Linux atau Windows) atau AWS Tools for PowerShell untuk membuat AWS AppConfig lingkungan.

Untuk membuat lingkungan selangkah demi selangkah

1. Buka AWS CLI.
2. Jalankan perintah berikut untuk membuat lingkungan.

   Linux

   ```
   aws appconfig create-environment \
     --application-id The_application_ID \
     --name A_name_for_the_environment \
     --description A_description_of_the_environment \
     --monitors
    "AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
    role_for_AWS AppConfig_to_monitor_AlarmArn" \
     --tags User_defined_key_value_pair_metadata_of_the_environment
   ```

   Windows

   ```
   aws appconfig create-environment ^
     --application-id The_application_ID ^
     --name A_name_for_the_environment ^
     --description A_description_of_the_environment ^
     --monitors
    "AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
    role_for_AWS AppConfig_to_monitor_AlarmArn" ^
     --tags User_defined_key_value_pair_metadata_of_the_environment
   ```

   PowerShell

   ```
   New-APPCEnvironment `
     -Name Name_for_the_environment `
     -ApplicationId The_application_ID
     -Description Description_of_the_environment `
     -Monitors
    @{"AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
    role_for_AWS AppConfig_to_monitor_AlarmArn"} `
   ```

```
  -Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_environment
```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "ApplicationId": "The application ID",
   "Id": "The_environment ID",
   "Name": "Name of the environment",
   "State": "The state of the environment",
   "Description": "Description of the environment",

   "Monitors": [
      {
         "AlarmArn": "ARN of the Amazon CloudWatch alarm",
         "AlarmRoleArn": "ARN of the IAM role for AppConfig to monitor AlarmArn"
      }
   ]
}
```

Windows

```
{
   "ApplicationId": "The application ID",
   "Id": "The environment ID",
   "Name": "Name of the environment",
   "State": "The state of the environment"
   "Description": "Description of the environment",

   "Monitors": [
      {
         "AlarmArn": "ARN of the Amazon CloudWatch alarm",
         "AlarmRoleArn": "ARN of the IAM role for AppConfig to monitor AlarmArn"
      }
   ]
}
```

PowerShell

```
ApplicationId     : The application ID
```

```
ContentLength     : Runtime of the command
Description       : Description of the environment
HttpStatusCode    : HTTP Status of the runtime
Id                : The environment ID
Monitors          : {ARN of the Amazon CloudWatch alarm, ARN of the IAM role for
 AppConfig to monitor AlarmArn}
Name              : Name of the environment
Response Metadata : Runtime Metadata
State             : State of the environment
```

Lanjut ke Membuat profil konfigurasi di AWS AppConfig.

Membuat profil konfigurasi di AWS AppConfig

Profil konfigurasi mencakup, antara lain, URI yang memungkinkan AWS AppConfig untuk menemukan data konfigurasi Anda di lokasi yang disimpan dan jenis konfigurasi. AWS AppConfig mendukung dua jenis profil konfigurasi: bendera fitur dan konfigurasi bentuk bebas. Konfigurasi flag fitur menyimpan data di toko konfigurasi yang AWS AppConfig dihosting dan URI sederhana`hosted`. Konfigurasi bentuk bebas dapat menyimpan data di penyimpanan konfigurasi yang AWS AppConfig dihosting, berbagai kemampuan Systems Manager, atau AWS layanan yang terintegrasi dengannya. AWS AppConfig Untuk informasi selengkapnya, lihat Membuat profil konfigurasi formulir gratis di AWS AppConfig.

Profil konfigurasi juga dapat menyertakan validator opsional untuk memastikan data konfigurasi Anda benar secara sintaksis dan semantik. AWS AppConfig melakukan pemeriksaan menggunakan validator saat Anda memulai penerapan. Jika ada kesalahan yang terdeteksi, penerapan berhenti sebelum membuat perubahan apa pun pada target konfigurasi.

> **Note**
>
> Jika memungkinkan, kami sarankan untuk menghosting data konfigurasi Anda di toko konfigurasi yang AWS AppConfig dihosting karena menawarkan sebagian besar fitur dan penyempurnaan.

Topik

- Tentang validator
- Membuat profil konfigurasi bendera fitur di AWS AppConfig

- Membuat profil konfigurasi formulir gratis di AWS AppConfig

Tentang validator

Saat Anda membuat profil konfigurasi, Anda memiliki opsi untuk menentukan hingga dua validator. Validator memastikan bahwa data konfigurasi Anda benar secara sintaksis dan semantik. Jika Anda berencana untuk menggunakan validator, Anda harus membuatnya sebelum Anda membuat profil konfigurasi. AWS AppConfig mendukung jenis validator berikut:

- AWS Lambda fungsi: Didukung untuk bendera fitur dan konfigurasi formulir gratis.
- Skema JSON: Didukung untuk konfigurasi formulir gratis. (AWS AppConfig secara otomatis memvalidasi flag fitur terhadap Skema JSON.)

Topik

- AWS Lambda validator fungsi
- Validator Skema JSON

AWS Lambda validator fungsi

Validator fungsi Lambda harus dikonfigurasi dengan skema acara berikut. AWS AppConfig menggunakan skema ini untuk memanggil fungsi Lambda. Konten adalah string yang dikodekan base64, dan URI adalah string.

```
{
    "applicationId": "The application ID of the configuration profile being
 validated",
    "configurationProfileId": "The ID of the configuration profile being validated",
    "configurationVersion": "The version of the configuration profile being validated",
    "content": "Base64EncodedByteString",
    "uri": "The configuration uri"
}
```

AWS AppConfig memverifikasi bahwa header `X-Amz-Function-Error` Lambda diatur dalam respons. Lambda menetapkan header ini jika fungsi melempar pengecualian. Untuk informasi selengkapnya`X-Amz-Function-Error`, lihat Penanganan Kesalahan dan Percobaan Ulang Otomatis AWS Lambda di Panduan AWS Lambda Pengembang.

Berikut adalah contoh sederhana dari kode respons Lambda untuk validasi yang berhasil.

```
import json

def handler(event, context):
     #Add your validation logic here
     print("We passed!")
```

Berikut adalah contoh sederhana dari kode respons Lambda untuk validasi yang gagal.

```
def handler(event, context):
     #Add your validation logic here
     raise Exception("Failure!")
```

Berikut adalah contoh lain yang memvalidasi hanya jika parameter konfigurasi adalah bilangan prima.

```
function isPrime(value) {
    if (value < 2) {
        return false;
    }

    for (i = 2; i < value; i++) {
        if (value % i === 0) {
            return false;
        }
    }

    return true;
}

exports.handler = async function(event, context) {
    console.log('EVENT: ' + JSON.stringify(event, null, 2));
    const input = parseInt(Buffer.from(event.content, 'base64').toString('ascii'));
    const prime = isPrime(input);
    console.log('RESULT: ' + input + (prime ? ' is' : ' is not') + ' prime');
    if (!prime) {
        throw input + "is not prime";
    }
}
```

AWS AppConfig memanggil Lambda validasi Anda saat memanggil operasi `ValidateConfigurationActivity` dan `StartDeployment` API. Anda harus memberikan `appconfig.amazonaws.com` izin untuk memanggil Lambda Anda. Untuk informasi selengkapnya,

lihat Memberikan Akses Fungsi ke AWS Layanan. AWS AppConfig membatasi validasi waktu berjalan Lambda hingga 15 detik, termasuk latensi start-up.

Validator Skema JSON

Jika Anda membuat konfigurasi dalam dokumen SSM, maka Anda harus menentukan atau membuat Skema JSON untuk konfigurasi itu. Skema JSON mendefinisikan properti yang diizinkan untuk setiap pengaturan konfigurasi aplikasi. Skema JSON berfungsi seperti seperangkat aturan untuk memastikan bahwa pengaturan konfigurasi baru atau yang diperbarui sesuai dengan praktik terbaik yang diperlukan oleh aplikasi Anda. Inilah contohnya.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "$id$",
  "description": "BasicFeatureToggle-1",
  "type": "object",
  "additionalProperties": false,
  "patternProperties": {
      "[^\\s]+$": {
          "type": "boolean"
      }
  },
  "minProperties": 1
}
```

Saat Anda membuat konfigurasi dari dokumen SSM, sistem secara otomatis memverifikasi bahwa konfigurasi sesuai dengan persyaratan skema. Jika tidak, AWS AppConfig mengembalikan kesalahan validasi.

Important

Perhatikan informasi penting berikut tentang validator Skema JSON:

- Data konfigurasi yang disimpan dalam dokumen SSM harus memvalidasi terhadap Skema JSON terkait sebelum Anda dapat menambahkan konfigurasi ke sistem. Parameter SSM tidak memerlukan metode validasi, tetapi kami menyarankan Anda membuat pemeriksaan validasi untuk konfigurasi parameter SSM baru atau yang diperbarui dengan menggunakan. AWS Lambda

- Konfigurasi dalam dokumen SSM menggunakan jenis `ApplicationConfiguration` dokumen. Skema JSON yang sesuai, menggunakan jenis `ApplicationConfigurationSchema` dokumen.
- AWS AppConfig mendukung JSON Schema versi 4.X untuk skema inline. Jika konfigurasi aplikasi Anda memerlukan versi JSON Schema yang berbeda, maka Anda harus membuat validator Lambda.

Membuat profil konfigurasi bendera fitur di AWS AppConfig

Anda dapat menggunakan bendera fitur untuk mengaktifkan atau menonaktifkan fitur dalam aplikasi Anda atau untuk mengonfigurasi karakteristik yang berbeda dari fitur aplikasi Anda menggunakan atribut bendera. AWS AppConfig menyimpan konfigurasi bendera fitur di penyimpanan konfigurasi yang AWS AppConfig dihosting dalam format bendera fitur yang berisi data dan metadata tentang bendera Anda dan atribut bendera. Untuk informasi selengkapnya tentang penyimpanan konfigurasi yang AWS AppConfig dihosting, lihat Tentang toko konfigurasi yang AWS AppConfig dihosting bagian.

Topik

- Membuat profil konfigurasi bendera fitur (konsol)
- Membuat bendera fitur dan profil konfigurasi bendera fitur (baris perintah)
- Jenis referensi untuk AWS.AppConfig.FeatureFlags

Sebelum Anda mulai

Dalam prosedur berikut, di bagian Enkripsi opsional, Anda dapat memilih kunci AWS Key Management Service (AWS KMS). Kunci yang dikelola pelanggan ini memungkinkan Anda mengenkripsi versi data konfigurasi baru di toko konfigurasi yang AWS AppConfig dihosting. Untuk informasi selengkapnya tentang kunci ini, lihat AWS AppConfig mendukung kunci manajer pelangganKeamanan di AWS AppConfig.

Prosedur berikut juga memberi Anda opsi untuk mengaitkan ekstensi dengan profil konfigurasi bendera fitur. Ekstensi menambah kemampuan Anda untuk menyuntikkan logika atau perilaku pada titik yang berbeda selama AWS AppConfig alur kerja membuat atau menerapkan konfigurasi. Untuk informasi selengkapnya, lihat Tentang AWS AppConfig ekstensi.

Terakhir, di bagian atribut bendera Fitur, saat Anda memasukkan detail atribut dari bendera fitur baru, Anda dapat menentukan batasan. Batasan memastikan bahwa nilai atribut yang tidak terduga tidak diterapkan ke aplikasi Anda. AWS AppConfig mendukung jenis atribut bendera berikut dan kendala yang sesuai.

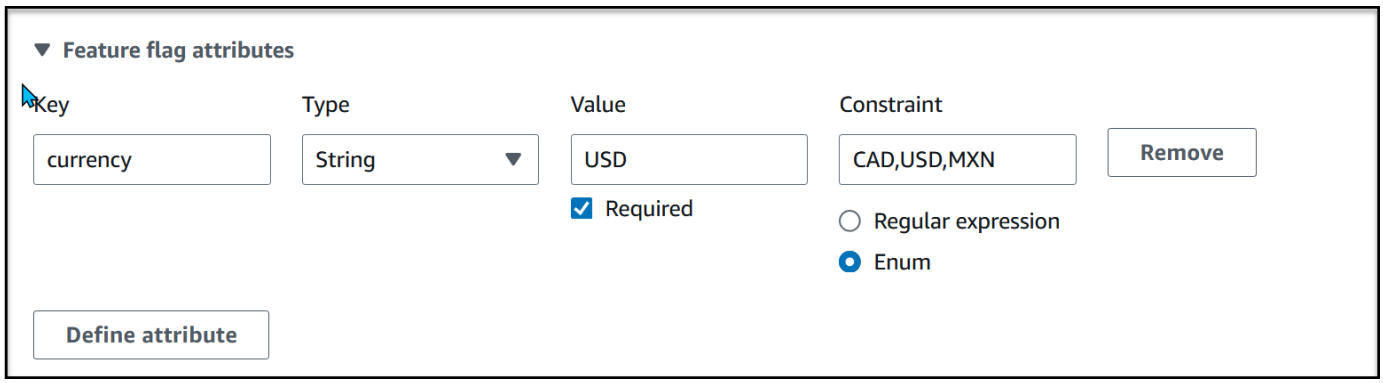
Tipe	Kendala	Deskripsi
Tali	Ekspresi reguler	Pola Regex untuk string
	Enum	Daftar nilai yang dapat diterima untuk string
Nomor	Minimum	Nilai numerik minimum untuk atribut
	Maksimum	Nilai numerik maksimum untuk atribut
Boolean	Tidak ada	Tidak ada
Array string	Ekspresi reguler	Pola Regex untuk elemen array
	Enum	Daftar nilai yang dapat diterima untuk elemen array
Array nomor	Minimum	Nilai numerik minimum untuk elemen array
	Maksimum	Nilai numerik maksimum untuk elemen array

Membuat profil konfigurasi bendera fitur (konsol)

Gunakan prosedur berikut untuk membuat profil konfigurasi tanda AWS AppConfig fitur dengan menggunakan AWS AppConfig konsol.

Untuk membuat profil konfigurasi

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi yang Anda buat. Membuat namespace untuk aplikasi Anda di AWS AppConfig
3. Pilih tab Configuration profiles and feature flags, lalu pilih Create configuration.
4. Di bagian Opsi konfigurasi, pilih Bendera fitur.
5. Gulir ke bawah. Di bagian Profil konfigurasi, untuk nama profil Konfigurasi, masukkan nama.
6. (Opsional) Perluas Deskripsi dan masukkan deskripsi.
7. (Opsional) Perluas Opsi tambahan dan selesaikan yang berikut ini, seperlunya.
   a. Dalam daftar Enkripsi, pilih tombol AWS Key Management Service (AWS KMS) dari daftar.
   b. Di bagian Ekstensi asosiasi, pilih ekstensi dari daftar.
   c. Di bagian Tag, pilih Tambahkan tag baru, lalu tentukan kunci dan nilai opsional.
8. Pilih Selanjutnya.
9. Di bagian Definisi bendera fitur, untuk nama Bendera, masukkan nama.
10. Untuk tombol Bendera masukkan pengenal bendera untuk membedakan bendera dalam profil konfigurasi yang sama. Bendera dalam profil konfigurasi yang sama tidak dapat memiliki kunci yang sama. Setelah bendera dibuat, Anda dapat mengedit nama bendera, tetapi bukan tombol bendera.
11. (Opsional) Perluas Deskripsi dan masukkan informasi tentang bendera ini.
12. Pilih Ini adalah bendera jangka pendek dan secara opsional memilih tanggal kapan bendera harus dinonaktifkan atau dihapus. Perhatikan bahwa AWS AppConfig tidak menonaktifkan bendera.
13. Di bagian atribut Bendera, pilih Tentukan atribut. Atribut memungkinkan Anda untuk memberikan nilai tambahan dalam flag Anda.
14. Untuk Kunci, tentukan tombol bendera dan pilih jenisnya dari daftar Jenis. Anda dapat secara opsional memvalidasi nilai atribut terhadap kendala tertentu. Gambar berikut menunjukkan sebuah contoh.



Pilih Tentukan atribut untuk menambahkan atribut tambahan.

> **Note**
>
> Perhatikan informasi berikut.
>
> - Untuk nama atribut, kata “diaktifkan” dicadangkan. Anda tidak dapat membuat atribut flag fitur yang disebut “enabled”. Tidak ada kata-kata lain yang dicadangkan.
> - Atribut flag fitur hanya disertakan dalam `GetLatestConfiguration` respons jika flag tersebut diaktifkan.
> - Kunci atribut bendera untuk bendera tertentu harus unik.
> - Pilih Nilai yang diperlukan untuk menentukan apakah nilai atribut diperlukan.

15. Di bagian Nilai tanda fitur, pilih Diaktifkan untuk mengaktifkan bendera. Gunakan sakelar yang sama ini untuk menonaktifkan tanda saat mencapai tanggal penghentian yang ditentukan, jika berlaku.
16. Pilih Selanjutnya.
17. Pada halaman Tinjau dan simpan, verifikasi detail bendera dan kemudian Simpan dan lanjutkan untuk menerapkan.

Lanjut ke Menyebarkan flag fitur dan data konfigurasi di AWS AppConfig.

Membuat bendera fitur dan profil konfigurasi bendera fitur (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS Command Line Interface (di Linux atau Windows) atau Alat untuk Windows PowerShell untuk membuat profil konfigurasi bendera AWS AppConfig fitur. Jika mau, Anda dapat menggunakan AWS CloudShell untuk menjalankan perintah

yang tercantum di bawah ini. Untuk informasi selengkapnya, lihat Apa itu AWS CloudShell? dalam AWS CloudShell Panduan Pengguna.

Untuk membuat konfigurasi flag fitur langkah demi langkah

1. Buka AWS CLI.
2. Buat profil konfigurasi bendera fitur yang menentukan Tipe sebagai`AWS.AppConfig.FeatureFlags`. Profil konfigurasi harus digunakan `hosted` untuk URI lokasi.

   Linux

   ```
   aws appconfig create-configuration-profile \
     --application-id The_application_ID \
     --name A_name_for_the_configuration_profile \
     --location-uri hosted \
     --type AWS.AppConfig.FeatureFlags
   ```

   Windows

   ```
   aws appconfig create-configuration-profile ^
     --application-id The_application_ID ^
     --name A_name_for_the_configuration_profile ^
     --location-uri hosted ^
     --type AWS.AppConfig.FeatureFlags
   ```

   PowerShell

   ```
   New-APPCConfigurationProfile `
     -Name A_name_for_the_configuration_profile `
     -ApplicationId The_application_ID `
     -LocationUri hosted `
     -Type AWS.AppConfig.FeatureFlags
   ```

3. Buat data konfigurasi bendera fitur Anda. Data Anda harus dalam format JSON dan sesuai dengan skema `AWS.AppConfig.FeatureFlags` JSON. Untuk informasi lebih lanjut tentang skema, lihatJenis referensi untuk AWS.AppConfig.FeatureFlags.

4. Gunakan `CreateHostedConfigurationVersion` API untuk menyimpan data konfigurasi flag fitur Anda AWS AppConfig.

   Linux

   ```
   aws appconfig create-hosted-configuration-version \
     --application-id The_application_ID \
     --configuration-profile-id The_configuration_profile_id \
     --content-type "application/json" \
     --content file://path/to/feature_flag_configuration_data \
     file_name_for_system_to_store_configuration_data
   ```

   Windows

   ```
   aws appconfig create-hosted-configuration-version ^
     --application-id The_application_ID ^
     --configuration-profile-id The_configuration_profile_id ^
     --content-type "application/json" ^
     --content file://path/to/feature_flag_configuration_data ^
     file_name_for_system_to_store_configuration_data
   ```

   PowerShell

   ```
   New-APPCHostedConfigurationVersion `
     -ApplicationId The_application_ID `
     -ConfigurationProfileId The_configuration_profile_id `
     -ContentType "application/json" `
     -Content file://path/to/feature_flag_configuration_data `
     file_name_for_system_to_store_configuration_data
   ```

   Berikut adalah contoh perintah Linux.

   ```
   aws appconfig create-hosted-configuration-version \
     --application-id 1a2b3cTestApp \
     --configuration-profile-id 4d5e6fTestConfigProfile \
     --content-type "application/json" \
     --content Base64Content
   ```

   `content`Parameter menggunakan data yang `base64` dikodekan berikut.

```
{
  "flags": {
    "flagkey": {
      "name": "WinterSpecialBanner"
    }
  },
  "values": {
    "flagkey": {
      "enabled": true
    }
  },
  "version": "1"
}
```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "ApplicationId"          : "1a2b3cTestApp",
   "ConfigurationProfileId" : "4d5e6fTestConfigProfile",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
}
```

Windows

```
{
   "ApplicationId"          : "1a2b3cTestApp",
   "ConfigurationProfileId" : "4d5e6fTestConfigProfile",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
}
```

PowerShell

```
ApplicationId          : 1a2b3cTestApp
ConfigurationProfileId : 4d5e6fTestConfigProfile
VersionNumber          : 1
ContentType            : application/json
```

`service_returned_content_file`Berisi data konfigurasi Anda yang mencakup beberapa AWS AppConfig metadata yang dihasilkan.

> **Note**
>
> Saat Anda membuat versi konfigurasi yang dihosting, AWS AppConfig verifikasi bahwa data Anda sesuai dengan skema `AWS.AppConfig.FeatureFlags` JSON. AWS AppConfig Selain itu memvalidasi bahwa setiap atribut flag fitur dalam data Anda memenuhi batasan yang Anda tetapkan untuk atribut tersebut.

Jenis referensi untuk AWS.AppConfig.FeatureFlags

Gunakan skema `AWS.AppConfig.FeatureFlags` JSON sebagai referensi untuk membuat data konfigurasi flag fitur Anda.

```
{
      "$schema": "http://json-schema.org/draft-07/schema#",
      "definitions": {
        "flagSetDefinition": {
          "type": "object",
          "properties": {
            "version": {
              "$ref": "#/definitions/flagSchemaVersions"
            },
            "flags": {
              "$ref": "#/definitions/flagDefinitions"
            },
            "values": {
              "$ref": "#/definitions/flagValues"
            }
          },
          "required": ["version", "flags"],
          "additionalProperties": false
        },
        "flagDefinitions": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-]{0,63}$": {
              "$ref": "#/definitions/flagDefinition"
```

```
            }
          },
          "maxProperties": 100,
          "additionalProperties": false
        },
        "flagDefinition": {
          "type": "object",
          "properties": {
            "name": {
              "$ref": "#/definitions/customerDefinedName"
            },
            "description": {
              "$ref": "#/definitions/customerDefinedDescription"
            },
            "_createdAt": {
              "type": "string"
            },
            "_updatedAt": {
              "type": "string"
            },
            "_deprecation": {
              "type": "object",
              "properties": {
                "status": {
                  "type": "string",
                  "enum": ["planned"]
                }
              },
             "additionalProperties": false
            },
            "attributes": {
              "$ref": "#/definitions/attributeDefinitions"
            }
          },
          "additionalProperties": false
        },
        "attributeDefinitions": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/attributeDefinition"
            }
          },
          "maxProperties": 25,
```

```
          "additionalProperties": false
        },
        "attributeDefinition": {
          "type": "object",
          "properties": {
            "description": {
              "$ref": "#/definitions/customerDefinedDescription"
            },
            "constraints": {
              "oneOf": [
                { "$ref": "#/definitions/numberConstraints" },
                { "$ref": "#/definitions/stringConstraints" },
                { "$ref": "#/definitions/arrayConstraints" },
                { "$ref": "#/definitions/boolConstraints" }
              ]
            }
          },
          "additionalProperties": false
        },
        "flagValues": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/flagValue"
            }
          },
          "maxProperties": 100,
          "additionalProperties": false
        },
        "flagValue": {
          "type": "object",
          "properties": {
            "enabled": {
              "type": "boolean"
            },
            "_createdAt": {
              "type": "string"
            },
            "_updatedAt": {
              "type": "string"
            }
          },
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
```

```
            "$ref": "#/definitions/attributeValue",
            "maxProperties": 25
          }
        },
        "required": ["enabled"],
        "additionalProperties": false
      },
      "attributeValue": {
        "oneOf": [
          { "type": "string", "maxLength": 1024 },
          { "type": "number" },
          { "type": "boolean" },
          {
            "type": "array",
            "oneOf": [
              {
                "items": {
                  "type": "string",
                  "maxLength": 1024
                }
              },
              {
                "items": {
                  "type": "number"
                }
              }
            ]
          }
        ],
        "additionalProperties": false
      },
      "stringConstraints": {
        "type": "object",
        "properties": {
          "type": {
            "type": "string",
            "enum": ["string"]
    },
          "required": {
            "type": "boolean"
          },
          "pattern": {
            "type": "string",
            "maxLength": 1024
```

```
            },
            "enum": {
              "type": "array",
              "maxLength": 100,
              "items": {
                "oneOf": [
                  {
                    "type": "string",
                    "maxLength": 1024
                  },
                  {
                    "type": "integer"
                  }
                ]
              }
            }
          },
          "required": ["type"],
          "not": {
            "required": ["pattern", "enum"]
          },
          "additionalProperties": false
        },
        "numberConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["number"]
      },
            "required": {
              "type": "boolean"
            },
            "minimum": {
              "type": "integer"
            },
            "maximum": {
              "type": "integer"
            }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "arrayConstraints": {
```

```
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["array"]
      },
            "required": {
              "type": "boolean"
            },
            "elements": {
              "$ref": "#/definitions/elementConstraints"
      }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "boolConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["boolean"]
      },
            "required": {
              "type": "boolean"
            }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "elementConstraints": {
          "oneOf": [
            { "$ref": "#/definitions/numberConstraints" },
            { "$ref": "#/definitions/stringConstraints" }
          ]
        },
        "customerDefinedName": {
          "type": "string",
          "pattern": "^[^\\n]{1,64}$"
        },
        "customerDefinedDescription": {
          "type": "string",
          "maxLength": 1024
        },
```

```
        "flagSchemaVersions": {
          "type": "string",
          "enum": ["1"]
        }
      },
      "type": "object",
      "$ref": "#/definitions/flagSetDefinition",
      "additionalProperties": false
    }
```

> **Important**
>
> Untuk mengambil data konfigurasi flag fitur, aplikasi Anda harus memanggil `GetLatestConfiguration` API. Anda tidak dapat mengambil data konfigurasi flag fitur dengan memanggil`GetConfiguration`, yang sudah usang. Untuk informasi selengkapnya, lihat GetLatestKonfigurasi di Referensi AWS AppConfig API.

Saat aplikasi Anda memanggil GetLatestKonfigurasi dan menerima konfigurasi yang baru diterapkan, informasi yang menentukan flag dan atribut fitur Anda akan dihapus. JSON yang disederhanakan berisi peta kunci yang cocok dengan masing-masing tombol bendera yang Anda tentukan. JSON yang disederhanakan juga berisi nilai yang dipetakan dari `true` atau `false` untuk atribut. `enabled` Jika bendera disetel `enabled` ke`true`, atribut apa pun dari bendera akan hadir juga. Skema JSON berikut menjelaskan format output JSON.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "patternProperties": {
    "^[a-z][a-zA-Z\\d-_]{0,63}$": {
      "$ref": "#/definitions/attributeValuesMap"
    }
  },
  "maxProperties": 100,
  "additionalProperties": false,
  "definitions": {
    "attributeValuesMap": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
```

```
        }
      },
      "required": ["enabled"],
      "patternProperties": {
        "^[a-z][a-zA-Z\\d-_]{0,63}$": {
          "$ref": "#/definitions/attributeValue"
        }
      },
      "maxProperties": 25,
      "additionalProperties": false
    },
    "attributeValue": {
      "oneOf": [
        { "type": "string","maxLength": 1024 },
        { "type": "number" },
        { "type": "boolean" },
        {
          "type": "array",
          "oneOf": [
            {
              "items": {
                "oneOf": [
                  {
                    "type": "string",
                    "maxLength": 1024
                  }
                ]
              }
            },
            {
              "items": {
                "oneOf": [
                  {
                    "type": "number"
                  }
                ]
              }
            }
          ]
        }
      ],
      "additionalProperties": false
    }
  }
```

```
}
```

Membuat profil konfigurasi formulir gratis di AWS AppConfig

Profil konfigurasi mencakup, antara lain, URI yang memungkinkan AWS AppConfig untuk menemukan data konfigurasi Anda di lokasi yang disimpan dan jenis profil. AWS AppConfig mendukung dua jenis profil konfigurasi: bendera fitur dan konfigurasi bentuk bebas. Profil konfigurasi bendera fitur menyimpan data mereka di toko konfigurasi yang AWS AppConfig dihosting dan URI sederhana`hosted`. Untuk profil konfigurasi bentuk bebas, Anda dapat menyimpan data Anda di penyimpanan konfigurasi yang AWS AppConfig dihosting atau salah satu AWS layanan berikut dan kemampuan Systems Manager:

Lokasi	Tipe file yang didukung
AWS AppConfig toko konfigurasi yang dihosting	YAMM, JSON, dan teks jika ditambahkan menggunakan file. AWS Management Console Tipe file apa pun jika ditambahkan menggunak an tindakan AWS AppConfig CreateHos tedConfigurationVersionAPI.
Amazon Simple Storage Service (Amazon S3)	Setiap
AWS CodePipeline	Pipeline (seperti yang didefinisikan oleh layanan)
AWS Secrets Manager	Rahasia (sebagaimana didefinisikan oleh layanan)
AWS Systems Manager Toko Parameter	Parameter string standar dan aman (seperti yang didefinisikan oleh Parameter Store)
AWS Systems Manager toko dokumen (dokumen SSM)	YAMG, JSON, teks

Profil konfigurasi juga dapat menyertakan validator opsional untuk memastikan data konfigurasi Anda benar secara sintaksis dan semantik. AWS AppConfig melakukan pemeriksaan menggunakan validator saat Anda memulai penerapan. Jika ada kesalahan yang terdeteksi, penerapan berhenti sebelum membuat perubahan apa pun pada target konfigurasi.

> **Note**
>
> Jika memungkinkan, kami sarankan untuk menghosting data konfigurasi Anda di toko konfigurasi yang AWS AppConfig dihosting karena menawarkan sebagian besar fitur dan penyempurnaan.

Untuk konfigurasi bentuk bebas yang disimpan di penyimpanan konfigurasi yang AWS AppConfig dihosting atau dokumen SSM, Anda dapat membuat konfigurasi bentuk bebas dengan menggunakan konsol Systems Manager pada saat Anda membuat profil konfigurasi. Prosesnya dijelaskan nanti dalam topik ini.

Untuk konfigurasi bentuk bebas yang disimpan di Parameter Store, Secrets Manager, atau Amazon S3, Anda harus membuat parameter, rahasia, atau objek terlebih dahulu dan menyimpannya di penyimpanan konfigurasi yang relevan. Setelah Anda menyimpan data konfigurasi, gunakan prosedur dalam topik ini untuk membuat profil konfigurasi.

Topik

Tentang kuota dan batasan penyimpanan konfigurasi

Toko konfigurasi yang didukung oleh AWS AppConfig memiliki kuota dan batasan berikut.

	AWS AppConfig toko konfigurasi yang dihosting	Amazon S3	Penyimpanan Parameter Systems Manager	AWS Secrets Manager	Toko dokumen Systems Manager	AWS CodePipeline
Batas ukuran konfigurasi	2 MB default,	2 MB	4 KB (tingkat gratis)/8	64 KB	64 KB	2 MB

	AWS AppConfig toko konfigurasi yang dihosting	Amazon S3	Penyimpanan Parameter Systems Manager	AWS Secrets Manager	Toko dokumen Systems Manager	AWS CodePipeline
	maksimum 4 MB	Ditegakkan oleh AWS AppConfig, bukan S3	KB (parameter lanjutan)			Dipaksakan oleh AWS AppConfig, tidak CodePipeline
Batas penyimpanan sumber daya	1 GB	Tidak terbatas.	10.000 parameter (tingkat gratis) /100.0 parameter (parameter lanjutan)	500.000	500 dokumen	Dibatasi oleh jumlah profil konfigurasi per aplikasi (100 profil per aplikasi)
Enkripsi sisi server	Ya	SSE-S3, SSE-KM	Ya	Ya	Tidak	Ya
AWS CloudFormation dukungan	Ya	Bukan untuk membuat atau memperbarui data	Ya	Ya	Tidak	Ya
Penetapan Harga	Gratis	Lihat harga Amazon S3	Lihat AWS Systems Manager harga	Lihat AWS Secrets Manager harga	Gratis	Lihat AWS CodePipeline harga

Tentang toko konfigurasi yang AWS AppConfig dihosting

AWS AppConfig termasuk penyimpanan konfigurasi internal atau yang dihosting. Konfigurasi harus 2 MB atau lebih kecil. Toko konfigurasi yang AWS AppConfig dihosting memberikan manfaat berikut dibandingkan opsi penyimpanan konfigurasi lainnya.

- Anda tidak perlu menyiapkan dan mengonfigurasi layanan lain seperti Amazon Simple Storage Service (Amazon S3) atau Parameter Store.
- Anda tidak perlu mengonfigurasi izin AWS Identity and Access Management (IAM) untuk menggunakan penyimpanan konfigurasi.
- Anda dapat menyimpan konfigurasi di YAMAL, JSON, atau sebagai dokumen teks.
- Tidak ada biaya untuk menggunakan penyimpanan.
- Anda dapat membuat konfigurasi dan menambahkannya ke penyimpanan saat membuat profil konfigurasi.

Tentang konfigurasi yang disimpan di Amazon S3

Anda dapat menyimpan konfigurasi di bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3). Saat membuat profil konfigurasi, Anda menentukan URI ke objek S3 tunggal dalam bucket. Anda juga menentukan Amazon Resource Name (ARN) dari peran AWS Identity and Access Management (IAM) yang memberikan AWS AppConfig izin untuk mendapatkan objek. Sebelum Anda membuat profil konfigurasi untuk objek Amazon S3, perhatikan batasan berikut.

Pembatasan	Detail
Size	Konfigurasi yang disimpan sebagai objek S3 dapat berukuran maksimal 1 MB.
Enkripsi objek	Profil konfigurasi dapat menargetkan objek terenkripsi SSE-S3 dan SSE-KMS.
Kelas penyimpanan	AWS AppConfig mendukung kelas penyimpan an S3 berikut:`STANDARD`,,`INTELLIGE NT_TIERING` , `REDUCED_REDUNDANCY` `STANDARD_IA` , dan`ONEZONE_IA` . Kelas-kelas berikut tidak didukung: Semua kelas S3 Glacier (`GLACIER`dan). `DEEP_ARCHIVE`

Pembatasan	Detail
Penentuan Versi	AWS AppConfig mengharuskan objek S3 menggunakan versi.

Mengonfigurasi izin untuk konfigurasi yang disimpan sebagai objek Amazon S3

Saat Anda membuat profil konfigurasi untuk konfigurasi yang disimpan sebagai objek S3, Anda harus menentukan ARN untuk peran IAM yang AWS AppConfig memberikan izin untuk mendapatkan objek. Peran harus menyertakan izin berikut.

Izin untuk mengakses objek S3

- s3: GetObject
- s3: Versi GetObject

Izin untuk mencantumkan bucket S3

s3: ListAll MyBuckets

Izin untuk mengakses bucket S3 tempat objek disimpan

- s3: Lokasi GetBucket
- s3: Versi GetBucket
- s3: ListBucket
- s3: Versi ListBucket

Selesaikan prosedur berikut untuk membuat peran yang memungkinkan AWS AppConfig untuk mendapatkan konfigurasi yang disimpan dalam objek S3.

Membuat Kebijakan IAM untuk Mengakses Objek S3

Gunakan prosedur berikut untuk membuat kebijakan IAM yang memungkinkan AWS AppConfig untuk mendapatkan konfigurasi yang disimpan dalam objek S3.

Untuk membuat kebijakan IAM untuk mengakses objek S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Kebijakan dan kemudian pilih Buat kebijakan.
3. Di halaman Buat kebijakan, pilih tab JSON.
4. Perbarui kebijakan sampel berikut dengan informasi tentang bucket S3 dan objek konfigurasi Anda. Kemudian tempelkan kebijakan ke bidang teks pada tab JSON. Ganti *nilai placeholder* dengan informasi Anda sendiri.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/my-configurations/my-
configuration.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

5. Pilih Tinjau kebijakan.
6. Pada halaman Kebijakan tinjauan, ketikkan nama di kotak Nama, lalu ketik deskripsi.
7. Pilih Buat kebijakan. Sistem mengembalikan Anda ke halaman Peran.

Membuat Peran IAM untuk Mengakses Objek S3

Gunakan prosedur berikut untuk membuat peran IAM yang memungkinkan AWS AppConfig untuk mendapatkan konfigurasi yang disimpan dalam objek S3.

Untuk membuat peran IAM untuk mengakses objek Amazon S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Pada bagian Pilih jenis entitas tepercaya, pilih AWS layanan.
4. Di bagian Pilih kasus penggunaan, di bawah Kasus penggunaan umum, pilih EC2, lalu pilih Berikutnya: Izin.
5. Pada halaman Lampirkan kebijakan izin, di kotak pencarian, masukkan nama kebijakan yang Anda buat di prosedur sebelumnya.
6. Pilih kebijakan dan kemudian pilih Berikutnya: Tag.
7. Pada halaman Tambahkan tag (opsional), masukkan kunci dan nilai opsional, lalu pilih Berikutnya: Tinjau.
8. Pada halaman Tinjauan, ketikkan nama di bidang Nama peran, lalu ketik deskripsi.
9. Pilih Buat peran. Sistem mengembalikan Anda ke halaman Peran.
10. Pada halaman Peran, pilih peran yang baru Anda buat untuk membuka halaman Ringkasan. Catat Nama peran dan Peran ARN. Anda akan menentukan peran ARN saat Anda membuat profil konfigurasi nanti dalam topik ini.

Menciptakan Hubungan Kepercayaan

Gunakan prosedur berikut untuk mengonfigurasi peran yang baru saja Anda buat agar dipercaya AWS AppConfig.

Untuk menambah hubungan kepercayaan

1. Di halaman Ringkasan untuk peran yang baru saja Anda buat, pilih tab Trust Relationships, lalu pilih Edit Trust Relationship.
2. Hapus `"ec2.amazonaws.com"` dan tambahkan`"appconfig.amazonaws.com"`, seperti yang ditunjukkan pada contoh berikut.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Pilih Perbarui Kebijakan Kepercayaan.

Membuat konfigurasi bentuk bebas dan profil konfigurasi

Bagian ini menjelaskan cara membuat konfigurasi bentuk bebas dan profil konfigurasi. Sebelum Anda mulai, perhatikan informasi berikut.

- Prosedur berikut mengharuskan Anda untuk menentukan peran layanan IAM sehingga AWS AppConfig dapat mengakses data konfigurasi Anda di penyimpanan konfigurasi yang Anda pilih. Peran ini tidak diperlukan jika Anda menggunakan penyimpanan konfigurasi yang AWS AppConfig dihosting. Jika Anda memilih S3, Parameter Store, atau penyimpanan dokumen Systems Manager, maka Anda harus memilih peran IAM yang ada atau memilih opsi agar sistem secara otomatis membuat peran untuk Anda. Untuk informasi lebih lanjut, tentang peran ini, lihatTentang peran IAM profil konfigurasi.
- Prosedur berikut juga memberi Anda opsi untuk mengaitkan ekstensi dengan profil konfigurasi bendera fitur. Ekstensi menambah kemampuan Anda untuk menyuntikkan logika atau perilaku pada titik yang berbeda selama AWS AppConfig alur kerja membuat atau menerapkan konfigurasi. Untuk informasi selengkapnya, lihat Tentang AWS AppConfig ekstensi.
- Jika Anda ingin membuat profil konfigurasi untuk konfigurasi yang disimpan di S3, Anda harus mengonfigurasi izin. Untuk informasi selengkapnya tentang izin dan persyaratan lain untuk menggunakan S3 sebagai penyimpanan konfigurasi, lihat. Tentang konfigurasi yang disimpan di Amazon S3
- Jika Anda ingin menggunakan validator, tinjau detail dan persyaratan untuk menggunakannya. Untuk informasi selengkapnya, lihat Tentang validator.

Topik

- Membuat profil konfigurasi AWS AppConfig bentuk bebas (konsol)
- Membuat profil konfigurasi AWS AppConfig bentuk bebas (baris perintah)

Membuat profil konfigurasi AWS AppConfig bentuk bebas (konsol)

Gunakan prosedur berikut untuk membuat profil konfigurasi AWS AppConfig bentuk bebas dan (opsional) konfigurasi bentuk bebas dengan menggunakan konsol. AWS Systems Manager

Untuk membuat profil konfigurasi bentuk bebas

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi yang Anda buat. Membuat namespace untuk aplikasi Anda di AWS AppConfig
3. Pilih tab Configuration profiles and feature flags, lalu pilih Create configuration.
4. Di bagian Opsi konfigurasi, pilih konfigurasi Freeform.
5. Untuk nama profil Konfigurasi, masukkan nama untuk profil konfigurasi.
6. (Opsional) Perluas Deskripsi dan masukkan deskripsi.
7. (Opsional) Perluas Opsi tambahan dan selesaikan yang berikut ini, seperlunya.
   a. Di bagian Ekstensi asosiasi, pilih ekstensi dari daftar.
   b. Di bagian Tag, pilih Tambahkan tag baru, lalu tentukan kunci dan nilai opsional.
8. Pilih Selanjutnya.
9. Pada halaman Tentukan data konfigurasi, di bagian Pembelaan konfigurasi, pilih opsi.
10. Lengkapi bidang untuk opsi yang Anda pilih, seperti yang dijelaskan dalam tabel berikut.

Opsi dipilih	Detail
AWS AppConfig konfigurasi yang dihosting	Pilih salah satu Teks, JSON, atau YAMG, dan masukkan konfigurasi Anda di bidang. Pergi ke Langkah 12 dalam prosedur ini.
Objek Amazon S3	Masukkan URI objek di bidang sumber objek S3 dan pergi ke Langkah 11 dalam prosedur ini.

Opsi dipilih	Detail
AWS CodePipeline	Pilih Berikutnya dan pergi ke Langkah 12 dalam prosedur ini.
Rahasia Secrets Manager	Pilih rahasia dari daftar, lanjutkan ke Langkah 11 dalam prosedur ini.
AWS Systems Manager parameter	Pilih parameter dari daftar dan pergi ke Langkah 11 dalam prosedur ini.
AWS Systems Manager dokumen	1. Pilih dokumen dari daftar atau pilih Buat dokumen baru. 2. Jika Anda memilih Buat dokumen baru, untuk nama Dokumen, masukkan nama. Secara opsional, perluas Nama versi dan masukkan nama untuk versi dokumen. 3. Untuk skema konfigurasi Aplikasi, pilih skema JSON dari daftar atau pilih Buat skema. Jika Anda memilih Create schema, Systems Manager akan membuka halaman Create schema. Masukkan detail skema, lalu pilih Buat skema konfigurasi aplikasi. 4. Di bagian Konten, pilih YAMAL atau JSON dan kemudian masukkan data konfigurasi di bidang.

11. Di bagian Peran layanan, pilih Peran layanan baru untuk AWS AppConfig membuat peran IAM yang menyediakan akses ke data konfigurasi. AWS AppConfig secara otomatis mengisi bidang Nama peran berdasarkan nama yang Anda masukkan sebelumnya. Atau, pilih Peran layanan yang ada. Pilih peran dengan menggunakan daftar ARN Peran.
12. Secara opsional, pada halaman Tambahkan validator, pilih Skema JSON atau. AWS Lambda Jika Anda memilih Skema JSON, masukkan Skema JSON di bidang. Jika Anda memilih AWS Lambda, pilih fungsi Amazon Resource Name (ARN) dan versi dari daftar.

> **Important**
>
> Data konfigurasi yang disimpan dalam dokumen SSM harus memvalidasi terhadap Skema JSON terkait sebelum Anda dapat menambahkan konfigurasi ke sistem. Parameter SSM tidak memerlukan metode validasi, tetapi kami menyarankan Anda membuat pemeriksaan validasi untuk konfigurasi parameter SSM baru atau yang diperbarui dengan menggunakan. AWS Lambda

13. Pilih Selanjutnya.
14. Pada halaman Tinjau dan simpan, pilih Simpan dan lanjutkan untuk menerapkan.

> **Important**
>
> Jika Anda membuat profil konfigurasi untuk AWS CodePipeline, maka Anda harus membuat pipeline CodePipeline yang menentukan AWS AppConfig sebagai penyedia penerapan. Anda tidak perlu tampilMenyebarkan flag fitur dan data konfigurasi di AWS AppConfig. Namun, Anda harus mengonfigurasi klien untuk menerima pembaruan konfigurasi aplikasi seperti yang dijelaskan dalamMengambil konfigurasi dengan langsung memanggil API. Untuk informasi tentang membuat pipeline yang ditetapkan AWS AppConfig sebagai penyedia penerapan, lihat Tutorial: Membuat Pipeline yang Digunakan AWS AppConfig sebagai Penyedia Deployment di Panduan Pengguna.AWS CodePipeline

Lanjut ke Menyebarkan flag fitur dan data konfigurasi di AWS AppConfig.

Membuat profil konfigurasi AWS AppConfig bentuk bebas (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS CLI (di Linux atau Windows) atau AWS Tools for PowerShell untuk membuat profil konfigurasi AWS AppConfig bentuk bebas. Jika mau, Anda dapat menggunakan AWS CloudShell untuk menjalankan perintah yang tercantum di bawah ini. Untuk informasi selengkapnya, lihat Apa itu AWS CloudShell? dalam AWS CloudShell Panduan Pengguna.

> **Note**
>
> Untuk konfigurasi bentuk bebas yang dihosting di penyimpanan konfigurasi yang AWS AppConfig dihosting, Anda menentukan `hosted` URI lokasi.

Untuk membuat profil konfigurasi dengan menggunakan AWS CLI

1. Buka AWS CLI.
2. Jalankan perintah berikut untuk membuat profil konfigurasi bentuk bebas.

   Linux

   ```
   aws appconfig create-configuration-profile \
     --application-id The_application_ID \
     --name A_name_for_the_configuration_profile \
     --description A_description_of_the_configuration_profile \
     --location-uri A_URI_to_locate_the_configuration or hosted \
     --retrieval-role-
   arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified
    \
     --tags User_defined_key_value_pair_metadata_of_the_configuration_profile \
     --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
    Lambda_function,Type=JSON_SCHEMA or LAMBDA"
   ```

   Windows

   ```
   aws appconfig create-configuration-profile ^
     --application-id The_application_ID ^
     --name A_name_for_the_configuration_profile ^
     --description A_description_of_the_configuration_profile ^
     --location-uri A_URI_to_locate_the_configuration or hosted  ^
     --retrieval-role-
   arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified
    ^
     --tags User_defined_key_value_pair_metadata_of_the_configuration_profile ^
     --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
    Lambda_function,Type=JSON_SCHEMA or LAMBDA"
   ```

PowerShell

```
New-APPCConfigurationProfile `
  -Name A_name_for_the_configuration_profile `
  -ApplicationId The_application_ID `
  -Description Description_of_the_configuration_profile `
  -LocationUri A_URI_to_locate_the_configuration or hosted `
  -
RetrievalRoleArn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_
 `
  -
Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_configuration_profile
 `
  -Validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Important

Perhatikan informasi penting berikut.

- Jika Anda membuat profil konfigurasi untuk AWS CodePipeline, maka Anda harus membuat pipeline CodePipeline yang menentukan AWS AppConfig sebagai penyedia penerapan. Anda tidak perlu tampilMenyebarkan flag fitur dan data konfigurasi di AWS AppConfig. Namun, Anda harus mengonfigurasi klien untuk menerima pembaruan konfigurasi aplikasi seperti yang dijelaskan dalamMengambil konfigurasi dengan langsung memanggil API. Untuk informasi tentang membuat pipeline yang ditetapkan AWS AppConfig sebagai penyedia penerapan, lihat Tutorial: Membuat Pipeline yang Digunakan AWS AppConfig sebagai Penyedia Deployment di Panduan Pengguna.AWS CodePipeline
- Jika Anda membuat konfigurasi di penyimpanan konfigurasi yang AWS AppConfig dihosting, Anda dapat membuat versi konfigurasi baru menggunakan operasi CreateHostedConfigurationVersionAPI. Untuk melihat AWS CLI detail dan contoh perintah untuk operasi API ini, lihat create-hosted-configuration-version di Command Reference.AWS CLI

Lanjut ke Menyebarkan flag fitur dan data konfigurasi di AWS AppConfig.

Sumber data konfigurasi lainnya

Topik ini mencakup informasi tentang AWS layanan lain yang terintegrasi dengan AWS AppConfig.

AWS AppConfig Integrasi dengan AWS Secrets Manager

Secrets Manager membantu Anda mengenkripsi, menyimpan, dan mengambil kredensi dengan aman untuk database dan layanan lainnya. Alih-alih membuat hardcoding kredenal di aplikasi Anda, Anda dapat melakukan panggilan ke Secrets Manager untuk mengambil kredensil Anda kapan pun diperlukan. Secrets Manager membantu Anda melindungi akses ke sumber daya dan data TI Anda dengan memungkinkan Anda untuk memutar dan mengelola akses ke rahasia Anda.

Saat Anda membuat profil konfigurasi bentuk bebas, Anda dapat memilih Secrets Manager sebagai sumber data konfigurasi Anda. Anda harus bergabung dengan Secrets Manager dan membuat rahasia sebelum membuat profil konfigurasi. Untuk informasi selengkapnya tentang Secrets Manager, lihat Apa itu AWS Secrets Manager? dalam AWS Secrets Manager User Guide. Untuk informasi tentang membuat profil konfigurasi yang menggunakan Secrets Manager, lihatMembuat flag fitur dan data konfigurasi formulir gratis di AWS AppConfig.

Menyebarkan flag fitur dan data konfigurasi di AWS AppConfig

Setelah Anda membuat artefak yang diperlukan untuk bekerja dengan flag fitur dan data konfigurasi bentuk bebas, Anda dapat membuat penerapan baru. Saat Anda membuat penyebaran baru, Anda menentukan informasi berikut:

- ID aplikasi
- ID profil konfigurasi
- Versi konfigurasi
- ID lingkungan tempat Anda ingin menyebarkan data konfigurasi
- ID strategi penerapan yang menentukan seberapa cepat Anda ingin perubahan diterapkan
- ID kunci AWS Key Management Service (AWS KMS) untuk mengenkripsi data menggunakan kunci yang dikelola pelanggan.

Saat Anda memanggil tindakan StartDeploymentAPI, AWS AppConfig lakukan tugas-tugas berikut:

1. Mengambil data konfigurasi dari penyimpanan data yang mendasarinya dengan menggunakan URI lokasi di profil konfigurasi.
2. Memverifikasi data konfigurasi secara sintaksis dan semantik benar dengan menggunakan validator yang Anda tentukan saat Anda membuat profil konfigurasi.
3. Cache salinan data sehingga siap untuk diambil oleh aplikasi Anda. Salinan cache ini disebut data yang digunakan.

AWS AppConfig terintegrasi dengan Amazon CloudWatch untuk memantau penerapan. Jika penerapan memicu alarm masuk CloudWatch, secara AWS AppConfig otomatis memutar kembali penerapan untuk meminimalkan dampak pada pengguna aplikasi Anda.

Topik

- Bekerja dengan strategi penyebaran
- Men-deploy konfigurasi
- AWS AppConfig integrasi penerapan dengan CodePipeline

Bekerja dengan strategi penyebaran

Strategi penerapan memungkinkan Anda untuk secara perlahan melepaskan perubahan pada lingkungan produksi selama beberapa menit atau jam. Strategi AWS AppConfig penyebaran mendefinisikan aspek-aspek penting berikut dari penerapan konfigurasi.

<table>
<tr><th>Pengaturan</th><th>Deskripsi</th></tr>
<tr><td>Jenis deployment</td><td>Jenis penerapan mendefinisikan bagaimana konfigurasi diterapkan atau diluncurkan. AWS AppConfig mendukung jenis penyebaran Linear dan Eksponensial.
Linear: Untuk jenis ini, AWS AppConfig memproses penerapan dengan penambahan faktor pertumbuhan yang didistribusikan secara merata selama penerapan. Berikut adalah contoh timeline untuk penerapan 10 jam yang menggunakan pertumbuhan linier 20%:<table><tr><th>Waktu berlalu</th><th>Kemajuan penerapan</th></tr><tr><td>0 jam</td><td>0%</td></tr><tr><td>2 jam</td><td>20%</td></tr><tr><td>4 jam</td><td>40%</td></tr><tr><td>6 jam</td><td>60%</td></tr><tr><td>8 jam</td><td>80%</td></tr><tr><td>10 jam</td><td>100%</td></tr></table>
Eksponensial: Untuk tipe ini, AWS AppConfig memproses deployment secara eksponensial menggunakan rumus berikut: <code>G*(2^N)</code>.</td></tr>
</table>

Pengaturan	Deskripsi

	Dalam rumus ini, `G` adalah persentase langkah yang ditentukan oleh pengguna dan `N` merupakan jumlah langkah hingga konfigurasi diterapkan ke semua target. Misalnya, jika Anda menentukan faktor pertumbuhan 2, maka sistem menggulung konfigurasi sebagai berikut:
	`2*(2^0)`
	`2*(2^1)`
	`2*(2^2)`
	Dinyatakan secara numerik, deployment diluncurkan sebagai berikut: 2% dari target, 4% dari target, 8% dari target, dan berlanjut sampai konfigurasi telah di-deploy ke semua target.

Persentase langkah (faktor pertumbuhan)	Pengaturan ini menentukan persentase penelepon yang akan ditargetkan selama setiap langkah penerapan. **Note** Dalam SDK dan Referensi AWS AppConfig API, `step percentage` disebut`growth factor`.
Waktu penyebaran	Pengaturan ini menentukan jumlah waktu selama AWS AppConfig penyebaran ke host. Ini bukan nilai batas waktu. Ini adalah jendela waktu di mana penyebaran diproses dalam interval.

Pengaturan	Deskripsi
Waktu panggang	Pengaturan ini menentukan jumlah AWS AppConfig monitor waktu untuk CloudWatch alarm Amazon setelah konfigurasi dikerahkan ke 100% targetnya, sebelum mempertimbangkan penerapan selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan. Anda harus mengonfigurasi izin AWS AppConfig untuk memutar kembali berdasarkan CloudWatch alarm. Untuk informasi selengkapnya, lihat (Opsional) Konfigurasikan izin untuk rollback berdasarkan alarm CloudWatch .

Anda dapat memilih strategi yang telah ditentukan yang disertakan dengan AWS AppConfig atau membuat strategi Anda sendiri.

Topik

Strategi penyebaran yang telah ditentukan

AWS AppConfig menyertakan strategi penerapan yang telah ditentukan untuk membantu Anda menerapkan konfigurasi dengan cepat. Alih-alih membuat strategi Anda sendiri, Anda dapat memilih salah satu dari berikut ini saat Anda menerapkan konfigurasi.

Strategi penyebaran	Deskripsi
AppConfig.Linear20 6 PercentEvery Menit	AWS direkomendasikan: Strategi ini menerapkan konfigurasi hingga 20% dari semua target setiap enam menit selama 30 menit penerapan. Sistem memonitor CloudWatch alarm Amazon selama 30 menit.

Strategi penyebaran	Deskripsi
	Jika tidak ada alarm yang diterima saat ini, penyebaran selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan. Kami merekomendasikan penggunaan strategi ini untuk penerapan produksi karena selaras dengan praktik AWS terbaik dan mencakup penekanan tambahan pada keselamatan penerapan karena durasinya yang lama dan waktu pemanggangan.
AppConfig.Canary10Percent20Menit	AWS direkomendasikan: Strategi ini memproses penyebaran secara eksponensial menggunakan faktor pertumbuh an 10% selama 20 menit. Sistem memonitor CloudWatch alarm selama 10 menit. Jika tidak ada alarm yang diterima saat ini, penyebara n selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan. Sebaiknya gunakan strategi ini untuk penerapan produksi karena strategi ini sejalan dengan praktik AWS terbaik untuk penerapan konfigurasi.
AppConfig.AllAtOnce	Cepat: Strategi ini segera menyebarkan konfigurasi ke semua target. Sistem memonitor CloudWatc h alarm selama 10 menit. Jika tidak ada alarm yang diterima saat ini, penyebaran selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan.

Strategi penyebaran	Deskripsi
AppConfig.Linear50 30 Detik PercentEvery	Pengujian/demonstrasi: Strategi ini menyebarkan konfigurasi ke setengah dari semua target setiap 30 detik untuk penerapan satu menit. Sistem memonitor CloudWatch alarm Amazon selama 1 menit. Jika tidak ada alarm yang diterima saat ini, penyebaran selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan. Sebaiknya gunakan strategi ini hanya untuk tujuan pengujian atau demonstrasi karena memiliki durasi dan waktu pemanggangan yang singkat.

Buat strategi deployment

Jika Anda tidak ingin menggunakan salah satu strategi penerapan yang telah ditentukan sebelumnya, Anda dapat membuatnya sendiri. Anda dapat membuat maksimal 20 strategi penyebaran. Saat menerapkan konfigurasi, Anda dapat memilih strategi penerapan yang paling sesuai untuk aplikasi dan lingkungan.

Membuat strategi AWS AppConfig penerapan (konsol)

Gunakan prosedur berikut untuk membuat strategi AWS AppConfig penyebaran dengan menggunakan AWS Systems Manager konsol.

Untuk membuat strategi penyebaran

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Strategi penyebaran, lalu pilih Buat strategi penerapan.
3. Untuk Nama, masukkan nama untuk strategi penyebaran.
4. Untuk Deskripsi, masukkan informasi tentang strategi penyebaran.

5. Untuk jenis Deployment, pilih tipe.
6. Untuk persentase Langkah, pilih persentase penelepon yang akan ditargetkan selama setiap langkah penerapan.
7. Untuk waktu Deployment, masukkan total durasi penerapan dalam hitungan menit atau jam.
8. Untuk waktu Bake, masukkan total waktu, dalam menit atau jam, untuk memantau CloudWatch alarm Amazon sebelum melanjutkan ke langkah penerapan berikutnya atau sebelum mempertimbangkan penerapan selesai.
9. Di bagian Tag, masukkan kunci dan nilai opsional. Anda dapat menentukan maksimum 50 tag untuk sumber daya.
10. Pilih Buat strategi penerapan.

> **Important**
>
> Jika Anda membuat profil konfigurasi untuk AWS CodePipeline, maka Anda harus membuat pipeline CodePipeline yang menentukan AWS AppConfig sebagai penyedia penerapan. Anda tidak perlu tampilMen-deploy konfigurasi. Namun, Anda harus mengonfigurasi klien untuk menerima pembaruan konfigurasi aplikasi seperti yang dijelaskan dalamMengambil konfigurasi dengan langsung memanggil API. Untuk informasi tentang membuat pipeline yang ditetapkan AWS AppConfig sebagai penyedia penerapan, lihat Tutorial: Membuat Pipeline yang Digunakan AWS AppConfig sebagai Penyedia Deployment di Panduan Pengguna.AWS CodePipeline

Lanjut ke Men-deploy konfigurasi.

Membuat strategi AWS AppConfig penyebaran (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS CLI (di Linux atau Windows) atau AWS Tools for PowerShell untuk membuat strategi AWS AppConfig penyebaran.

Untuk membuat strategi penyebaran langkah demi langkah

1. Buka AWS CLI.
2. Jalankan perintah berikut untuk membuat strategi penyebaran.

Linux

```
aws appconfig create-deployment-strategy \
  --name A_name_for_the_deployment_strategy \
  --description A_description_of_the_deployment_strategy \
  --deployment-duration-in-minutes Total_amount_of_time_for_a_deployment_to_last
 \
  --final-bake-time-in-minutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 \
  --growth-
factor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_interva
 \
  --growth-
type The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_time
 \
  --replicate-
to To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document \
  --tags User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

Windows

```
aws appconfig create-deployment-strategy ^
  --name A_name_for_the_deployment_strategy ^
  --description A_description_of_the_deployment_strategy ^
  --deployment-duration-in-minutes Total_amount_of_time_for_a_deployment_to_last
 ^
  --final-bake-time-in-minutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 ^
  --growth-
factor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_interva
 ^
  --growth-
type The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_time
 ^
  --name A_name_for_the_deployment_strategy ^
  --replicate-
to To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document ^
  --tags User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

PowerShell

```
New-APPCDeploymentStrategy `
  --Name A_name_for_the_deployment_strategy `
  --Description A_description_of_the_deployment_strategy `
  --DeploymentDurationInMinutes Total_amount_of_time_for_a_deployment_to_last `
  --FinalBakeTimeInMinutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 `
  --
GrowthFactor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_i
 `
  --
GrowthType The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_
 `
  --
ReplicateTo To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document
 `
  --
Tag Hashtable_type_User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "Id": "Id of the deployment strategy",
   "Name": "Name of the deployment strategy",
   "Description": "Description of the deployment strategy",
   "DeploymentDurationInMinutes": "Total amount of time the deployment lasted",
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
   "GrowthFactor": "The percentage of targets that received a deployed
 configuration during each interval",
   "FinalBakeTimeInMinutes": "The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete",
   "ReplicateTo": "The Systems Manager (SSM) document where the deployment
 strategy is saved"
}
```

Windows

```
{
   "Id": "Id of the deployment strategy",
   "Name": "Name of the deployment strategy",
   "Description": "Description of the deployment strategy",
   "DeploymentDurationInMinutes": "Total amount of time the deployment lasted",
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
   "GrowthFactor": "The percentage of targets that received a deployed
 configuration during each interval",
   "FinalBakeTimeInMinutes": "The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete",
   "ReplicateTo": "The Systems Manager (SSM) document where the deployment
 strategy is saved"
}
```

PowerShell

```
ContentLength               : Runtime of the command
DeploymentDurationInMinutes : Total amount of time the deployment lasted
Description                 : Description of the deployment strategy
FinalBakeTimeInMinutes      : The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete
GrowthFactor                : The percentage of targets that received a deployed
 configuration during each interval
GrowthType                  : The linear or exponential algorithm used to define
 how percentage grew over time
HttpStatusCode              : HTTP Status of the runtime
Id                          : The deployment strategy ID
Name                        : Name of the deployment strategy
ReplicateTo                 : The Systems Manager (SSM) document where the
 deployment strategy is saved
ResponseMetadata            : Runtime Metadata
```

Men-deploy konfigurasi

Setelah Anda membuat artefak yang diperlukan untuk bekerja dengan flag fitur dan data konfigurasi bentuk bebas, Anda dapat membuat penerapan baru dengan menggunakan AWS Management Console, the AWS CLI, atau SDK. Memulai penerapan dalam AWS AppConfig memanggil operasi

StartDeploymentAPI. Panggilan ini mencakup ID dari aplikasi, lingkungan, profil konfigurasi, dan (secara opsional) versi data konfigurasi AWS AppConfig yang di-deploy. Panggilan juga mencakup ID dari strategi deployment yang digunakan, yang menentukan cara data konfigurasi di-deploy.

Jika Anda menerapkan rahasia yang disimpan dalam AWS Secrets Manager, objek Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) yang dienkripsi dengan kunci yang dikelola pelanggan, atau parameter string aman yang disimpan di AWS Systems Manager Parameter Store yang dienkripsi dengan kunci yang dikelola pelanggan, Anda harus menentukan nilai untuk parameter tersebut. `KmsKeyIdentifier` Jika konfigurasi Anda tidak dienkripsi atau dienkripsi dengan Kunci yang dikelola AWS, menentukan nilai untuk parameter tidak diperlukan. `KmsKeyIdentifier`

> **Note**
>
> Nilai yang Anda tentukan `KmsKeyIdentifier` harus berupa kunci yang dikelola pelanggan. Ini tidak harus menjadi kunci yang sama yang Anda gunakan untuk mengenkripsi konfigurasi Anda.
> Saat Anda memulai penerapan dengan a`KmsKeyIdentifier`, kebijakan izin yang dilampirkan pada prinsipal AWS Identity and Access Management (IAM) Anda harus mengizinkan operasi. `kms:GenerateDataKey`

AWS AppConfig memantau distribusi ke semua host dan melaporkan status. Jika distribusi gagal, maka AWS AppConfig putar kembali konfigurasi.

> **Note**
>
> Anda hanya dapat menerapkan satu konfigurasi pada satu waktu ke lingkungan. Namun, Anda dapat menerapkan satu konfigurasi masing-masing ke lingkungan yang berbeda secara bersamaan.

Menyebarkan konfigurasi (konsol)

Gunakan prosedur berikut untuk menerapkan AWS AppConfig konfigurasi menggunakan AWS Systems Manager konsol.

Untuk menerapkan konfigurasi dengan menggunakan konsol

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi yang Anda buat. Membuat namespace untuk aplikasi Anda di AWS AppConfig
3. Pada tab Lingkungan, isi tombol radio untuk lingkungan, lalu pilih Lihat detail.
4. Pilih Mulai penerapan.
5. Untuk Konfigurasi, pilih konfigurasi dari daftar.
6. Bergantung pada sumber konfigurasi Anda, gunakan daftar versi untuk memilih versi yang ingin Anda gunakan.
7. Untuk strategi Deployment, pilih strategi dari daftar.
8. (Opsional) Untuk deskripsi Deployment, masukkan deskripsi.
9. Untuk opsi enkripsi tambahan, pilih AWS Key Management Service kunci dari daftar.
10. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai opsional. Anda dapat menentukan maksimum 50 tag untuk sumber daya.
11. Pilih Mulai penerapan.

Menyebarkan konfigurasi (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS CLI (di Linux atau Windows) atau AWS Tools for PowerShell untuk menyebarkan AWS AppConfig konfigurasi.

Untuk menerapkan konfigurasi langkah demi langkah

1. Buka AWS CLI.
2. Jalankan perintah berikut untuk menyebarkan konfigurasi.

   Linux

```
aws appconfig start-deployment \
  --application-id The_application_ID \
  --environment-id The_environment_ID \
  --deployment-strategy-id The_deployment_strategy_ID \
  --configuration-profile-id The_configuration_profile_ID \
  --configuration-version The_configuration_version_to_deploy \
  --description A_description_of_the_deployment \
```

```
  --tags User_defined_key_value_pair_metadata_of_the_deployment
```

Windows

```
aws appconfig start-deployment ^
  --application-id The_application_ID ^
  --environment-id The_environment_ID ^
  --deployment-strategy-id The_deployment_strategy_ID ^
  --configuration-profile-id The_configuration_profile_ID ^
  --configuration-version The_configuration_version_to_deploy ^
  --description A_description_of_the_deployment ^
  --tags User_defined_key_value_pair_metadata_of_the_deployment
```

PowerShell

```
Start-APPCDeployment `
  -ApplicationId The_application_ID `
  -ConfigurationProfileId The_configuration_profile_ID `
  -ConfigurationVersion The_configuration_version_to_deploy `
  -DeploymentStrategyId The_deployment_strategy_ID `
  -Description A_description_of_the_deployment `
  -EnvironmentId The_environment_ID `
  -Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_deployment
```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
   "DeploymentNumber": The sequence number of the deployment,
   "ConfigurationName": "The name of the configuration",
   "ConfigurationLocationUri": "Information about the source location of the
 configuration",
   "ConfigurationVersion": "The configuration version that was deployed",
   "Description": "The description of the deployment",
```

```
  "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
  "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
  "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
  "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
  "State": "The state of the deployment",

  "EventLog": [
     {
        "Description": "A description of the deployment event",
        "EventType": "The type of deployment event",
        "OccurredAt": The date and time the event occurred,
        "TriggeredBy": "The entity that triggered the deployment event"
     }
  ],

  "PercentageComplete": The percentage of targets for which the deployment is
 available,
  "StartedAt": The time the deployment started,
  "CompletedAt": The time the deployment completed
}
```

Windows

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
   "DeploymentNumber": The sequence number of the deployment,
   "ConfigurationName": "The name of the configuration",
   "ConfigurationLocationUri": "Information about the source location of the
 configuration",
   "ConfigurationVersion": "The configuration version that was deployed",
   "Description": "The description of the deployment",
   "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
```

```
  "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
  "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
  "State": "The state of the deployment",

  "EventLog": [
     {
        "Description": "A description of the deployment event",
        "EventType": "The type of deployment event",
        "OccurredAt": The date and time the event occurred,
        "TriggeredBy": "The entity that triggered the deployment event"
     }
  ],

  "PercentageComplete": The percentage of targets for which the deployment is
 available,
  "StartedAt": The time the deployment started,
  "CompletedAt": The time the deployment completed
}
```

PowerShell

```
ApplicationId               : The ID of the application that was deployed
CompletedAt                 : The time the deployment completed
ConfigurationLocationUri    : Information about the source location of the
 configuration
ConfigurationName           : The name of the configuration
ConfigurationProfileId      : The ID of the configuration profile that was
 deployed
ConfigurationVersion        : The configuration version that was deployed
ContentLength               : Runtime of the deployment
DeploymentDurationInMinutes : Total amount of time the deployment lasted
DeploymentNumber            : The sequence number of the deployment
DeploymentStrategyId        : The ID of the deployment strategy that was
 deployed
Description                 : The description of the deployment
EnvironmentId               : The ID of the environment that was deployed
EventLog                    : {Description : A description of the deployment
 event, EventType : The type of deployment event, OccurredAt : The date and time
 the event occurred,
        TriggeredBy : The entity that triggered the deployment event}
```

```
FinalBakeTimeInMinutes      : Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete
GrowthFactor                : The percentage of targets to receive a deployed
 configuration during each interval
GrowthType                  : The linear or exponential algorithm used to define
 how percentage grew over time
HttpStatusCode              : HTTP Status of the runtime
PercentageComplete          : The percentage of targets for which the deployment
 is available
ResponseMetadata            : Runtime Metadata
StartedAt                   : The time the deployment started
State                       : The state of the deployment
```

AWS AppConfig integrasi penerapan dengan CodePipeline

AWS AppConfig adalah tindakan penerapan terintegrasi untuk AWS CodePipeline (CodePipeline). CodePipeline adalah layanan pengiriman berkelanjutan yang dikelola sepenuhnya yang membantu Anda mengotomatiskan saluran pipa rilis Anda untuk pembaruan aplikasi dan infrastruktur yang cepat dan andal. CodePipeline mengotomatiskan fase build, test, dan deploy dari proses rilis Anda setiap kali ada perubahan kode, berdasarkan model rilis yang Anda tentukan. Untuk informasi selengkapnya, lihat Apa itu AWS CodePipeline?

Integrasi AWS AppConfig dengan CodePipeline menawarkan manfaat berikut:

- Pelanggan yang menggunakan CodePipeline untuk mengelola orkestrasi sekarang memiliki cara yang ringan untuk menerapkan perubahan konfigurasi ke aplikasi mereka tanpa harus menyebarkan seluruh basis kode mereka.
- Pelanggan yang ingin menggunakan AWS AppConfig untuk mengelola penerapan konfigurasi tetapi terbatas karena AWS AppConfig tidak mendukung kode atau penyimpanan konfigurasi mereka saat ini, sekarang memiliki opsi tambahan. CodePipeline mendukung AWS CodeCommit, GitHub, dan BitBucket (untuk beberapa nama).

> **Note**
>
> AWS AppConfig integrasi dengan CodePipeline hanya didukung di Wilayah AWS tempat CodePipeline yang tersedia.

Bagaimana integrasi bekerja

Anda mulai dengan mengatur dan mengonfigurasi. CodePipeline Ini termasuk menambahkan konfigurasi Anda ke penyimpanan kode CodePipeline yang didukung. Selanjutnya, Anda mengatur AWS AppConfig lingkungan Anda dengan melakukan tugas-tugas berikut:

- Buat namespace dan profil konfigurasi
- Pilih strategi penerapan yang telah ditentukan sebelumnya atau buat sendiri

Setelah menyelesaikan tugas ini, Anda membuat pipeline CodePipeline yang menentukan AWS AppConfig sebagai penyedia penerapan. Anda kemudian dapat membuat perubahan pada konfigurasi Anda dan mengunggahnya ke toko CodePipeline kode Anda. Mengunggah konfigurasi baru secara otomatis memulai penerapan baru di. CodePipeline Setelah penerapan selesai, Anda dapat memverifikasi perubahan Anda. Untuk informasi tentang membuat pipeline yang ditetapkan AWS AppConfig sebagai penyedia penerapan, lihat Tutorial: Membuat Pipeline yang Digunakan AWS AppConfig sebagai Penyedia Deployment di Panduan Pengguna.AWS CodePipeline

Mengambil flag fitur dan data konfigurasi di AWS AppConfig

Aplikasi Anda mengambil flag fitur dan data konfigurasi formulir gratis dengan membuat sesi konfigurasi menggunakan layanan AWS AppConfig Data. Jika Anda menggunakan salah satu metode pengambilan sederhana yang dijelaskan di bagian ini, ekstensi AWS AppConfig Agen Lambda AWS AppConfig atau Agen mengelola serangkaian panggilan API dan token sesi atas nama Anda. Anda mengonfigurasi AWS AppConfig Agen sebagai host lokal dan memiliki polling agen AWS AppConfig untuk pembaruan konfigurasi. Agen memanggil tindakan StartConfigurationSesi dan GetLatestKonfigurasi API dan menyimpan data konfigurasi Anda secara lokal. Untuk mengambil data, aplikasi Anda membuat panggilan HTTP ke server localhost. AWS AppConfig Agen mendukung beberapa kasus penggunaan, seperti yang dijelaskan dalamMetode pengambilan yang disederhanakan.

Jika mau, Anda dapat memanggil tindakan API ini secara manual untuk mengambil konfigurasi. Proses API berfungsi sebagai berikut:

Aplikasi Anda membuat sesi konfigurasi menggunakan tindakan `StartConfigurationSession` API. Klien sesi Anda kemudian melakukan panggilan berkala `GetLatestConfiguration` untuk memeriksa dan mengambil data terbaru yang tersedia.

Saat memanggil`StartConfigurationSession`, kode Anda mengirimkan pengenal (ID atau nama) dari AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi yang dilacak sesi.

Sebagai tanggapan, AWS AppConfig berikan `InitialConfigurationToken` untuk diberikan kepada klien sesi dan digunakan saat pertama kali memanggil `GetLatestConfiguration` sesi itu.

Saat meneleponGetLatestConfiguration, kode klien Anda mengirimkan `ConfigurationToken` nilai terbaru yang dimilikinya dan diterima sebagai tanggapan:

- `NextPollConfigurationToken`: `ConfigurationToken` nilai yang akan digunakan pada panggilan berikutnya keGetLatestConfiguration.
- Konfigurasi: data terbaru yang ditujukan untuk sesi tersebut. Ini mungkin kosong jika klien sudah memiliki versi konfigurasi terbaru.

Bagian ini mencakup informasi berikut.

Daftar Isi

- Metode pengambilan yang disederhanakan
- Mengambil konfigurasi dengan langsung memanggil API

Tentang layanan pesawat AWS AppConfig data

Pada 18 November 2021, AWS AppConfig merilis layanan pesawat data baru. Layanan ini menggantikan proses pengambilan data konfigurasi sebelumnya dengan menggunakan tindakan `GetConfiguration` API. Layanan data plane menggunakan dua tindakan API baru, StartConfigurationSession dan GetLatestConfiguration. Layanan pesawat data juga menggunakan titik akhir baru.

Jika Anda mulai menggunakan AWS AppConfig sebelum 28 Januari 2022, layanan mungkin memanggil tindakan `GetConfiguration` API secara langsung atau mungkin menggunakan klien yang disediakan oleh AWS, seperti ekstensi AWS AppConfig Agen Lambda, untuk memanggil tindakan API ini. Jika Anda memanggil tindakan `GetConfiguration` API secara langsung, ambil langkah-langkah untuk menggunakan tindakan `StartConfigurationSession` dan `GetLatestConfiguration` API. Jika Anda menggunakan ekstensi AWS AppConfig Agen Lambda, lihat bagian berjudul Bagaimana perubahan ini berdampak pada ekstensi Agen AWS AppConfig Lambda nanti dalam topik ini.

Tindakan API bidang data baru menawarkan manfaat berikut dibandingkan tindakan `GetConfiguration` API, yang sekarang sudah tidak digunakan lagi.

1. Anda tidak perlu mengelola `ClientID` parameter. Dengan layanan pesawat data, dikelola `ClientID` secara internal oleh token sesi yang dibuat oleh`StartConfigurationSession`.
2. Anda tidak perlu lagi menyertakan `ClientConfigurationVersion` untuk menunjukkan versi cache dari data konfigurasi Anda. Dengan layanan pesawat data, dikelola `ClientConfigurationVersion` secara internal oleh token sesi yang dibuat oleh`StartConfigurationSession`.
3. Titik akhir khusus baru untuk panggilan API bidang data meningkatkan struktur kode dengan memisahkan panggilan bidang kontrol dan pesawat data.
4. Layanan pesawat data baru meningkatkan ekstensibilitas future untuk operasi pesawat data. Dengan memanfaatkan sesi konfigurasi yang mengelola pengambilan data konfigurasi, AWS AppConfig tim dapat membuat perangkat tambahan yang lebih kuat di masa depan.

Migrasi dari ke **`GetConfigurationGetLatestConfiguration`**

Untuk mulai menggunakan layanan pesawat data baru, Anda perlu memperbarui kode yang memanggil tindakan `GetConfiguration` API. Mulai sesi konfigurasi dengan menggunakan aksi `StartConfigurationSession` API, lalu panggil tindakan `GetLatestConfiguration` API untuk mengambil data konfigurasi. Untuk meningkatkan kinerja, kami sarankan Anda menyimpan data konfigurasi Anda secara lokal. Untuk informasi selengkapnya, lihat Mengambil konfigurasi dengan langsung memanggil API.

Bagaimana perubahan ini berdampak pada ekstensi AWS AppConfig Agen Lambda

Perubahan ini tidak berdampak langsung pada cara kerja ekstensi AWS AppConfig Agen Lambda. Versi lama dari ekstensi AWS AppConfig Agen Lambda disebut tindakan `GetConfiguration` API atas nama Anda. Versi yang lebih baru memanggil tindakan API bidang data. Jika Anda menggunakan ekstensi AWS AppConfig Lambda, kami sarankan Anda memperbarui ekstensi Anda ke Nama Sumber Daya Amazon (ARN) terbaru dan memperbarui izin untuk panggilan API baru. Untuk informasi selengkapnya, lihat Mengambil data konfigurasi menggunakan ekstensi AWS AppConfig Agen Lambda.

Metode pengambilan yang disederhanakan

AWS AppConfig menawarkan beberapa metode yang disederhanakan untuk mengambil data konfigurasi. Jika Anda menggunakan flag AWS AppConfig fitur atau data konfigurasi formulir gratis dalam suatu AWS Lambda fungsi, Anda dapat menggunakan ekstensi AWS AppConfig Agen Lambda untuk mengambil konfigurasi. Jika Anda memiliki aplikasi yang berjalan di instans Amazon EC2, Anda dapat menggunakan AWS AppConfig Agen untuk mengambil konfigurasi. AWS AppConfig Agen juga mendukung aplikasi yang berjalan di Amazon Elastic Kubernetes Service (Amazon EKS) atau Amazon Elastic Container Service (Amazon ECS) image container.

Setelah Anda menyelesaikan penyiapan awal, metode pengambilan data konfigurasi ini lebih sederhana daripada memanggil AWS AppConfig API secara langsung. Mereka secara otomatis menerapkan praktik terbaik dan dapat menurunkan biaya penggunaan Anda AWS AppConfig sebagai akibat dari lebih sedikit panggilan API untuk mengambil konfigurasi.

Topik

- Mengambil data konfigurasi menggunakan ekstensi AWS AppConfig Agen Lambda
- Mengambil data konfigurasi dari instans Amazon EC2
- Mengambil data konfigurasi dari Amazon ECS dan Amazon EKS
- Fitur pengambilan tambahan

- AWS AppConfig Agen pengembangan lokal

Mengambil data konfigurasi menggunakan ekstensi AWS AppConfig Agen Lambda

AWS Lambda Ekstensi adalah proses pendamping yang menambah kemampuan fungsi Lambda. Ekstensi dapat dimulai sebelum fungsi dipanggil, dijalankan secara paralel dengan fungsi, dan terus berjalan setelah pemanggilan fungsi diproses. Intinya, ekstensi Lambda seperti klien yang berjalan secara paralel dengan pemanggilan Lambda. Klien paralel ini dapat berinteraksi dengan fungsi Anda kapan saja selama siklus hidupnya.

Jika Anda menggunakan flag AWS AppConfig fitur atau data konfigurasi dinamis lainnya dalam fungsi Lambda, maka sebaiknya Anda menambahkan ekstensi AWS AppConfig Agen Lambda sebagai lapisan ke fungsi Lambda Anda. Ini membuat flag fitur panggilan lebih sederhana, dan ekstensi itu sendiri mencakup praktik terbaik yang menyederhanakan penggunaan AWS AppConfig sekaligus mengurangi biaya. Pengurangan biaya dihasilkan dari lebih sedikit panggilan API ke AWS AppConfig layanan dan waktu pemrosesan fungsi Lambda yang lebih pendek. Untuk informasi selengkapnya tentang ekstensi Lambda, lihat Ekstensi Lambda di Panduan Pengembang.AWS Lambda

> **Note**
>
> AWS AppConfig Penetapan harga didasarkan pada berapa kali konfigurasi dipanggil dan diterima. Biaya Anda meningkat jika Lambda Anda melakukan beberapa cold start dan sering mengambil data konfigurasi baru.

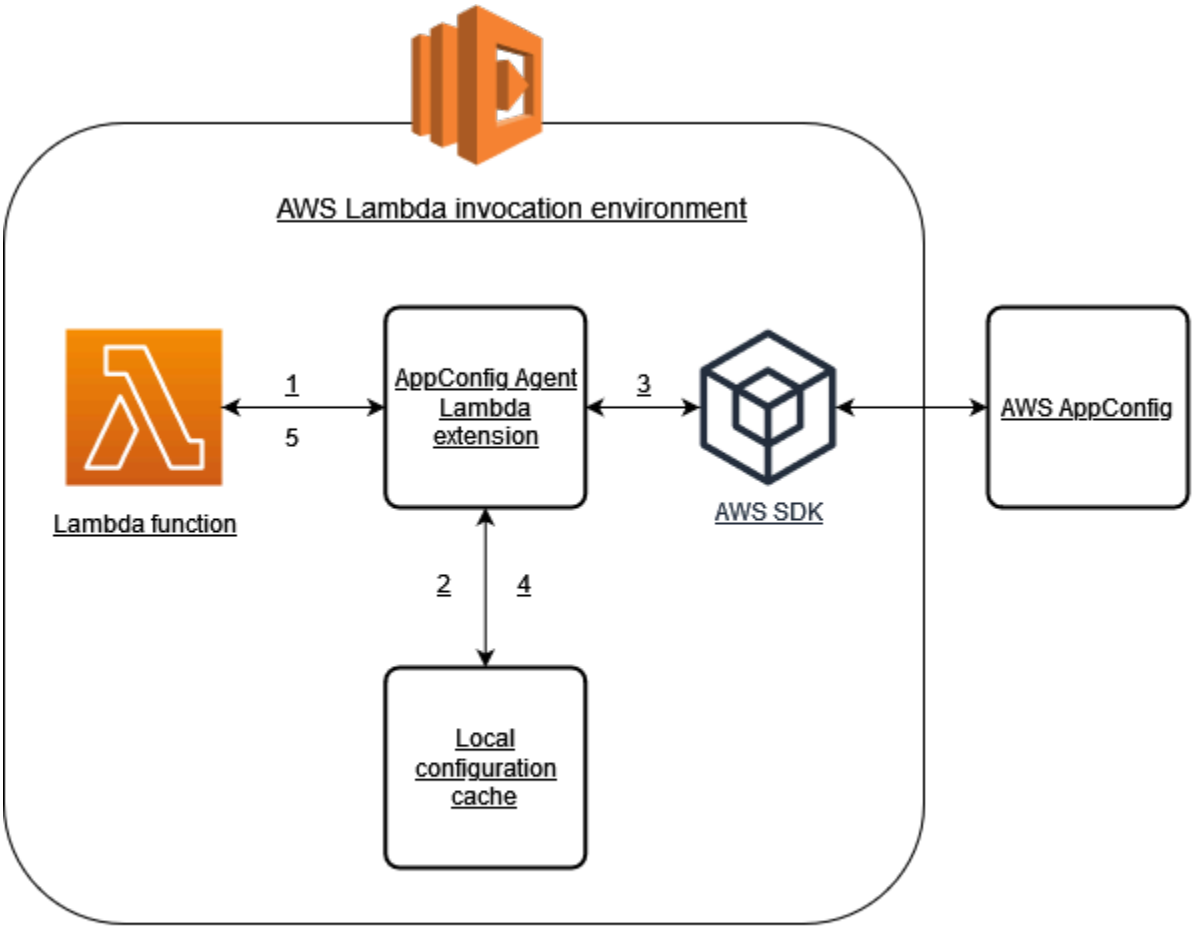
Topik ini mencakup informasi tentang ekstensi AWS AppConfig Agen Lambda dan prosedur cara mengonfigurasi ekstensi agar berfungsi dengan fungsi Lambda Anda.

Cara kerjanya

Jika Anda menggunakan AWS AppConfig untuk mengelola konfigurasi untuk fungsi Lambda tanpa ekstensi Lambda, maka Anda harus mengonfigurasi fungsi Lambda Anda untuk menerima pembaruan konfigurasi dengan mengintegrasikan dengan tindakan API Sesi dan Konfigurasi. StartConfiguration GetLatest

Mengintegrasikan ekstensi AWS AppConfig Agen Lambda dengan fungsi Lambda Anda menyederhanakan proses ini. Ekstensi menangani panggilan AWS AppConfig layanan, mengelola

cache lokal dari data yang diambil, melacak token konfigurasi yang diperlukan untuk panggilan layanan berikutnya, dan secara berkala memeriksa pembaruan konfigurasi di latar belakang. Diagram berikut menunjukkan cara kerjanya.



1. Anda mengonfigurasi ekstensi AWS AppConfig Agen Lambda sebagai lapisan fungsi Lambda Anda.
2. Untuk mengakses data konfigurasinya, fungsi Anda memanggil AWS AppConfig ekstensi pada titik akhir HTTP yang sedang `localhost:2772` berjalan.
3. Ekstensi mempertahankan cache lokal dari data konfigurasi. Jika data tidak ada dalam cache, ekstensi memanggil AWS AppConfig untuk mendapatkan data konfigurasi.
4. Setelah menerima konfigurasi dari layanan, ekstensi menyimpannya di cache lokal dan meneruskannya ke fungsi Lambda.

5. AWS AppConfig Ekstensi Agen Lambda secara berkala memeriksa pembaruan data konfigurasi Anda di latar belakang. Setiap kali fungsi Lambda Anda dipanggil, ekstensi memeriksa waktu yang telah berlalu sejak mengambil konfigurasi. Jika waktu yang berlalu lebih besar dari interval polling yang dikonfigurasi, ekstensi akan memanggil AWS AppConfig untuk memeriksa data yang baru digunakan, memperbarui cache lokal jika ada perubahan, dan mengatur ulang waktu yang telah berlalu.

> **Note**
>
> - Lambda membuat instance terpisah yang sesuai dengan tingkat konkurensi yang dibutuhkan fungsi Anda. Setiap instance diisolasi dan memelihara cache lokal sendiri dari data konfigurasi Anda. Untuk informasi selengkapnya tentang instans dan konkurensi Lambda, lihat Mengelola konkurensi untuk fungsi Lambda.
> - Jumlah waktu yang diperlukan agar perubahan konfigurasi muncul dalam fungsi Lambda, setelah Anda menerapkan konfigurasi yang diperbarui AWS AppConfig, bergantung pada strategi penerapan yang Anda gunakan untuk penerapan dan interval polling yang Anda konfigurasikan untuk ekstensi.

Sebelum Anda mulai

Sebelum Anda mengaktifkan ekstensi AWS AppConfig Agen Lambda, lakukan hal berikut:

- Atur konfigurasi dalam fungsi Lambda Anda sehingga Anda dapat mengeksternalisasikannya. AWS AppConfig
- Buat AWS AppConfig artefak dan data konfigurasi, termasuk flag fitur atau data konfigurasi bentuk bebas. Untuk informasi selengkapnya, lihat Membuat flag fitur dan data konfigurasi formulir gratis di AWS AppConfig.
- Tambahkan `appconfig:StartConfigurationSession` dan `appconfig:GetLatestConfiguration` ke kebijakan AWS Identity and Access Management (IAM) yang digunakan oleh peran eksekusi fungsi Lambda. Untuk informasi selengkapnya, lihat peran AWS Lambda eksekusi di Panduan AWS Lambda Pengembang. Untuk informasi selengkapnya tentang AWS AppConfig izin, lihat Tindakan, sumber daya, dan kunci kondisi AWS AppConfig di Referensi Otorisasi Layanan.

Menambahkan ekstensi AWS AppConfig Agen Lambda

Untuk menggunakan ekstensi AWS AppConfig Agen Lambda, Anda perlu menambahkan ekstensi ke Lambda Anda. Ini dapat dilakukan dengan menambahkan ekstensi AWS AppConfig Agen Lambda ke fungsi Lambda Anda sebagai lapisan atau dengan mengaktifkan ekstensi pada fungsi Lambda sebagai gambar wadah.

> **Note**
>
> AWS AppConfig Ekstensi ini agnostik runtime dan mendukung semua runtime.

Menambahkan ekstensi AWS AppConfig Agen Lambda dengan menggunakan layer dan ARN

Untuk menggunakan ekstensi AWS AppConfig Agen Lambda, Anda menambahkan ekstensi ke fungsi Lambda Anda sebagai lapisan. Untuk informasi tentang cara menambahkan lapisan ke fungsi Anda, lihat Mengonfigurasi ekstensi di Panduan AWS Lambda Pengembang. Nama ekstensi di AWS Lambda konsol adalah AWS- AppConfig -Extension. Perhatikan juga bahwa ketika Anda menambahkan ekstensi sebagai lapisan ke Lambda Anda, Anda harus menentukan Nama Sumber Daya Amazon (ARN). Pilih ARN dari salah satu daftar berikut yang sesuai dengan platform dan Wilayah AWS tempat Anda membuat Lambda.

- platform x86-64
- Platform ARM64

Jika Anda ingin menguji ekstensi sebelum menambahkannya ke fungsi Anda, Anda dapat memverifikasi bahwa itu berfungsi dengan menggunakan contoh kode berikut.

```
import urllib.request


def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name'
    config = urllib.request.urlopen(url).read()
    return config
```

Untuk mengujinya, buat fungsi Lambda baru untuk Python, tambahkan ekstensi, lalu jalankan fungsi Lambda. Setelah Anda menjalankan fungsi Lambda, fungsi AWS AppConfig Lambda mengembalikan

konfigurasi yang Anda tentukan untuk jalur http://localhost:2772. Untuk informasi tentang membuat fungsi Lambda, lihat Membuat fungsi Lambda dengan konsol di Panduan Pengembang.AWS Lambda

Untuk menambahkan ekstensi AWS AppConfig Agen Lambda sebagai gambar kontainer, lihat. Menggunakan gambar kontainer untuk menambahkan ekstensi AWS AppConfig Agen Lambda

Mengkonfigurasi ekstensi AWS AppConfig Agen Lambda

Anda dapat mengonfigurasi ekstensi dengan mengubah variabel AWS Lambda lingkungan berikut. Untuk informasi selengkapnya, lihat Menggunakan variabel AWS Lambda lingkungan di Panduan AWS Lambda Pengembang.

Prefetching data konfigurasi

Variabel lingkungan `AWS_APPCONFIG_EXTENSION_PREFETCH_LIST` dapat meningkatkan waktu start-up fungsi Anda. Ketika ekstensi AWS AppConfig Agen Lambda diinisialisasi, ekstensi ini akan mengambil konfigurasi yang ditentukan sebelum AWS AppConfig Lambda mulai menginisialisasi fungsi Anda dan memanggil handler Anda. Dalam beberapa kasus, data konfigurasi sudah tersedia di cache lokal sebelum fungsi Anda memintanya.

Untuk menggunakan kemampuan prefetch, atur nilai variabel lingkungan ke jalur yang sesuai dengan data konfigurasi Anda. Misalnya, jika konfigurasi Anda sesuai dengan aplikasi, lingkungan, dan profil konfigurasi masing-masing bernama “my_application”, “my_environment”, dan “my_configuration_data”, jalurnya adalah. `/applications/my_application/environments/my_environment/configurations/my_configuration_data` Anda dapat menentukan beberapa item konfigurasi dengan mencantumkannya sebagai daftar yang dipisahkan koma (Jika Anda memiliki nama sumber daya yang menyertakan koma, gunakan nilai ID sumber daya alih-alih namanya).

Mengakses data konfigurasi dari akun lain

Ekstensi AWS AppConfig Agen Lambda dapat mengambil data konfigurasi dari akun lain dengan menentukan peran IAM yang memberikan izin ke data. Untuk mengaturnya, ikuti langkah-langkah berikut:

1. Di akun tempat AWS AppConfig digunakan untuk mengelola data konfigurasi, buat peran dengan kebijakan kepercayaan yang memberi akun yang menjalankan fungsi Lambda akses ke `appconfig:StartConfigurationSession` `appconfig:GetLatestConfiguration` dan

tindakan, bersama dengan ARN paral atau lengkap yang sesuai dengan sumber daya konfigurasi. AWS AppConfig

2. Di akun yang menjalankan fungsi Lambda, tambahkan variabel `AWS_APPCONFIG_EXTENSION_ROLE_ARN` lingkungan ke fungsi Lambda dengan ARN peran yang dibuat pada langkah 1.
3. (Opsional) Jika diperlukan, ID eksternal dapat ditentukan menggunakan variabel `AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID` lingkungan. Demikian pula, nama sesi dapat dikonfigurasi menggunakan variabel `AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME` lingkungan.

> **Note**
>
> Perhatikan informasi berikut.
>
> - Ekstensi AWS AppConfig Agen Lambda hanya dapat mengambil data dari satu akun. Jika Anda menentukan peran IAM, ekstensi tidak akan dapat mengambil data konfigurasi dari akun tempat fungsi Lambda berjalan.
> - AWS Lambda mencatat informasi tentang ekstensi AWS AppConfig Agen Lambda dan fungsi Lambda dengan menggunakan Amazon Logs. CloudWatch

Variabel lingkungan	Detail	Nilai default
`AWS_APPCONFIG_EXTENSION_HTTP_PORT`	Variabel lingkungan ini menentukan port di mana server HTTP lokal yang menghosting ekstensi berjalan.	2772
`AWS_APPCONFIG_EXTENSION_LOG_LEVEL`	Variabel lingkungan ini menentukan log AWS AppConfig khusus ekstensi mana yang dikirim ke Amazon CloudWatch Logs untuk suatu fungsi. Nilai yang valid dan tidak peka huruf besar/kecil	`info`

Variabel lingkungan	Detail	Nilai default
	adalah:`debug`,,, `infowarn`, `error` dan. `none` Debug mencakup informasi terperinci, termasuk informasi waktu, tentang ekstensi.	
`AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS`	Variabel lingkungan ini mengonfigurasi jumlah maksimum koneksi yang digunakan ekstensi untuk mengambil konfigurasi dari. AWS AppConfig	3
`AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS`	Variabel lingkungan ini mengontrol seberapa sering ekstensi melakukan polling AWS AppConfig untuk konfigurasi yang diperbarui dalam hitungan detik.	45
`AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS`	Variabel lingkungan ini mengontrol jumlah waktu maksimum, dalam milidetik, ekstensi menunggu respons dari AWS AppConfig saat menyegarkan data dalam cache. Jika AWS AppConfig tidak merespons dalam jumlah waktu yang ditentukan, ekstensi melewatkan interval polling ini dan mengembalikan data cache yang diperbarui sebelumnya.	3000

Variabel lingkungan	Detail	Nilai default
`AWS_APPCONFIG_EXTENSION_PREFETCH_LIST`	Variabel lingkungan ini menentukan data konfigurasi yang ekstensi mulai mengambil sebelum fungsi diinisialisasi dan handler berjalan. Ini dapat mengurangi waktu mulai dingin fungsi secara signifikan.	Tidak ada
`AWS_APPCONFIG_EXTENSION_PROXY_HEADERS`	Variabel lingkungan ini menentukan header yang diperlukan oleh proxy yang direferensikan dalam variabel lingkungan. `AWS_APPCONFIG_EXTENSION_PROXY_URL` Nilainya adalah daftar header yang dipisahkan koma. Setiap header menggunakan formulir berikut: `"header: value"`	Tidak ada
`AWS_APPCONFIG_EXTENSION_PROXY_URL`	Variabel lingkungan ini menentukan URL proxy yang akan digunakan untuk koneksi dari AWS AppConfig ekstensi ke Layanan AWS. `HTTPS`dan `HTTP` URL didukung.	Tidak ada
`AWS_APPCONFIG_EXTENSION_ROLE_ARN`	Variabel lingkungan ini menentukan peran IAM ARN sesuai dengan peran yang harus diasumsikan oleh AWS AppConfig ekstensi untuk mengambil konfigurasi.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`	Variabel lingkungan ini menentukan id eksternal untuk digunakan dalam hubungannya dengan peran ARN yang diasumsikan.	Tidak ada
`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada
`AWS_APPCONFIG_EXTENSION_SERVICE_REGION`	Variabel lingkungan ini menentukan Wilayah alternatif yang harus digunakan ekstensi untuk memanggil AWS AppConfig layanan. Saat tidak ditentukan, ekstensi menggunakan titik akhir di Wilayah saat ini.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`AWS_APPCONFIG_EXTENSION_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk memanfaatkan fitur per-konfigurasi tambahan seperti pengambilan multi-akun dan menyimpan konfigurasi ke disk. Anda dapat memasukkan salah satu nilai berikut: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Untuk informasi lebih lanjut tentang fitur ini, lihat Fitur pengambilan tambahan.	true
`AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesaikan startup.	true

Mengambil satu atau beberapa flag dari konfigurasi flag fitur

Untuk konfigurasi flag fitur (konfigurasi tipe`AWS.AppConfig.FeatureFlags`), ekstensi Lambda memungkinkan Anda untuk mengambil satu flag atau subset flag dalam konfigurasi. Mengambil satu atau dua flag berguna jika Lambda Anda hanya perlu menggunakan beberapa flag dari profil konfigurasi. Contoh berikut menggunakan Python.

> **Note**
>
> Kemampuan untuk memanggil bendera fitur tunggal atau subset flag dalam konfigurasi hanya tersedia di ekstensi Agen AWS AppConfig Lambda versi 2.0.45 dan yang lebih tinggi.

Anda dapat mengambil data AWS AppConfig konfigurasi dari titik akhir HTTP lokal. Untuk mengakses bendera tertentu atau daftar bendera, gunakan parameter `?flag=flag_name` kueri untuk profil AWS AppConfig konfigurasi.

Untuk mengakses bendera tunggal dan atributnya

```
import urllib.request

def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name'
    config = urllib.request.urlopen(url).read()
    return config
```

Untuk mengakses beberapa bendera dan atributnya

```
import urllib.request

def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two'
    config = urllib.request.urlopen(url).read()
    return config
```

Melihat log AWS AppConfig ekstensi Agen Lambda

Anda dapat melihat data log untuk ekstensi AWS AppConfig Agen Lambda di log. AWS Lambda Entri log diawali dengan. `appconfig agent` Inilah contohnya.

```
[appconfig agent] 2024/05/07 04:19:01 ERROR retrieve failure for
 'SourceEventConfig:SourceEventConfigEnvironment:SourceEventConfigProfile':
 StartConfigurationSession: api error AccessDenied: User:
 arn:aws:sts::0123456789:assumed-role/us-east-1-LambdaRole/extension1 is not authorized
```

```
to perform: sts:AssumeRole on resource: arn:aws:iam::0123456789:role/test1 (retry in
60s)
```

Versi ekstensi AWS AppConfig Agen Lambda yang tersedia

Topik ini mencakup informasi tentang versi ekstensi AWS AppConfig Agen Lambda. Ekstensi AWS AppConfig Agen Lambda mendukung fungsi Lambda yang dikembangkan untuk platform x86-64 dan ARM64 (Graviton2). Agar berfungsi dengan baik, fungsi Lambda Anda harus dikonfigurasi untuk menggunakan Nama Sumber Daya Amazon (ARN) tertentu untuk Wilayah AWS tempat saat ini di-host. Anda dapat melihat Wilayah AWS dan detail ARN nanti di bagian ini.

Important

Perhatikan detail penting berikut tentang ekstensi AWS AppConfig Agen Lambda.

- Tindakan `GetConfiguration` API tidak digunakan lagi pada 28 Januari 2022. Panggilan untuk menerima data konfigurasi harus menggunakan `GetLatestConfiguration` API `StartConfigurationSession` dan sebagai gantinya. Jika Anda menggunakan versi ekstensi AWS AppConfig Agen Lambda yang dibuat sebelum 28 Januari 2022, Anda mungkin harus mengonfigurasi izin ke API baru. Untuk informasi selengkapnya, lihat Tentang layanan pesawat AWS AppConfig data.
- AWS AppConfig mendukung semua versi yang tercantum dalamVersi ekstensi yang lebih lama. Kami menyarankan Anda memperbarui secara berkala ke versi terbaru untuk memanfaatkan peningkatan ekstensi.

Topik

AWS AppConfig Catatan rilis Agen Lambda Extension

Tabel berikut menjelaskan perubahan yang dibuat untuk versi terbaru dari ekstensi AWS AppConfig Lambda.

Versi	Tanggal peluncuran	Catatan
2.0.358	12/01/2023	Ditambahkan dukungan untuk fitur pengambilan berikut: • Pengambilan multi-akun: Gunakan AWS AppConfig Agen dari primer atau pengambilan Akun AWS untuk mengambil data konfigurasi dari beberapa akun vendor. • Tulis salinan konfigurasi ke disk: Gunakan AWS AppConfig Agen untuk menulis data konfigurasi ke disk. Fitur ini memungkin kan pelanggan dengan aplikasi yang membaca data konfigurasi dari disk untuk diintegrasikan AWS AppConfig.
2.0.181	08/14/2023	Menambahkan dukungan untuk Israel (Tel Aviv) Wilayah AWS il-central-1.
2.0.165	02/21/2023	Perbaikan bug minor. Tidak lagi membatasi penggunaa n ekstensi ke versi runtime tertentu melalui konsol. AWS Lambda Ditambahkan dukungan untuk yang berikut Wilayah AWS: • Timur Tengah (UEA), me-central-1

Versi	Tanggal peluncuran	Catatan
		• Asia Pasifik (Hyderabad), ap-south-2 • Asia Pasifik (Melbourne), ap-southeast-4 • Eropa (Spanyol), eu-south-2 • Eropa (Zürich), eu-central-2
2.0.122	23/08/2022	Menambahkan dukungan untuk proxy tunneling, yang dapat dikonfigurasi dengan variabel `AWS_APPCONFIG_EXTENSION_PROXY_URL` dan `AWS_APPCONFIG_EXTENSION_PROXY_HEADERS` lingkungan. Menambahkan .NET 6 sebagai runtime. Untuk informasi lebih lanjut tentang variabel lingkungan, lihat Mengkonfigurasi ekstensi AWS AppConfig Agen Lambda.
2.0.58	05/03/2022	Peningkatan dukungan untuk prosesor Graviton2 (ARM64) di Lambda.

Versi	Tanggal peluncuran	Catatan
2.0.45	03/15/2022	Menambahkan dukungan untuk memanggil bendera fitur tunggal. Sebelumnya, pelanggan menyebut flag fitur yang dikelompokkan ke dalam profil konfigurasi dan harus mengurai respons sisi klien. Dengan rilis ini, pelanggan dapat menggunak an `flag=<flag-name>` parameter saat memanggil titik akhir localhost HTTP untuk mendapatkan nilai dari satu flag. Juga menambahkan dukungan awal untuk prosesor Graviton2 (ARM64).

Menemukan nomor versi ekstensi Lambda Anda

Gunakan prosedur berikut untuk menemukan nomor versi ekstensi AWS AppConfig Agen Lambda yang saat ini dikonfigurasi. Agar berfungsi dengan baik, fungsi Lambda Anda harus dikonfigurasi untuk menggunakan Nama Sumber Daya Amazon (ARN) tertentu untuk Wilayah AWS tempat saat ini di-host.

1. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih fungsi Lambda di mana Anda ingin menambahkan layer. `AWS-AppConfig-Extension`
3. Di bagian Layers, pilih Add a layer.
4. Di bagian Pilih lapisan, pilih AWS- AppConfig -Extension dari daftar AWS layer.
5. Gunakan daftar Versi untuk memilih nomor versi.
6. Pilih Tambahkan.
7. Gunakan tab Uji untuk menguji fungsi.

8. Setelah pengujian selesai, lihat output log. Temukan versi ekstensi AWS AppConfig Agen Lambda di bagian Detail Eksekusi. Versi ini harus sesuai dengan URL yang diperlukan untuk versi tersebut.

platform x86-64

Saat Anda menambahkan ekstensi sebagai lapisan ke Lambda Anda, Anda harus menentukan ARN. Pilih ARN dari tabel berikut yang sesuai dengan Wilayah AWS tempat Anda membuat Lambda. ARN ini untuk fungsi Lambda yang dikembangkan untuk platform x86-64.

Versi 2.0.358

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106`

Wilayah	ARN
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:76`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:76`

Wilayah	ARN
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107`

Wilayah	ARN
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:83`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:22`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54`

Platform ARM64

Saat Anda menambahkan ekstensi sebagai lapisan ke Lambda Anda, Anda harus menentukan ARN. Pilih ARN dari tabel berikut yang sesuai dengan Wilayah AWS tempat Anda membuat Lambda. ARN ini untuk fungsi Lambda yang dikembangkan untuk platform ARM64.

Versi 2.0.358

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5`

Wilayah	ARN
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16`

Wilayah	ARN
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11`

Wilayah	ARN
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:5`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:13`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:5`

Versi ekstensi yang lebih lama

Bagian ini mencantumkan ARN dan Wilayah AWS untuk versi lama dari ekstensi AWS AppConfig Lambda. Daftar ini tidak berisi informasi untuk semua versi ekstensi AWS AppConfig Agen Lambda sebelumnya, tetapi akan diperbarui saat versi baru dirilis.

Versi ekstensi yang lebih lama (platform x86-64)

Tabel berikut mencantumkan ARN dan versi Wilayah AWS lama dari ekstensi AWS AppConfig Agen Lambda yang dikembangkan untuk platform x86-64.

Tanggal diganti dengan ekstensi yang lebih baru: 12/01/2023

Versi 2.0.181

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:113`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:81`

Wilayah	ARN
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142`

Wilayah	ARN
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:68`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:68`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91`

Wilayah	ARN
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:73`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:7`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34`

Wilayah	ARN
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46`

Tanggal diganti dengan ekstensi yang lebih baru: 08/14/2023

Versi 2.0.165

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143`

Wilayah	ARN
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26`

Wilayah	ARN
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:66`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:66`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60`

Wilayah	ARN
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44`

Tanggal diganti dengan ekstensi yang lebih baru: 02/21/2023

Versi 2.0.122

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59`

Wilayah	ARN
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:52`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:52`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59`

Wilayah	ARN
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29`

Tanggal diganti dengan ekstensi yang lebih baru: 08/23/2022

Versi 2.0.58

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50`

Wilayah	ARN
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46`

Wilayah	ARN
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23`

Tanggal diganti dengan ekstensi yang lebih baru: 04/21/2022

Versi 2.0.45

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49`

Wilayah	ARN
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45`

Wilayah	ARN
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22`

Tanggal diganti dengan ekstensi yang lebih baru: 03/15/2022

Versi 2.0.30

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47`

Wilayah	ARN
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54`

Wilayah	ARN
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20`

Versi ekstensi yang lebih lama (platform ARM64)

Tabel berikut mencantumkan ARN dan versi Wilayah AWS lama dari ekstensi AWS AppConfig Agen Lambda yang dikembangkan untuk platform ARM64.

Tanggal diganti dengan ekstensi yang lebih baru: 12/01/2023

Versi 2.0.181

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:46`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:33`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:1`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:48`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:1`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:36`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:48`

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43`

Wilayah	ARN
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1`

Tanggal diganti dengan ekstensi yang lebih baru: 03/30/2023

Versi 2.0.165

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43`

Wilayah	ARN
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41`
Asia Pacific (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34`

Tanggal diganti dengan ekstensi yang lebih baru: 02/21/2023

Versi 2.0.122

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:15`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:11`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:16`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:13`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:20`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:11`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:15`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:16`

Wilayah	ARN
Asia Pacific (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:13`

Tanggal diganti dengan ekstensi yang lebih baru: 08/23/2022

Versi 2.0.58

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7`

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3`
Asia Pacific (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2`

Tanggal diganti dengan ekstensi yang lebih baru: 04/21/2022

Versi 2.0.45

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1`

Wilayah	ARN
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2`
Asia Pacific (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1`

Menggunakan gambar kontainer untuk menambahkan ekstensi AWS AppConfig Agen Lambda

Anda dapat mengemas ekstensi AWS AppConfig Agen Lambda Anda sebagai gambar kontainer untuk mengunggahnya ke registri penampung yang dihosting di Amazon Elastic Container Registry (Amazon ECR).

Untuk menambahkan ekstensi AWS AppConfig Agen Lambda sebagai gambar wadah Lambda

1. Masukkan Wilayah AWS dan Nama Sumber Daya Amazon (ARN) di AWS Command Line Interface (AWS CLI) seperti yang ditunjukkan di bawah ini. Ganti nilai Region dan ARN dengan Region Anda dan ARN yang cocok untuk mengambil salinan layer Lambda. AWS AppConfig menyediakan ARN untuk platform x86-64 dan ARM64.

   ```
   aws lambda get-layer-version-by-arn \
     --region Wilayah AWS \
     --arn extension ARN
   ```

   Inilah contohnya.

   ```
   aws lambda get-layer-version-by-arn \
     --region us-east-1 \
     --arn arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128
   ```

   Responsnya terlihat seperti berikut:

   ```
   {
     "LayerVersionArn": "arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-
   Extension:128",
     "Description": "AWS AppConfig extension: Use dynamic configurations deployed via
    AWS AppConfig for your AWS Lambda functions",
     "CreatedDate": "2021-04-01T02:37:55.339+0000",
     "LayerArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension",

     "Content": {
       "CodeSize": 5228073,
       "CodeSha256": "8otOgbLQbexpUm3rKlMhvcE6Q5TvwcLCKrc4Oe+vmMY=",
       "Location" : "S3-Bucket-Location-URL"
     },
   ```

```
  "Version": 30,
  "CompatibleRuntimes": [
    "python3.8",
    "python3.7",
    "nodejs12.x",
    "ruby2.7"
  ],
}
```

2. Dalam respons di atas, nilai yang dikembalikan `Location` adalah URL bucket Amazon Simple Storage Service (Amazon S3) yang berisi ekstensi Lambda. Rekatkan URL ke browser web Anda untuk mengunduh file ekstensi.zip Lambda.

   > **Note**
   >
   > URL bucket Amazon S3 hanya tersedia selama 10 menit.

   (Opsional) Atau, Anda juga dapat menggunakan `curl` perintah berikut untuk mengunduh ekstensi Lambda.

   ```
   curl -o extension.zip "S3-Bucket-Location-URL"
   ```

   (Opsional) Atau, Anda dapat menggabungkan Langkah 1 dan Langkah 2 untuk mengambil ARN dan mengunduh file ekstensi.zip sekaligus.

   ```
   aws lambda get-layer-version-by-arn \
     --arn extension ARN \
     | jq -r '.Content.Location' \
     | xargs curl -o extension.zip
   ```

3. Tambahkan baris berikut di Anda `Dockerfile` untuk menambahkan ekstensi ke gambar kontainer Anda.

   ```
   COPY extension.zip extension.zip
   RUN yum install -y unzip \
     && unzip extension.zip /opt \
     && rm -f extension.zip
   ```

4. Pastikan bahwa peran eksekusi fungsi Lambda memiliki appconfig: permission set. GetConfiguration

Contoh

Bagian ini mencakup contoh untuk mengaktifkan ekstensi AWS AppConfig Agen Lambda pada fungsi Python Lambda berbasis gambar kontainer.

1. Buat `Dockerfile` yang mirip dengan yang berikut ini.

```
FROM public.ecr.aws/lambda/python:3.8 AS builder
COPY extension.zip extension.zip
RUN yum install -y unzip \
  && unzip extension.zip -d /opt \
  && rm -f extension.zip

FROM public.ecr.aws/lambda/python:3.8
COPY --from=builder /opt /opt
COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. Unduh layer ekstensi ke direktori yang sama dengan file`Dockerfile`.

```
aws lambda get-layer-version-by-arn \
  --arn extension ARN \
  | jq -r '.Content.Location' \
  | xargs curl -o extension.zip
```

3. Buat file Python bernama `index.py` dalam direktori yang sama dengan file. `Dockerfile`

```
import urllib.request

def handler(event, context):
    return {
       # replace parameters here with your application, environment, and
 configuration names
       'configuration': get_configuration('myApp', 'myEnv', 'myConfig'),
    }

def get_configuration(app, env, config):
    url = f'http://localhost:2772/applications/{app}/environments/{env}/
configurations/{config}'
```

```
    return urllib.request.urlopen(url).read()
```

4. Jalankan langkah-langkah berikut untuk membuat `docker` gambar dan mengunggahnya ke Amazon ECR.

```
// set environment variables
export ACCOUNT_ID = <YOUR_ACCOUNT_ID>
export REGION = <AWS_REGION>

// create an ECR repository
aws ecr create-repository --repository-name test-repository

// build the docker image
docker build -t test-image .

// sign in to AWS
aws ecr get-login-password \
  | docker login \
  --username AWS \
  --password-stdin "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com"

// tag the image
docker tag test-image:latest "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/test-
repository:latest"

// push the image to ECR
docker push "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/test-repository:latest"
```

5. Gunakan gambar Amazon ECR yang Anda buat di atas untuk membuat fungsi Lambda. Untuk informasi selengkapnya tentang fungsi Lambda sebagai wadah, lihat Membuat fungsi Lambda yang didefinisikan sebagai gambar kontainer.
6. Pastikan bahwa peran eksekusi fungsi Lambda memiliki appconfig: permission set. GetConfiguration

Integrasi dengan OpenAPI

Anda dapat menggunakan spesifikasi YAMAL berikut untuk OpenAPI untuk membuat SDK menggunakan alat seperti OpenAPI Generator. Anda dapat memperbarui spesifikasi ini untuk menyertakan nilai hardcode untuk Aplikasi, Lingkungan, atau Konfigurasi. Anda juga dapat menambahkan jalur tambahan (jika Anda memiliki beberapa jenis konfigurasi) dan menyertakan

skema konfigurasi untuk menghasilkan model yang diketik khusus konfigurasi untuk klien SDK Anda. Untuk informasi selengkapnya tentang OpenAPI (yang juga dikenal sebagai Swagger), lihat spesifikasi OpenAPI.

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: AppConfig Agent Lambda extension API
  description: An API model for the AppConfig Agent Lambda extension.
servers:
  - url: https://localhost:{port}/
    variables:
      port:
        default:
          '2772'
paths:
  /applications/{Application}/environments/{Environment}/configurations/
{Configuration}:
    get:
      operationId: getConfiguration
      tags:
        - configuration
      parameters:
        - in: path
          name: Application
          description: The application for the configuration to get. Specify either the
 application name or the application ID.
          required: true
          schema:
            type: string
        - in: path
          name: Environment
          description: The environment for the configuration to get. Specify either the
 environment name or the environment ID.
          required: true
          schema:
            type: string
        - in: path
          name: Configuration
          description: The configuration to get. Specify either the configuration name
 or the configuration ID.
          required: true
          schema:
```

```
            type: string
      responses:
        200:
          headers:
            ConfigurationVersion:
              schema:
                type: string
          content:
            application/octet-stream:
              schema:
                type: string
                format: binary
          description: successful config retrieval
        400:
          description: BadRequestException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        404:
          description: ResourceNotFoundException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        500:
          description: InternalServerException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        502:
          description: BadGatewayException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        504:
          description: GatewayTimeoutException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
```

```
components:
  schemas:
    Error:
      type: string
      description: The response error
```

Mengambil data konfigurasi dari instans Amazon EC2

Anda dapat berintegrasi AWS AppConfig dengan aplikasi yang berjalan di instans Amazon Elastic Compute Cloud (Amazon EC2) Linux dengan menggunakan Agen. AWS AppConfig Agen meningkatkan pemrosesan dan manajemen aplikasi dengan cara berikut:

- Agen memanggil AWS AppConfig atas nama Anda dengan menggunakan peran AWS Identity and Access Management (IAM) dan mengelola cache lokal data konfigurasi. Dengan menarik data konfigurasi dari cache lokal, aplikasi Anda memerlukan lebih sedikit pembaruan kode untuk mengelola data konfigurasi, mengambil data konfigurasi dalam milidetik, dan tidak terpengaruh oleh masalah jaringan yang dapat mengganggu panggilan untuk data tersebut. *
- Agen menawarkan pengalaman asli untuk mengambil dan menyelesaikan flag AWS AppConfig fitur.
- Di luar kotak, agen menyediakan praktik terbaik untuk strategi caching, interval polling, dan ketersediaan data konfigurasi lokal sambil melacak token konfigurasi yang diperlukan untuk panggilan layanan berikutnya.
- Saat berjalan di latar belakang, agen secara berkala melakukan polling bidang AWS AppConfig data untuk pembaruan data konfigurasi. Aplikasi Anda dapat mengambil data dengan menghubungkan ke localhost pada port 2772 (nilai port default yang dapat disesuaikan) dan memanggil HTTP GET untuk mengambil data.

*AWS AppConfig Agen menyimpan data saat pertama kali layanan mengambil data konfigurasi Anda. Untuk alasan ini, panggilan pertama untuk mengambil data lebih lambat dari panggilan berikutnya.

Topik

- Langkah 5: (Diperlukan) Mengambil data konfigurasi
- Langkah 6 (Opsional, tetapi disarankan): Mengotomatiskan pembaruan ke Agen AWS AppConfig

Langkah 1: (Diperlukan) Membuat sumber daya dan mengonfigurasi izin

Untuk berintegrasi AWS AppConfig dengan aplikasi yang berjalan di instans Amazon EC2, Anda harus membuat AWS AppConfig artefak dan data konfigurasi, termasuk flag fitur atau data konfigurasi bentuk bebas. Untuk informasi selengkapnya, lihat Membuat flag fitur dan data konfigurasi formulir gratis di AWS AppConfig.

Untuk mengambil data konfigurasi yang dihosting oleh AWS AppConfig, aplikasi Anda harus dikonfigurasi dengan akses ke bidang AWS AppConfig data. Untuk memberikan akses kepada aplikasi Anda, perbarui kebijakan izin IAM yang ditetapkan ke peran instans Amazon EC2. Secara khusus, Anda harus menambahkan `appconfig:StartConfigurationSession` dan `appconfig:GetLatestConfiguration` tindakan ke kebijakan. Inilah contohnya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:StartConfigurationSession",
                "appconfig:GetLatestConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk informasi selengkapnya tentang menambahkan izin ke kebijakan, lihat Menambahkan dan menghapus izin identitas IAM di Panduan Pengguna IAM.

Langkah 2: (Diperlukan) Menginstal dan memulai AWS AppConfig Agen di instans Amazon EC2

AWS AppConfig Agen di-host di bucket Amazon Simple Storage Service (Amazon S3) yang dikelola oleh Amazon Simple Storage Service (Amazon S3). AWS Gunakan prosedur berikut untuk menginstal versi terbaru agen pada instance Linux Anda. Jika aplikasi Anda didistribusikan

di beberapa instance, maka Anda harus melakukan prosedur ini pada setiap instance yang menghosting aplikasi.

> **Note**
>
> Perhatikan informasi berikut:
>
> - AWS AppConfig Agen tersedia untuk sistem operasi Linux yang menjalankan kernel versi 4.15 atau lebih tinggi. Sistem berbasis Debian, seperti Ubuntu, tidak didukung.
> - Agen mendukung arsitektur x86_64 dan ARM64.
> - Untuk aplikasi terdistribusi, sebaiknya tambahkan perintah instal dan startup ke data pengguna Amazon EC2 dari grup Auto Scaling Anda. Jika Anda melakukannya, setiap instance menjalankan perintah secara otomatis. Untuk informasi selengkapnya, lihat Menjalankan perintah pada instans Linux Anda saat diluncurkan di Panduan Pengguna Amazon EC2. Selain itu, lihat Tutorial: Mengonfigurasi data pengguna untuk mengambil status siklus hidup target melalui metadata instans di Panduan Pengguna Auto Scaling Amazon EC2.
> - Prosedur di seluruh topik ini menjelaskan cara melakukan tindakan seperti menginstal agen dengan masuk ke instance untuk menjalankan perintah. Anda dapat menjalankan perintah dari mesin klien lokal dan menargetkan satu atau lebih instance dengan menggunakan Run Command, yang merupakan kemampuan. AWS Systems Manager Untuk informasi selengkapnya, lihat Run Command AWS Systems Manager di Panduan Pengguna AWS Systems Manager .
> - AWS AppConfig Agen di Amazon EC2 Linux instans adalah layanan. `systemd`

Untuk menginstal dan memulai AWS AppConfig Agen pada sebuah instance

1. Masuk ke instance Linux Anda.
2. Buka terminal dan jalankan perintah berikut dengan izin Administrator untuk arsitektur x86_64:

   ```
   sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/x86_64/latest/aws-appconfig-agent.rpm
   ```

   Untuk arsitektur ARM64, jalankan perintah berikut:

```
sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/arm64/latest/aws-appconfig-agent.rpm
```

Jika Anda ingin menginstal versi AWS AppConfig Agen tertentu, ganti `latest` di URL dengan nomor versi tertentu. Berikut adalah contoh untuk x86_64:

```
sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/x86_64/2.0.2/aws-appconfig-agent.rpm
```

3. Jalankan perintah berikut untuk memulai agen:

```
sudo systemctl start aws-appconfig-agent
```

4. Jalankan perintah berikut untuk memverifikasi agen sedang berjalan:

```
sudo systemctl status aws-appconfig-agent
```

Jika berhasil, perintah mengembalikan informasi seperti berikut:

```
aws-appconfig-agent.service - aws-appconfig-agent
  ...
  Active: active (running) since Mon 2023-07-26 00:00:00 UTC; 0s ago
  ...
```

Note

Untuk menghentikan agen, jalankan perintah berikut:

```
sudo systemctl stop aws-appconfig-agent
```

Langkah 3: (Opsional, tetapi disarankan) Mengirim file log ke CloudWatch Log

Secara default, AWS AppConfig Agen menerbitkan log ke STDERR. Systemd mengalihkan STDOUT dan STDERR untuk semua layanan yang berjalan pada instance Linux ke jurnal systemd. Anda dapat melihat dan mengelola data log di jurnal systemd jika Anda menjalankan AWS AppConfig Agen hanya pada satu atau dua instance. Solusi yang lebih baik, solusi yang sangat kami rekomendasikan

untuk aplikasi terdistribusi, adalah menulis file log ke disk dan kemudian menggunakan CloudWatch agen Amazon untuk mengunggah data log ke AWS cloud. Selain itu, Anda dapat mengonfigurasi CloudWatch agen untuk menghapus file log lama dari instance Anda, yang mencegah instance Anda kehabisan ruang disk.

Untuk mengaktifkan logging ke disk, Anda harus mengatur variabel `LOG_PATH` lingkungan, seperti yang dijelaskan dalamLangkah 4: (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon EC2.

Untuk memulai dengan CloudWatch agen, lihat Mengumpulkan metrik dan log dari instans Amazon EC2 dan server lokal dengan CloudWatch agen di Panduan Pengguna Amazon. CloudWatch Anda dapat menggunakan Quick Setup, kemampuan Systems Manager untuk menginstal CloudWatch agen dengan cepat. Untuk informasi selengkapnya, lihat Pengaturan Cepat Manajemen Host di Panduan AWS Systems Manager Pengguna.

> **Warning**
>
> Jika Anda memilih untuk menulis file log ke disk tanpa menggunakan CloudWatch agen, Anda harus menghapus file log lama. AWS AppConfig Agen secara otomatis memutar file log setiap jam. Jika Anda tidak menghapus file log lama, instance Anda dapat kehabisan ruang disk.

Setelah Anda menginstal CloudWatch agen pada instance Anda, buat file konfigurasi CloudWatch agen. File konfigurasi menginstruksikan CloudWatch agen tentang cara bekerja dengan file log AWS AppConfig Agen. Untuk informasi selengkapnya tentang membuat file konfigurasi CloudWatch agen, lihat Membuat file konfigurasi CloudWatch agen.

Tambahkan `logs` bagian berikut ke file konfigurasi CloudWatch agen pada instance dan simpan perubahan Anda:

```
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/path_you_specified_for_logging",
          "log_group_name": "${YOUR_LOG_GROUP_NAME}/aws-appconfig-agent.log",
          "auto_removal": true
        },
```

```
        ...
      ]
    },
    ...
  },
  ...
}
```

Jika nilainya `auto_removal` adalah`true`, CloudWatch agen secara otomatis menghapus file log AWS AppConfig Agen yang diputar.

Langkah 4: (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon EC2

Anda dapat mengonfigurasi AWS AppConfig Agen untuk Amazon EC2 dengan menggunakan variabel lingkungan. Untuk mengatur variabel lingkungan untuk `systemd` layanan, Anda membuat file unit drop-in. Contoh berikut menunjukkan cara membuat file unit drop-in untuk mengatur tingkat logging AWS AppConfig Agen ke`DEBUG`.

Contoh cara membuat file unit drop-in untuk variabel lingkungan

1. Masuk ke instance Linux Anda.
2. Buka terminal dan jalankan perintah berikut dengan izin Administrator. Perintah membuat direktori konfigurasi:

   ```
   sudo mkdir /etc/systemd/system/aws-appconfig-agent.service.d
   ```

3. Jalankan perintah berikut untuk membuat file unit drop-in. Ganti *`file_name dengan nama`* untuk file tersebut. Ekstensi harus`.conf`:

   ```
   sudo touch /etc/systemd/system/aws-appconfig-agent.service.d/file_name.conf
   ```

4. Masukkan informasi dalam file unit drop-in. Contoh berikut menambahkan `Service` bagian yang mendefinisikan variabel lingkungan. Contoh ini menetapkan level log AWS AppConfig Agen ke`DEBUG`.

   ```
   [Service]
   Environment=LOG_LEVEL=DEBUG
   ```

5. Jalankan perintah berikut untuk memuat ulang konfigurasi systemd:

```
sudo systemctl daemon-reload
```

6. Jalankan perintah berikut untuk me-restart AWS AppConfig Agen:

```
sudo systemctl restart aws-appconfig-agent
```

Anda dapat mengonfigurasi AWS AppConfig Agen untuk Amazon EC2 dengan menentukan variabel lingkungan berikut dalam file unit drop-in.

Variabel lingkungan	Detail	Nilai default

`ACCESS_TOKEN`	Variabel lingkungan ini mendefinisikan token yang harus disediakan saat meminta data konfigurasi dari server HTTP agen. Nilai token harus diatur dalam header otorisasi permintaan HTTP dengan jenis otorisasi. `Bearer` Inilah contohnya.
	`GET /applications/my_app/...`
	`Host: localhost:2772`
	`Authorization: Bearer <token value>`
		Tidak ada

`BACKUP_DIRECTORY`	Variabel lingkungan ini memungkinkan AWS AppConfig Agen untuk menyimpan cadangan dari setiap konfigurasi yang diambil ke direktori yang ditentukan.	Tidak ada

Variabel lingkungan	Detail	Nilai default
	Important Konfigurasi yang dicadangkan ke disk tidak dienkripsi. Jika konfigurasi Anda berisi data sensitif, AWS AppConfig sarankan Anda mempraktikkan prinsip hak istimewa paling sedikit dengan izin sistem file Anda. Untuk informasi selengkapnya, lihat Keamanan di AWS AppConfig.	
`HTTP_PORT`	Variabel lingkungan ini menentukan port di mana server HTTP untuk agen berjalan.	2772

Variabel lingkungan	Detail	Nilai default
`LOG_LEVEL`	Variabel lingkungan ini menentukan tingkat detail yang dicatat agen. Setiap level mencakup level saat ini dan semua level yang lebih tinggi. Variabelnya peka huruf besar/kecil. Dari sebagian besar hingga yang paling tidak detail, level log adalah:`debug`,`info`,`warn`,`erro` dan`none`. `Debug`mencakup informasi terperinci, termasuk informasi waktu, tentang agen.	`info`
`LOG_PATH`	Lokasi disk tempat log ditulis. Jika tidak ditentukan, log ditulis ke stderr.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk memanfaatkan fitur per-konfigurasi tambahan seperti pengambilan multi-akun dan menyimpan konfigurasi ke disk. Anda dapat memasukkan salah satu nilai berikut: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Untuk informasi lebih lanjut tentang fitur ini, lihat Fitur pengambilan tambahan.	true
`MAX_CONNECTIONS`	Variabel lingkungan ini mengonfigurasi jumlah maksimum koneksi yang digunakan agen untuk mengambil konfigurasi dari. AWS AppConfig	3

Variabel lingkungan	Detail	Nilai default
`POLL_INTERVAL`	Variabel lingkungan ini mengontrol seberapa sering agen melakukan polling AWS AppConfig untuk data konfigurasi yang diperbaru i. Anda dapat menentukan sejumlah detik untuk interval tersebut. Anda juga dapat menentukan angka dengan satuan waktu: s untuk detik, m untuk menit, dan h selama berjam-jam. Jika unit tidak ditentukan, agen default ke detik. Misalnya, 60, 60-an, dan 1m menghasilkan interval jajak pendapat yang sama.	45 detik
`PREFETCH_LIST`	Variabel lingkungan ini menentukan data konfigura si permintaan agen AWS AppConfig segera setelah dimulai.	Tidak ada

Variabel lingkungan	Detail	Nilai default
PRELOAD_BACKUPS	Jika disetel ke`true`, AWS AppConfig Agen memuat cadangan konfigurasi yang ditemukan di memori BACKUP_DIRECTORY ke dalam dan segera memeriksa untuk melihat apakah ada versi yang lebih baru dari layanan. Jika disetel ke`false`, AWS AppConfig Agen hanya memuat konten dari cadangan konfigurasi jika tidak dapat mengambil data konfigurasi dari layanan, misalnya jika ada masalah dengan jaringan Anda.	true
PROXY_HEADERS	Variabel lingkungan ini menentukan header yang diperlukan oleh proxy yang direferensikan dalam variabel lingkungan. PROXY_URL Nilainya adalah daftar header yang dipisahkan koma. Setiap header menggunakan formulir berikut. `"header: value"`	Tidak ada

Variabel lingkungan	Detail	Nilai default
PROXY_URL	Variabel lingkungan ini menentukan URL proxy yang akan digunakan untuk koneksi dari agen ke Layanan AWS, termasuk AWS AppConfig . HTTPSdan HTTP URL didukung.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`REQUEST_TIMEOUT`	Variabel lingkungan ini mengontrol jumlah waktu agen menunggu respons. AWS AppConfig Jika layanan tidak merespons, permintaan gagal. Jika permintaan untuk pengambilan data awal, agen mengembalikan kesalahan ke aplikasi Anda. Jika batas waktu terjadi selama pemeriksaan latar belakang untuk data yang diperbarui, agen mencatat kesalahan dan mencoba lagi setelah penundaan singkat. Anda dapat menentukan jumlah milidetik untuk batas waktu. Anda juga dapat menentukan angka dengan satuan waktu: ms untuk milidetik dan s untuk detik. Jika unit tidak ditentukan, agen default ke milidetik. Sebagai contoh, 5000, 5000 ms dan 5s menghasilkan nilai batas waktu permintaan yang sama.	3000 milidetik

Variabel lingkungan	Detail	Nilai default
ROLE_ARN	Variabel lingkungan ini menentukan Amazon Resource Name (ARN) dari peran IAM. AWS AppConfig Agen mengasumsikan peran ini untuk mengambil data konfigurasi.	Tidak ada
ROLE_EXTERNAL_ID	Variabel lingkungan ini menentukan ID eksternal untuk digunakan dengan peran ARN yang diasumsikan.	Tidak ada
ROLE_SESSION_NAME	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada
SERVICE_REGION	Variabel lingkungan ini menentukan alternatif Wilayah AWS yang digunakan AWS AppConfig Agen untuk memanggil AWS AppConfig layanan. Jika dibiarkan tidak terdefinisi, agen mencoba menentukan Wilayah saat ini. Jika tidak bisa, agen gagal untuk memulai.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesa ikan startup.	true

Langkah 5: (Diperlukan) Mengambil data konfigurasi

Anda dapat mengambil data konfigurasi dari AWS AppConfig Agen dengan menggunakan panggilan localhost HTTP. Contoh berikut digunakan `curl` dengan klien HTTP. Anda dapat memanggil agen menggunakan klien HTTP yang tersedia yang didukung oleh bahasa aplikasi atau pustaka yang tersedia, termasuk AWS SDK.

Untuk mengambil konten lengkap dari konfigurasi yang diterapkan

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name"
```

Untuk mengambil bendera tunggal dan atributnya dari AWS AppConfig konfigurasi tipe **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name"
```

Untuk mengakses beberapa flag dan atributnya dari AWS AppConfig konfigurasi tipe **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two"
```

Langkah 6 (Opsional, tetapi disarankan): Mengotomatiskan pembaruan ke Agen AWS AppConfig

AWS AppConfig Agen diperbarui secara berkala. Untuk memastikan Anda menjalankan versi terbaru AWS AppConfig Agen pada instans Anda, kami sarankan Anda menambahkan perintah berikut ke data pengguna Amazon EC2 Anda. Anda dapat menambahkan perintah ke data pengguna baik pada instans atau grup EC2 Auto Scaling. Skrip menginstal dan memulai versi terbaru agen setiap kali instance dimulai atau reboot.

```
#!/bin/bash
# install the latest version of the agent
yum install -y https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/
linux/x86_64/latest/aws-appconfig-agent.rpm
# optional: configure the agent
mkdir /etc/systemd/system/aws-appconfig-agent.service.d
echo "${MY_AGENT_CONFIG}" > /etc/systemd/system/aws-appconfig-agent.service.d/
overrides.conf
systemctl daemon-reload
# start the agent
systemctl start aws-appconfig-agent
```

Mengambil data konfigurasi dari Amazon ECS dan Amazon EKS

Anda dapat berintegrasi AWS AppConfig dengan Amazon Elastic Container Service (Amazon ECS) dan Amazon Elastic Kubernetes Service (Amazon EKS) dengan menggunakan Agen. AWS AppConfig Agen berfungsi sebagai wadah sespan yang berjalan bersama aplikasi penampung Amazon ECS dan Amazon EKS Anda. Agen meningkatkan pemrosesan dan manajemen aplikasi kontainer dengan cara berikut:

- Agen memanggil AWS AppConfig atas nama Anda dengan menggunakan peran AWS Identity and Access Management (IAM) dan mengelola cache lokal data konfigurasi. Dengan menarik data konfigurasi dari cache lokal, aplikasi Anda memerlukan lebih sedikit pembaruan kode untuk mengelola data konfigurasi, mengambil data konfigurasi dalam milidetik, dan tidak terpengaruh oleh masalah jaringan yang dapat mengganggu panggilan untuk data tersebut. *
- Agen menawarkan pengalaman asli untuk mengambil dan menyelesaikan flag AWS AppConfig fitur.
- Di luar kotak, agen menyediakan praktik terbaik untuk strategi caching, interval polling, dan ketersediaan data konfigurasi lokal sambil melacak token konfigurasi yang diperlukan untuk panggilan layanan berikutnya.

- Saat berjalan di latar belakang, agen secara berkala melakukan polling bidang AWS AppConfig data untuk pembaruan data konfigurasi. Aplikasi kontainer Anda dapat mengambil data dengan menghubungkan ke localhost pada port 2772 (nilai port default yang dapat disesuaikan) dan memanggil HTTP GET untuk mengambil data.
- AWS AppConfig Agen memperbarui data konfigurasi dalam kontainer Anda tanpa harus memulai ulang atau mendaur ulang kontainer tersebut.

*AWS AppConfig Agen menyimpan data saat pertama kali layanan mengambil data konfigurasi Anda. Untuk alasan ini, panggilan pertama untuk mengambil data lebih lambat dari panggilan berikutnya.

Topik

Sebelum Anda mulai

Untuk berintegrasi AWS AppConfig dengan aplikasi container, Anda harus membuat AWS AppConfig artefak dan data konfigurasi, termasuk flag fitur atau data konfigurasi bentuk bebas. Untuk informasi selengkapnya, lihat Membuat flag fitur dan data konfigurasi formulir gratis di AWS AppConfig.

Untuk mengambil data konfigurasi yang dihosting oleh AWS AppConfig, aplikasi kontainer Anda harus dikonfigurasi dengan akses ke bidang AWS AppConfig data. Untuk memberikan akses kepada aplikasi Anda, perbarui kebijakan izin IAM yang digunakan oleh peran IAM layanan kontainer Anda. Secara khusus, Anda harus menambahkan `appconfig:StartConfigurationSession` dan `appconfig:GetLatestConfiguration` tindakan ke kebijakan. Peran IAM layanan kontainer meliputi:

- Peran tugas Amazon ECS
- Peran simpul Amazon EKS
- Peran eksekusi AWS Fargate (Fargate) pod (jika container Amazon EKS Anda menggunakan Fargate untuk pemrosesan komputasi)

Untuk informasi selengkapnya tentang menambahkan izin ke kebijakan, lihat Menambahkan dan menghapus izin identitas IAM di Panduan Pengguna IAM.

Memulai AWS AppConfig agen untuk integrasi Amazon ECS

Kontainer sespan AWS AppConfig Agen tersedia secara otomatis di lingkungan Amazon ECS Anda. Untuk menggunakan wadah sespan AWS AppConfig Agen, Anda harus memulainya.

Untuk memulai Amazon ECS (konsol)

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di panel navigasi, pilih Definisi tugas.
3. Pilih definisi tugas untuk aplikasi Anda, lalu pilih revisi terbaru.
4. Pilih Buat revisi baru, Buat revisi baru.
5. Pilih Tambahkan lebih banyak wadah.
6. Untuk Nama, masukkan nama unik untuk wadah AWS AppConfig Agen.
7. Untuk URI Gambar, masukkan: **`public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x`**
8. Untuk wadah Essential, pilih Ya.
9. Di bagian Pemetaan port, pilih Tambahkan pemetaan port.
10. Untuk port Kontainer, masukkan**2772**.

> **Note**
>
> AWS AppConfig Agen berjalan pada port 2772, secara default. Anda dapat menentukan port yang berbeda.

11. Pilih Buat. Amazon ECS membuat revisi kontainer baru dan menampilkan detailnya.
12. Di panel navigasi, pilih Cluster, lalu pilih cluster aplikasi Anda dalam daftar.
13. Pada tab Layanan, pilih layanan untuk aplikasi Anda.
14. Pilih Perbarui.
15. Di bawah konfigurasi Deployment, untuk Revisi, pilih revisi terbaru.
16. Pilih Perbarui. Amazon ECS menerapkan definisi tugas terbaru.
17. Setelah penerapan selesai, Anda dapat memverifikasi bahwa AWS AppConfig Agen berjalan di tab Konfigurasi dan tugas. Pada tab Tugas, pilih tugas yang sedang berjalan.

18. Di bagian Kontainer, verifikasi bahwa kontainer AWS AppConfig Agen terdaftar.
19. Untuk memverifikasi bahwa AWS AppConfig Agen dimulai, pilih tab Log. Temukan pernyataan seperti berikut untuk wadah AWS AppConfig Agen: `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Anda dapat menyesuaikan perilaku default AWS AppConfig Agen dengan memasukkan atau mengubah variabel lingkungan. Untuk informasi tentang variabel lingkungan yang tersedia, lihatMenggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS. Untuk informasi tentang cara mengubah variabel lingkungan di Amazon ECS, lihat Meneruskan variabel lingkungan ke wadah di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Memulai AWS AppConfig agen untuk integrasi Amazon EKS

Kontainer sespan AWS AppConfig Agen tersedia secara otomatis di lingkungan Amazon EKS Anda. Untuk menggunakan wadah sespan AWS AppConfig Agen, Anda harus memulainya. Prosedur berikut menjelaskan cara menggunakan alat baris `kubectl` perintah Amazon EKS untuk membuat perubahan dalam `kubeconfig` file untuk aplikasi penampung Anda. Untuk informasi selengkapnya tentang membuat atau mengedit `kubeconfig` file, lihat Membuat atau memperbarui file kubeconfig untuk klaster Amazon EKS di Panduan Pengguna Amazon EKS.

Untuk memulai AWS AppConfig Agen (alat baris perintah kubectl)

1. Buka `kubeconfig` file Anda dan verifikasi bahwa aplikasi Amazon EKS Anda berjalan sebagai penerapan kontainer tunggal. Isi file akan terlihat mirip dengan yang berikut ini.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-application-label
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: my-application-label
  template:
    metadata:
      labels:
        app: my-application-label
    spec:
      containers:
      - name: my-app
        image: my-repo/my-image
        imagePullPolicy: IfNotPresent
```

2. Tambahkan detail definisi kontainer AWS AppConfig Agen ke file penerapan YAMM Anda.

```
- name: appconfig-agent
        image: public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x
        ports:
        - name: http
          containerPort: 2772
          protocol: TCP
        env:
        - name: SERVICE_REGION
          value: region
        imagePullPolicy: IfNotPresent
```

> **Note**
>
> Perhatikan informasi berikut.
>
> - AWS AppConfig Agen berjalan pada port 2772, secara default. Anda dapat menentukan port yang berbeda.
> - Anda dapat menyesuaikan perilaku default AWS AppConfig Agen dengan memasukkan variabel lingkungan. Untuk informasi selengkapnya, lihat Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS.
> - Untuk *SERVICE_REGION*, tentukan Wilayah AWS kode (misalnya,`us-west-1`) tempat AWS AppConfig Agen mengambil data konfigurasi.

3. Jalankan perintah berikut di `kubectl` alat untuk menerapkan perubahan pada cluster Anda.

```
kubectl apply -f my-deployment.yml
```

4. Setelah penerapan selesai, verifikasi bahwa AWS AppConfig Agen sedang berjalan. Gunakan perintah berikut untuk melihat file log pod aplikasi.

   ```
   kubectl logs -n my-namespace -c appconfig-agent my-pod
   ```

   Temukan pernyataan seperti berikut untuk wadah AWS AppConfig Agen: `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Anda dapat menyesuaikan perilaku default AWS AppConfig Agen dengan memasukkan atau mengubah variabel lingkungan. Untuk informasi tentang variabel lingkungan yang tersedia, lihatMenggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS.

Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS

Anda dapat mengonfigurasi AWS AppConfig Agen dengan mengubah variabel lingkungan berikut untuk wadah agen Anda.

Variabel lingkungan	Detail	Nilai default
`ACCESS_TOKEN`	Variabel lingkungan ini mendefinisikan token yang harus disediakan saat meminta data konfigurasi dari server HTTP agen. Nilai token harus diatur dalam header otorisasi permintaan HTTP dengan jenis otorisasi. `Bearer` Inilah contohnya.	Tidak ada

Variabel lingkungan	Detail	Nilai default

	`GET /applications/my_app/...`
	`Host: localhost:2772`
	
	`Authorization: Bearer <token value>`	

`BACKUP_DIRECTORY`	Variabel lingkungan ini memungkinkan AWS AppConfig Agen untuk menyimpan cadangan dari setiap konfigurasi yang diambil ke direktori yang ditentukan. **Important** Konfigurasi yang dicadangkan ke disk tidak dienkripsi. Jika konfigurasi Anda berisi data sensitif, AWS AppConfig sarankan Anda mempraktikkan prinsip hak istimewa paling sedikit dengan izin sistem file Anda. Untuk informasi selengkapnya, lihat Keamanan di AWS AppConfig.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`HTTP_PORT`	Variabel lingkungan ini menentukan port di mana server HTTP untuk agen berjalan.	2772
`LOG_LEVEL`	Variabel lingkungan ini menentukan tingkat detail yang dicatat agen. Setiap level mencakup level saat ini dan semua level yang lebih tinggi. Variabelnya peka huruf besar/kecil. Dari sebagian besar hingga yang paling tidak detail, level log adalah:`debug`,`info`,`warn`,`erro` dan`none`. `Debug`mencakup informasi terperinci, termasuk informasi waktu, tentang agen.	`info`

Variabel lingkungan	Detail	Nilai default
`MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk memanfaatkan fitur per-konfigurasi tambahan seperti pengambilan multi-akun dan menyimpan konfigurasi ke disk. Anda dapat memasukkan salah satu nilai berikut: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Untuk informasi lebih lanjut tentang fitur ini, lihat Fitur pengambilan tambahan.	true
`MAX_CONNECTIONS`	Variabel lingkungan ini mengonfigurasi jumlah maksimum koneksi yang digunakan agen untuk mengambil konfigurasi dari. AWS AppConfig	3

Variabel lingkungan	Detail	Nilai default
`POLL_INTERVAL`	Variabel lingkungan ini mengontrol seberapa sering agen melakukan polling AWS AppConfig untuk data konfigurasi yang diperbaru i. Anda dapat menentukan sejumlah detik untuk interval tersebut. Anda juga dapat menentukan angka dengan satuan waktu: s untuk detik, m untuk menit, dan h selama berjam-jam. Jika unit tidak ditentukan, agen default ke detik. Misalnya, 60, 60-an, dan 1m menghasilkan interval jajak pendapat yang sama.	45 detik
`PREFETCH_LIST`	Variabel lingkungan ini menentukan data konfigura si permintaan agen AWS AppConfig segera setelah dimulai.	Tidak ada

Variabel lingkungan	Detail	Nilai default
PRELOAD_BACKUPS	Jika disetel ke`true`, AWS AppConfig Agen memuat cadangan konfigurasi yang ditemukan di memori BACKUP_DIRECTORY ke dalam dan segera memeriksa untuk melihat apakah ada versi yang lebih baru dari layanan. Jika disetel ke`false`, AWS AppConfig Agen hanya memuat konten dari cadangan konfigurasi jika tidak dapat mengambil data konfigurasi dari layanan, misalnya jika ada masalah dengan jaringan Anda.	true
PROXY_HEADERS	Variabel lingkungan ini menentukan header yang diperlukan oleh proxy yang direferensikan dalam variabel lingkungan. PROXY_URL Nilainya adalah daftar header yang dipisahkan koma. Setiap header menggunakan formulir berikut. `"header: value"`	Tidak ada

Variabel lingkungan	Detail	Nilai default
PROXY_URL	Variabel lingkungan ini menentukan URL proxy yang akan digunakan untuk koneksi dari agen ke Layanan AWS, termasuk AWS AppConfig . HTTPSdan HTTP URL didukung.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`REQUEST_TIMEOUT`	Variabel lingkungan ini mengontrol jumlah waktu agen menunggu respons. AWS AppConfig Jika layanan tidak merespons, permintaan gagal. Jika permintaan untuk pengambilan data awal, agen mengembalikan kesalahan ke aplikasi Anda. Jika batas waktu terjadi selama pemeriksaan latar belakang untuk data yang diperbarui, agen mencatat kesalahan dan mencoba lagi setelah penundaan singkat. Anda dapat menentukan jumlah milidetik untuk batas waktu. Anda juga dapat menentukan angka dengan satuan waktu: ms untuk milidetik dan s untuk detik. Jika unit tidak ditentukan, agen default ke milidetik. Sebagai contoh, 5000, 5000 ms dan 5s menghasilkan nilai batas waktu permintaan yang sama.	3000 milidetik

Variabel lingkungan	Detail	Nilai default
ROLE_ARN	Variabel lingkungan ini menentukan Amazon Resource Name (ARN) dari peran IAM. AWS AppConfig Agen mengasumsikan peran ini untuk mengambil data konfigurasi.	Tidak ada
ROLE_EXTERNAL_ID	Variabel lingkungan ini menentukan ID eksternal untuk digunakan dengan peran ARN yang diasumsikan.	Tidak ada
ROLE_SESSION_NAME	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada
SERVICE_REGION	Variabel lingkungan ini menentukan alternatif Wilayah AWS yang digunakan AWS AppConfig Agen untuk memanggil AWS AppConfig layanan. Jika dibiarkan tidak terdefinisi, agen mencoba menentukan Wilayah saat ini. Jika tidak bisa, agen gagal untuk memulai.	Tidak ada

Variabel lingkungan	Detail	Nilai default
`WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesa ikan startup.	true

Mengambil data konfigurasi

Anda dapat mengambil data konfigurasi dari AWS AppConfig Agen dengan menggunakan panggilan localhost HTTP. Contoh berikut digunakan `curl` dengan klien HTTP. Anda dapat menghubungi agen menggunakan klien HTTP yang tersedia yang didukung oleh bahasa aplikasi Anda atau pustaka yang tersedia.

> **Note**
>
> Untuk mengambil data konfigurasi jika aplikasi Anda menggunakan garis miring maju, misalnya “test-backend/test-service”, Anda harus menggunakan pengkodean URL.

Untuk mengambil konten lengkap dari konfigurasi yang diterapkan

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name"
```

Untuk mengambil bendera tunggal dan atributnya dari AWS AppConfig konfigurasi tipe **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name"
```

Untuk mengakses beberapa flag dan atributnya dari AWS AppConfig konfigurasi tipe **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two"
```

Fitur pengambilan tambahan

AWS AppConfig Agen menawarkan fitur tambahan berikut untuk membantu Anda mengambil konfigurasi untuk aplikasi Anda.

- Pengambilan multi-akun: Gunakan AWS AppConfig Agen dari primer atau pengambilan Akun AWS untuk mengambil data konfigurasi dari beberapa akun vendor.
- Tulis salinan konfigurasi ke disk: Gunakan AWS AppConfig Agen untuk menulis data konfigurasi ke disk. Fitur ini memungkinkan pelanggan dengan aplikasi yang membaca data konfigurasi dari disk untuk diintegrasikan AWS AppConfig.

Tentang manifestasi agen

Untuk mengaktifkan fitur AWS AppConfig Agen ini, Anda membuat manifes. Manifes adalah sekumpulan data konfigurasi yang Anda berikan untuk mengontrol tindakan yang dapat dilakukan agen. Sebuah manifes ditulis dalam JSON. Ini berisi satu set kunci tingkat atas yang sesuai dengan konfigurasi berbeda yang telah Anda gunakan. AWS AppConfig

Manifes dapat mencakup beberapa konfigurasi. Selanjutnya, setiap konfigurasi dalam manifes dapat mengidentifikasi satu atau lebih fitur agen yang akan digunakan untuk konfigurasi yang ditentukan. Isi manifes menggunakan format berikut:

```
{
    "application_name:environment_name:configuration_name": {
        "agent_feature_to_enable_1": {
            "feature-setting-key": "feature-setting-value"
        },
        "agent_feature_to_enable_2": {
            "feature-setting-key": "feature-setting-value"
        }
    }
}
```

Berikut adalah contoh JSON untuk manifes dengan dua konfigurasi. Konfigurasi pertama (*MyApp*) tidak menggunakan fitur AWS AppConfig Agen apa pun. Konfigurasi kedua (*My2ndApp*) menggunakan salinan konfigurasi tulis ke disk dan fitur pengambilan multi-akun:

```
{
        "MyApp:Test:MyAllowListConfiguration": {},

        "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
            "credentials": {
                "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
                "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
                "roleSessionName": "AwsAppConfigAgent",
                "credentialsDuration": "2h"
            },
            "writeTo": {
                "path": "/tmp/aws-appconfig/my-2nd-app/beta/my-enable-payments-feature-
flag-configuration.json"
            }
        }
    }
```

Cara memasok manifes agen

Anda dapat menyimpan manifes sebagai file di lokasi di mana AWS AppConfig Agen dapat membacanya. Atau, Anda dapat menyimpan manifes sebagai AWS AppConfig konfigurasi dan mengarahkan agen ke sana. Untuk menyediakan manifes agen, Anda harus menetapkan variabel `MANIFEST` lingkungan dengan salah satu nilai berikut:

Lokasi manifes	Nilai variabel lingkungan	Kasus penggunaan
File	berkas: /jalur/ke/agent-manifest.json	Gunakan metode ini jika manifes Anda tidak akan sering berubah.
AWS AppConfig konfigurasi	*nama aplikasi: nama-lingkungan : nama konfigurasi*	Gunakan metode ini untuk pembaruan dinamis. Anda dapat memperbarui dan menerapkan manifes yang disimpan AWS AppConfig sebagai konfigurasi dengan

Lokasi manifes	Nilai variabel lingkungan	Kasus penggunaan
		cara yang sama seperti Anda menyimpan AWS AppConfig konfigurasi lainnya.
Variabel lingkungan	Konten manifes (JSON)	Gunakan metode ini jika manifes Anda tidak akan sering berubah. Metode ini berguna dalam lingkunga n kontainer di mana lebih mudah untuk mengatur variabel lingkungan daripada mengekspos file.

Untuk informasi selengkapnya tentang menyetel variabel untuk AWS AppConfig Agen, lihat topik yang relevan untuk kasus penggunaan Anda:

- Mengkonfigurasi ekstensi AWS AppConfig Agen Lambda
- Menggunakan AWS AppConfig Agen dengan Amazon EC2
- Menggunakan AWS AppConfig Agen dengan Amazon ECS dan Amazon EKS

Pengambilan multi-akun

Anda dapat mengonfigurasi AWS AppConfig Agen untuk mengambil konfigurasi dari beberapa Akun AWS dengan memasukkan penggantian kredensi dalam manifes Agen. AWS AppConfig Penggantian kredensyal mencakup Nama Sumber Daya Amazon (ARN) peran AWS Identity and Access Management (IAM), ID peran, nama sesi, dan durasi berapa lama agen dapat mengambil peran tersebut.

Anda memasukkan detail ini di bagian “kredensil” di manifes. Bagian “kredensyal” menggunakan format berikut:

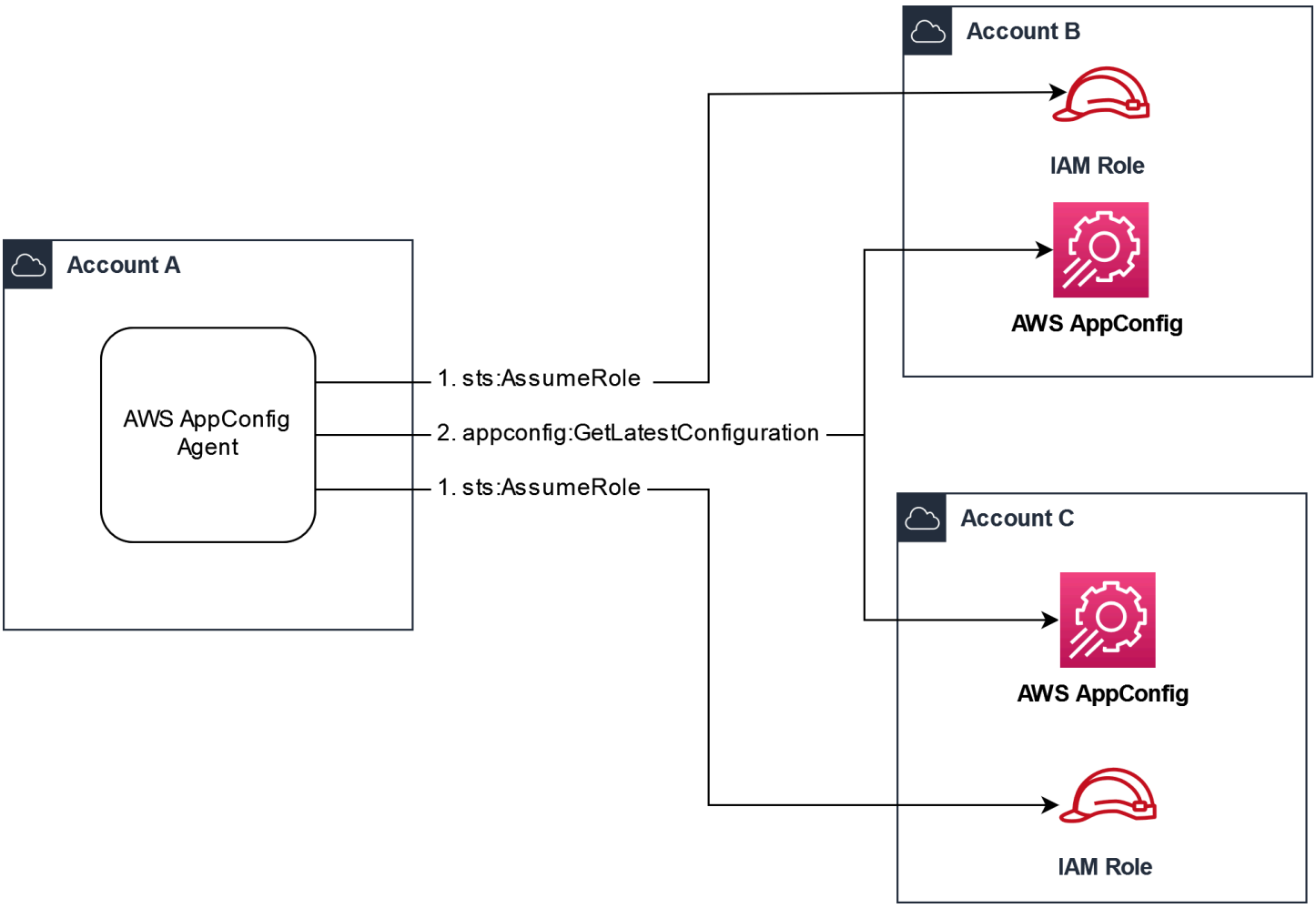
```
{
    "application_name:environment_name:configuration_name": {
        "credentials": {
            "roleArn": "arn:partition:iam::account_ID:role/roleName",
```

```
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
        }
    }
}
```

Inilah contohnya:

```
{
    "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
        "credentials": {
            "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
            "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
            "roleSessionName": "AWSAppConfigAgent",
            "credentialsDuration": "2h"
        }
    }
}
```

Sebelum mengambil konfigurasi, agen membaca detail kredensyal untuk konfigurasi dari manifes dan kemudian mengasumsikan peran IAM yang ditentukan untuk konfigurasi tersebut. Anda dapat menentukan kumpulan penggantian kredensyal yang berbeda untuk konfigurasi yang berbeda dalam satu manifes. Diagram berikut menunjukkan bagaimana AWS AppConfig Agen, saat berjalan di Akun A (akun pengambilan), mengasumsikan peran terpisah yang ditentukan untuk Akun B dan C (akun vendor) dan kemudian memanggil operasi API GetLatestKonfigurasi untuk mengambil data konfigurasi agar tidak AWS AppConfig berjalan di akun tersebut:



Konfigurasikan izin untuk mengambil data konfigurasi dari akun vendor

AWS AppConfig Agen yang berjalan di akun pengambilan memerlukan izin untuk mengambil data konfigurasi dari akun vendor. Anda memberikan izin agen dengan membuat peran AWS Identity and Access Management (IAM) di setiap akun vendor. AWS AppConfig Agen di akun pengambilan mengasumsikan peran ini untuk mendapatkan data dari akun vendor. Selesaikan prosedur di bagian ini untuk membuat kebijakan izin IAM, peran IAM, dan menambahkan penggantian agen ke manifes.

Sebelum Anda mulai

Kumpulkan informasi berikut sebelum Anda membuat kebijakan izin dan peran dalam IAM.

- ID untuk masing-masing Akun AWS. Akun pengambilan adalah akun yang akan memanggil akun lain untuk data konfigurasi. Akun vendor adalah akun yang akan menjual data konfigurasi ke akun pengambilan.

- Nama peran IAM yang digunakan oleh AWS AppConfig dalam akun pengambilan. Berikut daftar peran yang digunakan oleh AWS AppConfig, secara default:
  - Untuk Amazon Elastic Compute Cloud (Amazon EC2) AWS AppConfig , gunakan peran instans.
  - Untuk AWS Lambda, AWS AppConfig menggunakan peran eksekusi Lambda.
  - Untuk Amazon Elastic Container Service (Amazon ECS) Service Elastic Container Service (Amazon ECS) dan Amazon Elastic Kubernetes Service (Amazon EKS), gunakan peran container. AWS AppConfig

  Jika Anda mengonfigurasi AWS AppConfig Agen untuk menggunakan peran IAM yang berbeda dengan menentukan variabel `ROLE_ARN` lingkungan, catat nama itu.

Buat kebijakan izin

Gunakan prosedur berikut untuk membuat kebijakan izin menggunakan konsol IAM. Selesaikan prosedur di masing-masing Akun AWS yang akan menjual data konfigurasi untuk akun pengambilan.

Untuk membuat kebijakan IAM

1. Masuk ke AWS Management Console akun vendor.
2. Buka konsol IAM di https://console.aws.amazon.com/iam/.
3. Di panel navigasi, pilih Kebijakan dan kemudian pilih Buat kebijakan.
4. Pilih opsi JSON.
5. Di editor Kebijakan, ganti JSON default dengan pernyataan kebijakan berikut. Perbarui setiap *`placeholder sumber daya contoh`* dengan detail akun vendor.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "appconfig:StartConfigurationSession",
            "appconfig:GetLatestConfiguration"
        ],
        "Resource":
 "arn:partition:appconfig:region:vendor_account_ID:application/
vendor_application_ID/environment/vendor_environment_ID/
configuration/vendor_configuration_ID"
    }
   ]
```

```
}
```

Inilah contohnya:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "appconfig:StartConfigurationSession",
            "appconfig:GetLatestConfiguration"
        ],
        "Resource": "arn:aws:appconfig:us-east-2:111122223333:application/abc123/
environment/def456/configuration/hij789"
    }
   ]
}
```

6. Pilih Selanjutnya.
7. Di bidang Nama kebijakan, masukkan nama.
8. (Opsional) Untuk Menambahkan tag, tambahkan satu atau beberapa pasangan nilai kunci tag untuk mengatur, melacak, atau mengontrol akses kebijakan ini.
9. Pilih Buat kebijakan. Sistem mengembalikan Anda ke halaman Kebijakan.
10. Ulangi prosedur ini di masing-masing Akun AWS yang akan menjual data konfigurasi untuk akun pengambilan.

Buat peran IAM

Gunakan prosedur berikut untuk membuat peran IAM menggunakan konsol IAM. Selesaikan prosedur di masing-masing Akun AWS yang akan menjual data konfigurasi untuk akun pengambilan.

Untuk membuat IAM role

1. Masuk ke AWS Management Console akun vendor.
2. Buka konsol IAM di https://console.aws.amazon.com/iam/.
3. Di panel navigasi, pilih Peran, lalu pilih Buat kebijakan.
4. Untuk jenis entitas Tepercaya, pilih Akun AWS.
5. Di Akun AWSbagian ini, pilih Lain Akun AWS.

6. Di bidang ID Akun, masukkan ID akun pengambilan.
7. (Opsional) Sebagai praktik terbaik keamanan untuk peran asumsi ini, pilih Memerlukan ID eksternal dan masukkan string.
8. Pilih Selanjutnya.
9. Pada halaman Tambahkan izin, gunakan bidang Pencarian untuk menemukan kebijakan yang Anda buat di prosedur sebelumnya. Pilih kotak centang di sebelah namanya.
10. Pilih Selanjutnya.
11. Untuk nama Peran, masukkan nama.
12. (Opsional) Untuk Deskripsi, masukkan deskripsi.
13. Untuk Langkah 1: Pilih entitas tepercaya, pilih Edit. Ganti kebijakan kepercayaan JSON default dengan kebijakan berikut. Perbarui setiap *placeholder sumber daya contoh* dengan informasi dari akun pengambilan Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::retrieval_account_ID:role/appconfig_role_in_retrieval_account"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

14. (Opsional) Untuk Tag, tambahkan satu atau beberapa pasangan nilai tag-key untuk mengatur, melacak, atau mengontrol akses untuk peran ini.
15. Pilih Buat peran. Sistem mengembalikan Anda ke halaman Peran.
16. Cari peran yang baru saja Anda buat. Pilih itu. Di bagian ARN, salin ARN. Anda akan menentukan informasi ini dalam prosedur berikutnya.

Tambahkan penggantian kredensi ke manifes

Setelah Anda membuat peran IAM di akun vendor Anda, perbarui manifes di akun pengambilan. Secara khusus, tambahkan blok kredensyal dan ARN peran IAM untuk mengambil data konfigurasi dari akun vendor. Berikut adalah format JSON:

```
{
    "vendor_application_name:vendor_environment_name:vendor_configuration_name": {
        "credentials": {
            "roleArn":
 "arn:partition:iam::vendor_account_ID:role/name_of_role_created_in_vendor_account",
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
        }
    }
}
```

Inilah contohnya:

```
{
    "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
        "credentials": {
            "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
            "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
            "roleSessionName": "AwsAppConfigAgent",
            "credentialsDuration": "2h"
        }
    }
}
```

Validasi bahwa pengambilan multi-akun berfungsi

Anda dapat memvalidasi bahwa agen tersebut dapat mengambil data konfigurasi dari beberapa akun dengan meninjau log agen. AWS AppConfig Log `INFO` level untuk data awal yang diambil untuk '`YourApplicationNameYourEnvironmentName:YourConfigurationName`' adalah indikator terbaik untuk pengambilan yang berhasil. Jika pengambilan gagal, Anda akan melihat log `ERROR` level yang menunjukkan alasan kegagalan. Berikut adalah contoh untuk pengambilan yang berhasil dari akun vendor:

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
```

```
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MyTestApplication:MyTestEnvironment:MyDenyListConfiguration' in XX.Xms
```

Tulis salinan konfigurasi ke disk

Anda dapat mengonfigurasi AWS AppConfig Agen untuk secara otomatis menyimpan salinan konfigurasi ke disk dalam teks biasa. Fitur ini memungkinkan pelanggan dengan aplikasi yang membaca data konfigurasi dari disk untuk diintegrasikan AWS AppConfig.

Fitur ini tidak dirancang untuk digunakan sebagai fitur cadangan konfigurasi. AWS AppConfig Agen tidak membaca dari file konfigurasi yang disalin ke disk. Jika Anda ingin mencadangkan konfigurasi ke disk, lihat variabel `BACKUP_DIRECTORY` dan `PRELOAD_BACKUP` lingkungan untuk Menggunakan AWS AppConfig Agen dengan Amazon EC2 atau AWS AppConfig Menggunakan Agen dengan Amazon ECS dan Amazon EKS.

> **Warning**
>
> Perhatikan informasi penting berikut tentang fitur ini:
>
> - Konfigurasi yang disimpan ke disk disimpan dalam teks biasa dan dapat dibaca manusia. Jangan aktifkan fitur ini untuk konfigurasi yang menyertakan data sensitif.
> - Fitur ini menulis ke disk lokal. Gunakan prinsip hak istimewa terkecil untuk izin sistem file. Untuk informasi selengkapnya, lihat Terapkan akses hak akses paling rendah.

Untuk mengaktifkan konfigurasi tulis salin ke disk

1. Edit manifes.
2. Pilih konfigurasi yang ingin Anda tulis AWS AppConfig ke disk dan tambahkan `writeTo` elemen. Inilah contohnya:

```
{
    "application_name:environment_name:configuration_name": {
        "writeTo": {
            "path": "path_to_configuration_file"
        }
    }
}
```

Inilah contohnya:

```
{
    "MyTestApp:MyTestEnvironment:MyNewConfiguration": {
        "writeTo": {
            "path": "/tmp/aws-appconfig/mobile-app/beta/enable-mobile-payments"
        }
    }
}
```

3. Simpan perubahan Anda. File configuration.json akan diperbarui setiap kali data konfigurasi baru diterapkan.

Validasi bahwa menulis salinan konfigurasi ke disk berfungsi

Anda dapat memvalidasi bahwa salinan konfigurasi sedang ditulis ke disk dengan melihat dengan meninjau log AWS AppConfig agen. Entri `INFO` log dengan frasa “INFO menulis konfigurasi '*aplikasi*: *lingkungan*: *konfigurasi*' ke *file_path*" menunjukkan bahwa AWS AppConfig Agen menulis salinan konfigurasi ke disk.

Inilah contohnya:

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MobileApp:Beta:EnableMobilePayments' in XX.Xms
[appconfig agent] 2023/11/13 17:05:49 INFO wrote configuration
 'MobileApp:Beta:EnableMobilePayments' to /tmp/configs/your-app/your-env/your-
config.json
```

AWS AppConfig Agen pengembangan lokal

AWS AppConfig Agen mendukung mode pengembangan lokal. Jika Anda mengaktifkan mode pengembangan lokal, agen membaca data konfigurasi dari direktori tertentu pada disk. Itu tidak mengambil data konfigurasi dari AWS AppConfig. Anda dapat mensimulasikan penerapan konfigurasi dengan memperbarui file di direktori yang ditentukan. Kami merekomendasikan mode pengembangan lokal untuk kasus penggunaan berikut:

- Uji versi konfigurasi yang berbeda sebelum menerapkannya menggunakan AWS AppConfig.

- Uji opsi konfigurasi yang berbeda untuk fitur baru sebelum melakukan perubahan pada repositori kode Anda.
- Uji skenario konfigurasi yang berbeda untuk memverifikasi bahwa mereka berfungsi seperti yang diharapkan.

> **Warning**
>
> Jangan gunakan mode pengembangan lokal di lingkungan produksi. Mode ini tidak mendukung fitur AWS AppConfig keamanan penting seperti validasi penerapan dan rollback otomatis.

Gunakan prosedur berikut untuk mengkonfigurasi AWS AppConfig Agen untuk mode pengembangan lokal.

Untuk mengkonfigurasi AWS AppConfig Agen untuk mode pengembangan lokal

1. Instal agen menggunakan metode yang dijelaskan untuk lingkungan komputasi Anda. AWS AppConfig Agen bekerja dengan yang berikut Layanan AWS:
   - AWS Lambda
   - Amazon EC2
   - Amazon ECS dan Amazon EKS
2. Jika agen berjalan, hentikan.
3. Tambahkan `LOCAL_DEVELOPMENT_DIRECTORY` ke daftar variabel lingkungan. Tentukan direktori pada sistem file yang menyediakan agen dengan izin baca. Misalnya, `/tmp/local_configs`.
4. Buat file di direktori. Nama file harus menggunakan format berikut:

   ```
   application_name:environment_name:configuration_profile_name
   ```

   Inilah contohnya:

   ```
   Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
   ```

> **Note**
>
> (Opsional) Anda dapat mengontrol jenis konten yang dikembalikan agen untuk data konfigurasi Anda berdasarkan ekstensi yang Anda berikan pada file. Misalnya, jika Anda memberi nama file dengan ekstensi.json, agen mengembalikan jenis konten `application/json` saat aplikasi Anda memintanya. Jika Anda menghilangkan ekstensi, agen menggunakan `application/octet-stream` untuk jenis konten. Jika Anda memerlukan kontrol yang tepat, Anda dapat memberikan ekstensi dalam format `.type%subtype`. Agen akan mengembalikan jenis konten `type/subtype`.

5. Jalankan perintah berikut untuk me-restart agen dan meminta data konfigurasi.

```
curl http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name
```

Agen memeriksa perubahan pada file lokal pada interval polling yang ditentukan untuk agen. Jika interval polling tidak ditentukan, agen menggunakan interval default 45 detik. Pemeriksaan ini pada interval polling memastikan bahwa agen berperilaku sama di lingkungan pengembangan lokal seperti saat dikonfigurasi untuk berinteraksi dengan AWS AppConfig layanan.

> **Note**
>
> Untuk menyebarkan versi baru dari file konfigurasi pengembangan lokal, perbarui file dengan data baru.

Mengambil konfigurasi dengan langsung memanggil API

Aplikasi Anda mengambil data konfigurasi dengan terlebih dahulu membuat sesi konfigurasi menggunakan operasi StartConfigurationSession API. Klien sesi Anda kemudian membuat panggilan berkala ke GetLatestKonfigurasi untuk memeriksa dan mengambil data terbaru yang tersedia.

Saat meneleponStartConfigurationSession, kode Anda mengirimkan informasi berikut:

- Pengidentifikasi (ID atau nama) AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi yang dilacak sesi.

- (Opsional) Jumlah minimum waktu klien sesi harus menunggu di antara panggilan ke`GetLatestConfiguration`.

Sebagai tanggapan, AWS AppConfig berikan `InitialConfigurationToken` untuk diberikan kepada klien sesi dan digunakan saat pertama kali memanggil `GetLatestConfiguration` sesi itu.

> **Important**
>
> Token ini hanya boleh digunakan sekali dalam panggilan pertama Anda`GetLatestConfiguration`. Anda harus menggunakan token baru di `GetLatestConfiguration` response (`NextPollConfigurationToken`) di setiap panggilan berikutnya`GetLatestConfiguration`. Untuk mendukung kasus penggunaan polling yang panjang, token berlaku hingga 24 jam. Jika `GetLatestConfiguration` panggilan menggunakan token kedaluwarsa, sistem akan kembali`BadRequestException`.

Saat menelepon`GetLatestConfiguration`, kode klien Anda mengirimkan `ConfigurationToken` nilai terbaru yang dimilikinya dan diterima sebagai tanggapan:

- `NextPollConfigurationToken`: `ConfigurationToken` nilai yang akan digunakan pada panggilan berikutnya ke`GetLatestConfiguration`.
- `NextPollIntervalInSeconds`: durasi yang harus ditunggu klien sebelum melakukan panggilan berikutnya`GetLatestConfiguration`.
- Konfigurasi: data terbaru yang ditujukan untuk sesi tersebut. Ini mungkin kosong jika klien sudah memiliki versi konfigurasi terbaru.

> **Important**
>
> Perhatikan informasi penting berikut.
>
> - StartConfigurationSession API hanya boleh dipanggil sekali per aplikasi, lingkungan, profil konfigurasi, dan klien untuk membuat sesi dengan layanan. Ini biasanya dilakukan dalam startup aplikasi Anda atau segera sebelum pengambilan konfigurasi pertama.
> - Jika konfigurasi Anda digunakan menggunakan a`KmsKeyIdentifier`, permintaan Anda untuk menerima konfigurasi harus menyertakan izin untuk memanggil`kms:Decrypt`.

Untuk informasi selengkapnya, lihat Mendekripsi di Referensi AWS Key Management Service API.

- Operasi API yang sebelumnya digunakan untuk mengambil data konfigurasi`GetConfiguration`, tidak digunakan lagi. Operasi `GetConfiguration` API tidak mendukung konfigurasi terenkripsi.

Mengambil contoh konfigurasi

AWS CLI Contoh berikut menunjukkan cara mengambil data konfigurasi dengan menggunakan operasi AWS AppConfig Data `StartConfigurationSession` dan `GetLatestConfiguration` API. Perintah pertama memulai sesi konfigurasi. Panggilan ini mencakup ID (atau nama) AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi. API mengembalikan yang `InitialConfigurationToken` digunakan untuk mengambil data konfigurasi Anda.

```
aws appconfigdata start-configuration-session \
    --application-identifier application_name_or_ID \
    --environment-identifier environment_name_or_ID \
    --configuration-profile-identifier configuration_profile_name_or_ID
```

Sistem merespons dengan informasi dalam format berikut.

```
{
   "InitialConfigurationToken": initial configuration token
}
```

Setelah memulai sesi, gunakan InitialConfigurationToken untuk memanggil GetLatestKonfigurasi untuk mengambil data konfigurasi Anda. Data konfigurasi disimpan ke `mydata.json` file.

```
aws appconfigdata get-latest-configuration \
    --configuration-token initial configuration token mydata.json
```

Panggilan pertama untuk `GetLatestConfiguration` menggunakan yang `ConfigurationToken` diperoleh dari`StartConfigurationSession`. Informasi berikut dikembalikan.

```
{
    "NextPollConfigurationToken" : next configuration token,
    "ContentType" : content type of configuration,
```

```
    "NextPollIntervalInSeconds" : 60
}
```

Panggilan selanjutnya `GetLatestConfiguration` harus diberikan `NextPollConfigurationToken` dari respons sebelumnya.

```
aws appconfigdata get-latest-configuration \
    --configuration-token next configuration token mydata.json
```

Important

Perhatikan detail penting berikut tentang operasi `GetLatestConfiguration` API:

- `GetLatestConfiguration`Responsnya mencakup `Configuration` bagian yang menunjukkan data konfigurasi. `Configuration`Bagian ini hanya muncul jika sistem menemukan data konfigurasi baru atau yang diperbarui. Jika sistem tidak menemukan data konfigurasi baru atau yang diperbarui, maka `Configuration` datanya kosong.
- Anda menerima yang baru `ConfigurationToken` dalam setiap tanggapan dari`GetLatestConfiguration`.
- Sebaiknya atur frekuensi polling panggilan `GetLatestConfiguration` API Anda berdasarkan anggaran, frekuensi penerapan konfigurasi yang diharapkan, dan jumlah target untuk konfigurasi.

Memperluas alur kerja menggunakan ekstensi

Ekstensi menambah kemampuan Anda untuk menyuntikkan logika atau perilaku pada titik yang berbeda selama AWS AppConfig alur kerja membuat atau menerapkan konfigurasi. Misalnya, Anda dapat menggunakan ekstensi untuk melakukan jenis tugas berikut (untuk beberapa nama):

- Kirim pemberitahuan ke topik Amazon Simple Notification Service (Amazon SNS) saat profil konfigurasi digunakan.
- Gosok konten profil konfigurasi untuk data sensitif sebelum penerapan dimulai.
- Buat atau perbarui masalah Atlassian Jira setiap kali perubahan dilakukan pada bendera fitur.
- Gabungkan konten dari layanan atau sumber data ke dalam data konfigurasi saat memulai penerapan.
- Cadangkan konfigurasi ke bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) setiap kali konfigurasi diterapkan.

Anda dapat mengaitkan jenis tugas ini dengan AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi.

Daftar Isi

Tentang AWS AppConfig ekstensi

Topik ini memperkenalkan konsep dan terminologi AWS AppConfig ekstensi. Informasi dibahas dalam konteks setiap langkah yang diperlukan untuk mengatur dan menggunakan AWS AppConfig ekstensi.

Topik

- Langkah 3: Buat asosiasi ekstensi
- Langkah 4: Menyebarkan konfigurasi dan memverifikasi tindakan ekstensi dilakukan

Langkah 1: Tentukan apa yang ingin Anda lakukan dengan ekstensi

Apakah Anda ingin menerima pemberitahuan ke webhook yang mengirim pesan ke Slack kapan saja AWS AppConfig penerapan selesai? Apakah Anda ingin mencadangkan profil konfigurasi ke bucket Amazon Simple Storage Service (Amazon S3) sebelum konfigurasi diterapkan? Apakah Anda ingin menggosok data konfigurasi untuk informasi sensitif sebelum konfigurasi diterapkan? Anda dapat menggunakan ekstensi untuk melakukan jenis tugas ini dan banyak lagi. Anda dapat membuat ekstensi khusus atau menggunakan ekstensi yang AWS ditulis yang disertakan. AWS AppConfig

> **Note**
>
> Untuk sebagian besar kasus penggunaan, untuk membuat ekstensi khusus, Anda harus membuat AWS Lambda fungsi untuk melakukan perhitungan dan pemrosesan apa pun yang ditentukan dalam ekstensi. Untuk informasi selengkapnya, lihat Walkthrough: Membuat ekstensi khusus AWS AppConfig.

Ekstensi yang AWS ditulis berikut dapat membantu Anda mengintegrasikan penerapan konfigurasi dengan cepat dengan layanan lain. Anda dapat menggunakan ekstensi ini di AWS AppConfig konsol atau dengan memanggil tindakan API ekstensi langsung dari AWS CLI, AWS Tools for PowerShell, atau SDK.

Ekstensi	Deskripsi
Amazon CloudWatch Ternyata Pengujian A/B	Ekstensi ini memungkinkan aplikasi Anda untuk menetapkan variasi ke sesi pengguna secara lokal, bukan dengan memanggil operasi. EvaluateFeature Untuk informasi selengkap nya, lihat Bekerja dengan ekstensi Amazon CloudWatch Evidently.
AWS AppConfig peristiwa penyebaran ke EventBridge	Ekstensi ini mengirimkan peristiwa ke bus acara EventBridge default saat konfigurasi diterapkan.

Ekstensi	Deskripsi
AWS AppConfig peristiwa penyebaran ke Amazon Simple Notification Service (Amazon SNS)	Ekstensi ini mengirimkan pesan ke topik Amazon SNS yang Anda tentukan saat konfigurasi diterapkan.
AWS AppConfig peristiwa penyebaran ke Layanan Antrian Sederhana Amazon (Amazon Simple Service SQS)	Ekstensi ini memasukkan pesan ke antrean Amazon SQS Anda saat konfigurasi diterapkan.
Ekstensi Integrasi—Atlassian Jira	Ekstensi ini memungkinkan AWS AppConfig untuk membuat dan memperbarui masalah setiap kali Anda membuat perubahan pada bendera fitur.

Langkah 2: Tentukan kapan Anda ingin ekstensi berjalan

Ekstensi mendefinisikan satu atau beberapa tindakan yang dilakukannya selama AWS AppConfig alur kerja. Misalnya, `AWS AppConfig deployment events to Amazon SNS` ekstensi yang AWS ditulis menyertakan tindakan untuk mengirim pemberitahuan ke topik Amazon SNS. Setiap tindakan dipanggil baik ketika Anda berinteraksi dengan AWS AppConfig atau ketika AWS AppConfig melakukan proses atas nama Anda. Ini disebut titik tindakan. AWS AppConfig ekstensi mendukung poin tindakan berikut:

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`
- `ON_DEPLOYMENT_START`
- `ON_DEPLOYMENT_STEP`
- `ON_DEPLOYMENT_BAKING`
- `ON_DEPLOYMENT_COMPLETE`
- `ON_DEPLOYMENT_ROLLED_BACK`

Tindakan ekstensi yang dikonfigurasi pada titik `PRE_*` tindakan diterapkan setelah validasi permintaan, tetapi sebelum AWS AppConfig melakukan aktivitas yang sesuai dengan nama titik tindakan. Pemanggilan tindakan ini diproses pada saat yang sama sebagai permintaan. Jika lebih

dari satu permintaan dibuat, pemanggilan tindakan berjalan secara berurutan. Perhatikan juga bahwa titik `PRE_*` tindakan menerima dan dapat mengubah konten konfigurasi. `PRE_*`titik tindakan juga dapat merespons kesalahan dan mencegah tindakan terjadi.

Ekstensi juga dapat berjalan secara paralel dengan AWS AppConfig alur kerja dengan menggunakan titik `ON_*` tindakan. `ON_*`titik tindakan dipanggil secara asinkron. `ON_*`titik tindakan tidak menerima isi konfigurasi. Jika ekstensi mengalami kesalahan selama titik `ON_*` tindakan, layanan mengabaikan kesalahan dan melanjutkan alur kerja.

Langkah 3: Buat asosiasi ekstensi

Untuk membuat ekstensi, atau mengonfigurasi ekstensi yang AWS ditulis, Anda menentukan titik tindakan yang memanggil ekstensi saat AWS AppConfig sumber daya tertentu digunakan. Misalnya, Anda dapat memilih untuk menjalankan `AWS AppConfig deployment events to Amazon SNS` ekstensi dan menerima pemberitahuan tentang topik Amazon SNS kapan saja penerapan konfigurasi dimulai untuk aplikasi tertentu. Mendefinisikan titik tindakan mana yang memanggil ekstensi untuk AWS AppConfig sumber daya tertentu disebut asosiasi ekstensi. Asosiasi ekstensi adalah hubungan tertentu antara ekstensi dan AWS AppConfig sumber daya, seperti aplikasi atau profil konfigurasi.

Satu AWS AppConfig aplikasi dapat mencakup beberapa lingkungan dan profil konfigurasi. Jika Anda mengaitkan ekstensi ke aplikasi atau lingkungan, AWS AppConfig memanggil ekstensi untuk alur kerja apa pun yang terkait dengan sumber daya aplikasi atau lingkungan, jika berlaku.

Misalnya, Anda memiliki AWS AppConfig aplikasi bernama MobileApps yang menyertakan profil konfigurasi yang disebut AccessList. Dan katakanlah MobileApps aplikasi tersebut mencakup lingkungan Beta, Integrasi, dan Produksi. Anda membuat asosiasi ekstensi untuk ekstensi notifikasi Amazon SNS yang AWS ditulis dan mengaitkan ekstensi ke aplikasi. MobileApps Ekstensi notifikasi Amazon SNS dipanggil kapan saja konfigurasi diterapkan untuk aplikasi ke salah satu dari tiga lingkungan.

> **Note**
>
> Anda tidak perlu membuat ekstensi untuk menggunakan ekstensi yang AWS ditulis, tetapi Anda harus membuat asosiasi ekstensi.

Langkah 4: Menyebarkan konfigurasi dan memverifikasi tindakan ekstensi dilakukan

Setelah Anda membuat asosiasi, ketika konfigurasi yang dihosting dibuat atau konfigurasi diterapkan, AWS AppConfig memanggil ekstensi dan melakukan tindakan yang ditentukan. Ketika ekstensi dipanggil, jika sistem mengalami kesalahan selama titik `PRE-*` tindakan, AWS AppConfig mengembalikan informasi tentang kesalahan itu.

Bekerja dengan ekstensi yang AWS ditulis

AWS AppConfig termasuk ekstensi yang AWS ditulis berikut. Ekstensi ini dapat membantu Anda mengintegrasikan AWS AppConfig alur kerja dengan layanan lain. Anda dapat menggunakan ekstensi ini di AWS Management Console atau dengan memanggil tindakan API ekstensi langsung dari AWS CLI, AWS Tools for PowerShell, atau SDK.

Ekstensi	Deskripsi
Amazon CloudWatch Ternyata Pengujian A/B	Ekstensi ini memungkinkan aplikasi Anda untuk menetapkan variasi ke sesi pengguna secara lokal, bukan dengan memanggil operasi. EvaluateFeature Untuk informasi selengkap nya, lihat Bekerja dengan ekstensi Amazon CloudWatch Evidently.
AWS AppConfig peristiwa penyebaran ke EventBridge	Ekstensi ini mengirimkan peristiwa ke bus acara EventBridge default saat konfigurasi diterapkan.
AWS AppConfig peristiwa penyebaran ke Amazon Simple Notification Service (Amazon SNS)	Ekstensi ini mengirimkan pesan ke topik Amazon SNS yang Anda tentukan saat konfigurasi diterapkan.
AWS AppConfig peristiwa penyebaran ke Layanan Antrian Sederhana Amazon (Amazon Simple Service SQS)	Ekstensi ini memasukkan pesan ke antrean Amazon SQS Anda saat konfigurasi diterapkan.
Ekstensi Integrasi—Atlassian Jira	Ekstensi ini memungkinkan AWS AppConfig untuk membuat dan memperbarui masalah

Ekstensi	Deskripsi
	setiap kali Anda membuat perubahan pada bendera fitur.

Bekerja dengan ekstensi Amazon CloudWatch Evidently

Anda dapat menggunakan Amazon CloudWatch Terbukti untuk memvalidasi fitur baru dengan aman dengan menyajikannya ke persentase tertentu dari pengguna Anda saat Anda meluncurkan fitur tersebut. Anda dapat memantau performa fitur baru untuk membantu Anda memutuskan kapan harus menaikkan lalu lintas ke para pengguna Anda. Hal ini akan membantu Anda mengurangi risiko dan mengidentifikasi konsekuensi yang tidak Anda inginkan sebelum Anda meluncurkan fitur tersebut sepenuhnya. Anda juga dapat melakukan percobaan-percobaan A/B untuk membuat keputusan desain fitur berdasarkan bukti dan data.

AWS AppConfig Ekstensi untuk CloudWatch Evidently memungkinkan aplikasi Anda untuk menetapkan variasi ke sesi pengguna secara lokal, bukan dengan memanggil operasi. EvaluateFeature Sesi lokal mengurangi risiko latensi dan ketersediaan yang menyertai panggilan API. Untuk informasi tentang cara mengonfigurasi dan menggunakan ekstensi, lihat Melakukan peluncuran dan eksperimen A/B dengan CloudWatch Evidently in the Amazon CloudWatch User Guide.

Bekerja dengan **`AWS AppConfig deployment events to Amazon EventBridge`** ekstensi

`AWS AppConfig deployment events to Amazon EventBridge`Ekstensi adalah ekstensi yang AWS ditulis yang membantu Anda memantau dan bertindak pada alur kerja penerapan AWS AppConfig konfigurasi. Ekstensi mengirimkan pemberitahuan acara ke bus peristiwa EventBridge default setiap kali konfigurasi diterapkan. Setelah Anda mengaitkan ekstensi ke salah satu AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi, AWS AppConfig kirimkan pemberitahuan peristiwa ke bus acara setelah setiap penerapan konfigurasi dimulai, berakhir, dan rollback.

Jika ingin lebih mengontrol titik tindakan mana yang mengirim EventBridge notifikasi, Anda dapat membuat ekstensi khusus dan memasukkan bus peristiwa EventBridge default Amazon Resource Name (ARN) untuk bidang URI. Untuk informasi tentang membuat ekstensi, lihatWalkthrough: Membuat ekstensi khusus AWS AppConfig.

> **Important**
>
> Ekstensi ini hanya mendukung bus peristiwa EventBridge default.

Menggunakan ekstensi

Untuk menggunakan `AWS AppConfig deployment events to Amazon EventBridge` ekstensi, Anda terlebih dahulu melampirkan ekstensi ke salah satu AWS AppConfig sumber daya Anda dengan membuat asosiasi ekstensi. Anda membuat asosiasi menggunakan AWS AppConfig konsol atau aksi CreateExtensionAssociationAPI. Saat Anda membuat asosiasi, Anda menentukan ARN AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi. Jika Anda mengaitkan ekstensi ke aplikasi atau lingkungan, pemberitahuan peristiwa dikirim untuk profil konfigurasi apa pun yang terdapat dalam aplikasi atau lingkungan yang ditentukan.

Setelah Anda membuat asosiasi, ketika konfigurasi untuk AWS AppConfig sumber daya yang ditentukan diterapkan, AWS AppConfig memanggil ekstensi dan mengirim pemberitahuan sesuai dengan titik tindakan yang ditentukan dalam ekstensi.

> **Note**
>
> Ekstensi ini dipanggil oleh poin tindakan berikut:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Anda tidak dapat menyesuaikan titik tindakan untuk ekstensi ini. Untuk memanggil titik tindakan yang berbeda, Anda dapat membuat ekstensi Anda sendiri. Untuk informasi selengkapnya, lihat Walkthrough: Membuat ekstensi khusus AWS AppConfig.

Gunakan prosedur berikut untuk membuat asosiasi AWS AppConfig ekstensi dengan menggunakan AWS Systems Manager konsol atau AWS CLI.

Untuk membuat asosiasi ekstensi (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih AWS AppConfig.
3. Pada tab Ekstensi, pilih Tambahkan ke sumber daya.
4. Di bagian Detail sumber daya ekstensi, untuk jenis sumber daya, pilih jenis AWS AppConfig sumber daya. Bergantung pada sumber daya yang Anda pilih, AWS AppConfig meminta Anda untuk memilih sumber daya lain.
5. Pilih Buat asosiasi ke sumber daya.

Berikut adalah contoh peristiwa yang dikirim ke EventBridge saat ekstensi dipanggil.

```
{
   "version":"0",
   "id":"c53dbd72-c1a0-2302-9ed6-c076e9128277",
   "detail-type":"On Deployment Complete",
   "source":"aws.appconfig",
   "account":"111122223333",
   "time":"2022-07-09T01:44:15Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:appconfig:us-east-1:111122223333:extensionassociation/z763ff5"
   ],
   "detail":{
      "InvocationId":"5tfjcig",
       "Parameters":{

      },
      "Type":"OnDeploymentComplete",
      "Application":{
         "Id":"ba8toh7",
         "Name":"MyApp"
      },
      "Environment":{
         "Id":"pgil2o7",
         "Name":"MyEnv"
      },
      "ConfigurationProfile":{
         "Id":"ga3tqep",
```

```
            "Name":"MyConfigProfile"
        },
        "DeploymentNumber":1,
        "ConfigurationVersion":"1"
    }
}
```

Bekerja dengan **AWS AppConfig deployment events to Amazon SNS** ekstensi

`AWS AppConfig deployment events to Amazon SNS`Ekstensi adalah ekstensi yang AWS ditulis yang membantu Anda memantau dan bertindak pada alur kerja penerapan AWS AppConfig konfigurasi. Ekstensi menerbitkan pesan ke topik Amazon SNS setiap kali konfigurasi diterapkan. Setelah Anda mengaitkan ekstensi ke salah satu AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi, AWS AppConfig memublikasikan pesan ke topik setelah setiap penerapan konfigurasi dimulai, diakhiri, dan rollback.

Jika ingin lebih mengontrol titik tindakan mana yang mengirim notifikasi Amazon SNS, Anda dapat membuat ekstensi khusus dan memasukkan topik Amazon SNS Nama Sumber Daya Amazon (ARN) untuk bidang URI. Untuk informasi tentang membuat ekstensi, lihatWalkthrough: Membuat ekstensi khusus AWS AppConfig.

Menggunakan ekstensi

Bagian ini menjelaskan cara menggunakan `AWS AppConfig deployment events to Amazon SNS` ekstensi.

Langkah 1: Konfigurasikan AWS AppConfig untuk mempublikasikan pesan ke topik

Tambahkan kebijakan kontrol akses ke topik Amazon SNS Anda yang memberikan () izin publikasi AWS AppConfig (`appconfig.amazonaws.com`). `sns:Publish` Untuk informasi selengkapnya, lihat Contoh kasus untuk kontrol akses Amazon SNS.

Langkah 2: Buat asosiasi ekstensi

Lampirkan ekstensi ke salah satu AWS AppConfig sumber daya Anda dengan membuat asosiasi ekstensi. Anda membuat asosiasi menggunakan AWS AppConfig konsol atau aksi CreateExtensionAssociationAPI. Saat Anda membuat asosiasi, Anda menentukan ARN AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi. Jika Anda mengaitkan ekstensi ke aplikasi atau lingkungan, pemberitahuan dikirim untuk profil konfigurasi apa pun yang terdapat dalam aplikasi

atau lingkungan yang ditentukan. Saat Anda membuat asosiasi, Anda harus memasukkan nilai untuk `topicArn` parameter yang berisi ARN topik Amazon SNS yang ingin Anda gunakan.

Setelah Anda membuat asosiasi, ketika konfigurasi untuk AWS AppConfig sumber daya yang ditentukan diterapkan, AWS AppConfig memanggil ekstensi dan mengirim pemberitahuan sesuai dengan titik tindakan yang ditentukan dalam ekstensi.

> **Note**
>
> Ekstensi ini dipanggil oleh poin tindakan berikut:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Anda tidak dapat menyesuaikan titik tindakan untuk ekstensi ini. Untuk memanggil titik tindakan yang berbeda, Anda dapat membuat ekstensi Anda sendiri. Untuk informasi selengkapnya, lihat Walkthrough: Membuat ekstensi khusus AWS AppConfig.

Gunakan prosedur berikut untuk membuat asosiasi AWS AppConfig ekstensi dengan menggunakan AWS Systems Manager konsol atau AWS CLI.

Untuk membuat asosiasi ekstensi (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih AWS AppConfig.
3. Pada tab Ekstensi, pilih Tambahkan ke sumber daya.
4. Di bagian Detail sumber daya ekstensi, untuk jenis sumber daya, pilih jenis AWS AppConfig sumber daya. Bergantung pada sumber daya yang Anda pilih, AWS AppConfig meminta Anda untuk memilih sumber daya lain.
5. Pilih Buat asosiasi ke sumber daya.

Berikut adalah contoh pesan yang dikirim ke topik Amazon SNS saat ekstensi dipanggil.

```
{
```

```
    "Type": "Notification",
    "MessageId": "ae9d702f-9a66-51b3-8586-2b17932a9f28",
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic",
    "Message": {
        "InvocationId": "7itcaxp",
        "Parameters": {
            "topicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic"
        },
        "Application": {
            "Id": "1a2b3c4d",
            "Name": MyApp
        },
        "Environment": {
            "Id": "1a2b3c4d",
            "Name": MyEnv
        },
        "ConfigurationProfile": {
            "Id": "1a2b3c4d",
            "Name": "MyConfigProfile"
        },
        "Description": null,
        "DeploymentNumber": "3",
        "ConfigurationVersion": "1",
        "Type": "OnDeploymentComplete"
    },
    "Timestamp": "2022-06-30T20:26:52.067Z",
    "SignatureVersion": "1",
    "Signature": "<...>",
    "SigningCertURL": "<...>",
    "UnsubscribeURL": "<...>",
    "MessageAttributes": {
        "MessageType": {
            "Type": "String",
            "Value": "OnDeploymentStart"
        }
    }
}
```

Bekerja dengan **`AWS AppConfig deployment events to Amazon SQS`** ekstensi

`AWS AppConfig deployment events to Amazon SQS`Ekstensi adalah ekstensi yang AWS ditulis yang membantu Anda memantau dan bertindak pada alur kerja penerapan AWS AppConfig konfigurasi. Ekstensi mengantrekan pesan ke antrean Amazon Simple Queue Service (Amazon SQS) Anda setiap kali konfigurasi diterapkan. Setelah Anda mengaitkan ekstensi ke salah satu AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi, masukkan pesan ke dalam AWS AppConfig antrian setelah setiap penerapan konfigurasi dimulai, diakhiri, dan rollback.

Jika ingin lebih mengontrol titik tindakan mana yang mengirim notifikasi Amazon SQS, Anda dapat membuat ekstensi khusus dan memasukkan antrean Amazon SQS Nama Sumber Daya Amazon (ARN) untuk bidang URI. Untuk informasi tentang membuat ekstensi, lihatWalkthrough: Membuat ekstensi khusus AWS AppConfig.

Menggunakan ekstensi

Bagian ini menjelaskan cara menggunakan `AWS AppConfig deployment events to Amazon SQS` ekstensi.

Langkah 1: Konfigurasikan AWS AppConfig untuk mengantrekan pesan

Tambahkan kebijakan Amazon SQS ke antrean Amazon SQS Anda yang memberikan () AWS AppConfig izin kirim pesan `appconfig.amazonaws.com` (). `sqs:SendMessage` Untuk informasi selengkapnya, lihat Contoh dasar kebijakan Amazon SQS.

Langkah 2: Buat asosiasi ekstensi

Lampirkan ekstensi ke salah satu AWS AppConfig sumber daya Anda dengan membuat asosiasi ekstensi. Anda membuat asosiasi menggunakan AWS AppConfig konsol atau aksi CreateExtensionAssociationAPI. Saat Anda membuat asosiasi, Anda menentukan ARN AWS AppConfig aplikasi, lingkungan, atau profil konfigurasi. Jika Anda mengaitkan ekstensi ke aplikasi atau lingkungan, pemberitahuan dikirim untuk profil konfigurasi apa pun yang terdapat dalam aplikasi atau lingkungan yang ditentukan. Saat membuat asosiasi, Anda harus memasukkan `Here` parameter yang berisi ARN antrian Amazon SQS yang ingin Anda gunakan.

Setelah Anda membuat asosiasi, ketika konfigurasi untuk AWS AppConfig sumber daya tertentu dibuat atau diterapkan, AWS AppConfig memanggil ekstensi dan mengirimkan pemberitahuan sesuai dengan titik tindakan yang ditentukan dalam ekstensi.

> **Note**
>
> Ekstensi ini dipanggil oleh poin tindakan berikut:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Anda tidak dapat menyesuaikan titik tindakan untuk ekstensi ini. Untuk memanggil titik tindakan yang berbeda, Anda dapat membuat ekstensi Anda sendiri. Untuk informasi selengkapnya, lihat Walkthrough: Membuat ekstensi khusus AWS AppConfig.

Gunakan prosedur berikut untuk membuat asosiasi AWS AppConfig ekstensi dengan menggunakan AWS Systems Manager konsol atau AWS CLI.

Untuk membuat asosiasi ekstensi (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih AWS AppConfig.
3. Pada tab Ekstensi, pilih Tambahkan ke sumber daya.
4. Di bagian Detail sumber daya ekstensi, untuk jenis sumber daya, pilih jenis AWS AppConfig sumber daya. Bergantung pada sumber daya yang Anda pilih, AWS AppConfig meminta Anda untuk memilih sumber daya lain.
5. Pilih Buat asosiasi ke sumber daya.

Berikut adalah contoh pesan yang dikirim ke antrean Amazon SQS saat ekstensi dipanggil.

```
{
   "InvocationId":"7itcaxp",
   "Parameters":{
      "queueArn":"arn:aws:sqs:us-east-1:111122223333:MySQSQueue"
   },
   "Application":{
      "Id":"1a2b3c4d",
      "Name":MyApp
```

```
   },
   "Environment":{
      "Id":"1a2b3c4d",
      "Name":MyEnv
   },
   "ConfigurationProfile":{
      "Id":"1a2b3c4d",
      "Name":"MyConfigProfile"
   },
   "Description":null,
   "DeploymentNumber":"3",
   "ConfigurationVersion":"1",
   "Type":"OnDeploymentComplete"
}
```

Bekerja dengan ekstensi Atlassian Jira untuk AWS AppConfig

Dengan mengintegrasikan dengan Atlassian Jira, AWS AppConfig dapat membuat dan memperbarui masalah di konsol Atlassian setiap kali Anda membuat perubahan pada tanda fitur di Anda untuk yang ditentukan. Akun AWS Wilayah AWS Setiap masalah Jira mencakup nama bendera, ID aplikasi, ID profil konfigurasi, dan nilai bendera. Setelah Anda memperbarui, menyimpan, dan menerapkan perubahan bendera Anda, Jira memperbarui masalah yang ada dengan detail perubahan.

> **Note**
>
> Jira memperbarui masalah setiap kali Anda membuat atau memperbarui bendera fitur. Jira juga memperbarui masalah saat Anda menghapus atribut flag tingkat anak dari bendera tingkat induk. Jira tidak merekam informasi saat Anda menghapus bendera tingkat orang tua.

Untuk mengonfigurasi integrasi, Anda harus melakukan hal berikut:

- Mengkonfigurasi izin untuk AWS AppConfig integrasi Jira
- Mengkonfigurasi aplikasi integrasi AWS AppConfig Jira

Mengkonfigurasi izin untuk AWS AppConfig integrasi Jira

Saat Anda mengonfigurasi AWS AppConfig integrasi dengan Jira, Anda menentukan kredensional untuk pengguna. Secara khusus, Anda memasukkan ID kunci akses pengguna dan kunci rahasia

di aplikasi AWS AppConfig for Jira. Pengguna ini memberikan izin kepada Jira untuk berkomunikasi AWS AppConfig. AWS AppConfig menggunakan kredensi ini satu kali untuk membangun hubungan antara AWS AppConfig dan Jira. Kredensialnya tidak disimpan. Anda dapat menghapus asosiasi dengan mencopot pemasangan aplikasi AWS AppConfig for Jira.

Akun pengguna memerlukan kebijakan izin yang mencakup tindakan berikut:

- `appconfig:CreateExtensionAssociation`
- `appconfig:GetConfigurationProfile`
- `appconfig:ListApplications`
- `appconfig:ListConfigurationProfiles`
- `appconfig:ListExtensionAssociations`
- `sts:GetCallerIdentity`

Selesaikan tugas-tugas berikut untuk membuat kebijakan izin IAM dan pengguna untuk AWS AppConfig dan integrasi Jira:

Tugas

- Tugas 1: Buat kebijakan izin IAM untuk AWS AppConfig dan integrasi Jira
- Tugas 2: Buat pengguna untuk AWS AppConfig dan integrasi Jira

Tugas 1: Buat kebijakan izin IAM untuk AWS AppConfig dan integrasi Jira

Gunakan prosedur berikut untuk membuat kebijakan izin IAM yang memungkinkan Atlassian Jira untuk berkomunikasi. AWS AppConfig Kami menyarankan Anda membuat kebijakan baru dan melampirkan kebijakan ini ke peran IAM baru. Menambahkan izin yang diperlukan ke kebijakan dan peran IAM yang ada bertentangan dengan prinsip hak istimewa paling sedikit dan tidak disarankan.

Untuk membuat kebijakan IAM untuk AWS AppConfig dan integrasi Jira

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan dan kemudian pilih Buat kebijakan.
3. Pada halaman Buat kebijakan, pilih tab JSON dan ganti konten default dengan kebijakan berikut. Dalam kebijakan berikut, ganti *`Region`*, *`Account_ID`*, *`Application_ID`*, dan *`Configuration_PROFILE_ID`* dengan informasi dari lingkungan flag fitur Anda. AWS AppConfig

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "appconfig:CreateExtensionAssociation",
                      "appconfig:ListExtensionAssociations",
                      "appconfig:GetConfigurationProfile"
             ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID",

 "arn:aws:appconfig:Region:account_ID:application/application_ID/
configurationprofile/configuration_profile_ID"
             ]
        },
       {
            "Effect": "Allow",
            "Action": [
                      "appconfig:ListApplications"

             ],
            "Resource": [
                      "arn:aws:appconfig:Region:account_ID:*"
             ]
        },
         {
            "Effect": "Allow",
            "Action": [
                      "appconfig:ListConfigurationProfiles"
             ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID"
             ]
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
            "Resource": "*"
        }
```

```
    ]
}
```

4. Pilih Berikutnya: Tanda.
5. (Opsional) Tambahkan satu atau beberapa pasangan nilai kunci tag untuk mengatur, melacak, atau mengontrol akses kebijakan ini, lalu pilih Berikutnya: Tinjau.
6. Pada halaman Kebijakan ulasan, masukkan nama di kotak Nama, seperti**`AppConfigJiraPolicy`**, lalu masukkan deskripsi opsional.
7. Pilih Buat kebijakan.

Tugas 2: Buat pengguna untuk AWS AppConfig dan integrasi Jira

Gunakan prosedur berikut untuk membuat pengguna untuk AWS AppConfig dan integrasi Atlassian Jira. Setelah Anda membuat pengguna, Anda dapat menyalin ID kunci akses dan kunci rahasia, yang akan Anda tentukan saat Anda menyelesaikan integrasi.

Untuk membuat pengguna untuk AWS AppConfig dan integrasi Jira

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Pengguna, lalu pilih Tambah pengguna.
3. Di bidang Nama pengguna, masukkan nama, seperti**`AppConfigJiraUser`**.
4. Untuk Pilih jenis AWS kredensi, pilih Kunci akses - Akses terprogram.
5. Pilih Selanjutnya: Izin.
6. Di bawah halaman Setel izin, pilih Lampirkan kebijakan yang ada secara langsung. Cari dan pilih kotak centang untuk kebijakan yang Anda buatTugas 1: Buat kebijakan izin IAM untuk AWS AppConfig dan integrasi Jira, lalu pilih Berikutnya: Tag.
7. Pada halaman Tambahkan tag (opsional), tambahkan satu atau beberapa pasangan nilai kunci tag untuk mengatur, melacak, atau mengontrol akses bagi pengguna ini. Pilih Berikutnya: Peninjauan.
8. Pada halaman Ulasan, verifikasi detail pengguna.
9. Pilih Create user (Buat pengguna). Sistem menampilkan ID kunci akses pengguna dan kunci rahasia. Unduh file.csv atau salin kredensialnya ke lokasi terpisah. Anda akan menentukan kredensi ini ketika Anda mengonfigurasi integrasi.

Mengkonfigurasi aplikasi integrasi AWS AppConfig Jira

Gunakan prosedur berikut untuk mengonfigurasi opsi yang diperlukan dalam aplikasi AWS AppConfig for Jira. Setelah Anda menyelesaikan prosedur ini, Jira membuat masalah baru untuk setiap tanda fitur di Anda Akun AWS untuk yang ditentukan Wilayah AWS. Jika Anda membuat perubahan pada tanda fitur di AWS AppConfig, Jira mencatat detail dalam masalah yang ada.

> **Note**
>
> Bendera AWS AppConfig fitur dapat menyertakan beberapa atribut bendera tingkat anak. Jira membuat satu masalah untuk setiap flag fitur tingkat induk. Jika Anda mengubah atribut flag tingkat anak, Anda dapat melihat detail perubahan tersebut dalam masalah JIRA untuk flag tingkat induk.

Untuk mengkonfigurasi integrasi

1. Masuk ke Marketplace Atlassian.
2. Ketik **`AWS AppConfig`** di bidang pencarian dan tekan Enter.
3. Instal aplikasi pada instans Jira Anda.
4. Di konsol Atlassian, pilih Kelola aplikasi, lalu pilih AWS AppConfig untuk Jira.
5. Pilih Konfigurasikan
6. Di bawah Detail konfigurasi, pilih proyek Jira dan kemudian pilih proyek yang ingin Anda kaitkan dengan bendera AWS AppConfig fitur Anda.
7. Pilih Wilayah AWS, lalu pilih Wilayah tempat bendera AWS AppConfig fitur Anda berada.
8. Di bidang ID Aplikasi, masukkan nama AWS AppConfig aplikasi yang berisi bendera fitur Anda.
9. Di bidang Configuration profile ID, masukkan nama profil AWS AppConfig konfigurasi untuk flag fitur Anda.
10. Di kolom Access key ID dan Secret key, masukkan kredenal yang Anda salin. Tugas 2: Buat pengguna untuk AWS AppConfig dan integrasi Jira Secara opsional, Anda juga dapat menentukan token sesi.
11. Pilih Kirim.
12. Di konsol Atlassian, pilih Proyek, lalu pilih proyek yang Anda pilih untuk AWS AppConfig diintegrasikan. Halaman Masalah menampilkan masalah untuk setiap tanda fitur di yang ditentukan Akun AWS dan Wilayah AWS.

Menghapus aplikasi dan AWS AppConfig data untuk Jira

Jika Anda tidak lagi ingin menggunakan integrasi Jira dengan flag AWS AppConfig fitur, Anda dapat menghapus aplikasi AWS AppConfig for Jira di konsol Atlassian. Menghapus aplikasi integrasi melakukan hal berikut:

- Menghapus hubungan antara instans Jira Anda dan AWS AppConfig
- Menghapus detail instans Jira Anda dari AWS AppConfig

Untuk menghapus aplikasi AWS AppConfig untuk Jira

1. Di konsol Atlassian, pilih Kelola aplikasi.
2. Pilih AWS AppConfig untuk Jira.
3. Pilih Hapus Instalasi.

Walkthrough: Membuat ekstensi khusus AWS AppConfig

Untuk membuat AWS AppConfig ekstensi khusus, selesaikan tugas-tugas berikut. Setiap tugas dijelaskan secara lebih rinci dalam topik selanjutnya.

> **Note**
>
> Anda dapat melihat contoh AWS AppConfig ekstensi kustom pada GitHub:
>
> - Contoh ekstensi yang mencegah penerapan dengan kalender `blocked day` moratorium menggunakan Systems Manager Change Calendar
> - Ekstensi sampel yang mencegah rahasia bocor ke data konfigurasi menggunakan git-secret
> - Ekstensi sampel yang mencegah informasi identitas pribadi (PII) bocor ke data konfigurasi menggunakan Amazon Comprehend

1. Buat AWS Lambda fungsi

   Untuk sebagian besar kasus penggunaan, untuk membuat ekstensi khusus, Anda harus membuat AWS Lambda fungsi untuk melakukan perhitungan dan pemrosesan apa pun yang ditentukan dalam ekstensi. Pengecualian untuk aturan ini adalah jika Anda membuat versi kustom dari

ekstensi notifikasi yang AWS ditulis untuk menambah atau menghapus titik tindakan. Untuk detail selengkapnya tentang pengecualian ini, lihatMembuat AWS AppConfig ekstensi khusus.

2. Konfigurasikan izin untuk ekstensi kustom Anda

   Untuk mengonfigurasi izin untuk ekstensi kustom Anda, Anda dapat melakukan salah satu hal berikut:

   - Buat peran layanan AWS Identity and Access Management (IAM) yang menyertakan `InvokeFunction` izin.
   - Buat kebijakan sumber daya dengan menggunakan tindakan AddPermissionAPI Lambda.

   Panduan ini menjelaskan cara membuat peran layanan IAM.

3. Buat ekstensi

   Anda dapat membuat ekstensi menggunakan AWS AppConfig konsol atau dengan memanggil tindakan CreateExtensionAPI dari AWS CLI, AWS Tools for PowerShell, atau SDK. Walkthrough menggunakan konsol.

4. Buat asosiasi ekstensi

   Anda dapat membuat asosiasi ekstensi menggunakan AWS AppConfig konsol atau dengan memanggil tindakan CreateExtensionAssociationAPI dari AWS CLI, AWS Tools for PowerShell, atau SDK. Walkthrough menggunakan konsol.

5. Lakukan tindakan yang memanggil ekstensi

   Setelah Anda membuat asosiasi, AWS AppConfig memanggil ekstensi ketika titik tindakan yang ditentukan oleh ekstensi terjadi untuk sumber daya tersebut. Misalnya, jika Anda mengaitkan ekstensi yang berisi `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` tindakan, ekstensi akan dipanggil setiap kali Anda membuat versi konfigurasi baru yang dihosting.

Topik di bagian ini menjelaskan setiap tugas yang terlibat dalam membuat AWS AppConfig ekstensi khusus. Setiap tugas dijelaskan dalam konteks kasus penggunaan di mana pelanggan ingin membuat ekstensi yang secara otomatis mencadangkan konfigurasi ke bucket Amazon Simple Storage Service (Amazon S3). Ekstensi berjalan setiap kali konfigurasi yang dihosting dibuat (`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`) atau disebarkan (`PRE_START_DEPLOYMENT`).

Topik

- Membuat fungsi Lambda untuk ekstensi khusus AWS AppConfig
- Mengonfigurasi izin untuk ekstensi kustom AWS AppConfig

- Membuat AWS AppConfig ekstensi khusus
- Membuat asosiasi ekstensi untuk AWS AppConfig ekstensi kustom
- Melakukan tindakan yang memanggil ekstensi kustom AWS AppConfig

Membuat fungsi Lambda untuk ekstensi khusus AWS AppConfig

Untuk sebagian besar kasus penggunaan, untuk membuat ekstensi khusus, Anda harus membuat AWS Lambda fungsi untuk melakukan perhitungan dan pemrosesan apa pun yang ditentukan dalam ekstensi. Bagian ini mencakup kode contoh fungsi Lambda untuk ekstensi khusus AWS AppConfig . Bagian ini juga mencakup permintaan muatan dan detail referensi respons. Untuk informasi tentang membuat fungsi Lambda, lihat Memulai Lambda di Panduan Pengembang.AWS Lambda

Kode sampel

Contoh kode berikut untuk fungsi Lambda, saat dipanggil, secara otomatis mencadangkan AWS AppConfig konfigurasi ke bucket Amazon S3. Konfigurasi dicadangkan setiap kali konfigurasi baru dibuat atau diterapkan. Sampel menggunakan parameter ekstensi sehingga nama bucket tidak harus di-hardcode dalam fungsi Lambda. Dengan menggunakan parameter ekstensi, pengguna dapat melampirkan ekstensi ke beberapa aplikasi dan mencadangkan konfigurasi ke bucket yang berbeda. Contoh kode mencakup komentar untuk menjelaskan fungsi lebih lanjut.

Contoh fungsi Lambda untuk ekstensi AWS AppConfig

```
from datetime import datetime
import base64
import json

import boto3


def lambda_handler(event, context):
    print(event)

    # Extensions that use the PRE_CREATE_HOSTED_CONFIGURATION_VERSION and
 PRE_START_DEPLOYMENT
    # action points receive the contents of AWS AppConfig configurations in Lambda
 event parameters.
    # Configuration contents are received as a base64-encoded string, which the lambda
 needs to decode
```

```
    # in order to get the configuration data as bytes. For other action points, the
 content
    # of the configuration isn't present, so the code below will fail.
    config_data_bytes = base64.b64decode(event["Content"])

    # You can specify parameters for extensions. The CreateExtension API action lets
 you define
    # which parameters an extension supports. You supply the values for those
 parameters when you
    # create an extension association by calling the CreateExtensionAssociation API
 action.
    # The following code uses a parameter called S3_BUCKET to obtain the value
 specified in the
    # extension association. You can specify this parameter when you create the
 extension
    # later in this walkthrough.
    extension_association_params = event.get('Parameters', {})
    bucket_name = extension_association_params['S3_BUCKET']
    write_backup_to_s3(bucket_name, config_data_bytes)

    # The PRE_CREATE_HOSTED_CONFIGURATION_VERSION and PRE_START_DEPLOYMENT action
 points can
    # modify the contents of a configuration. The following code makes a minor change
    # for the purposes of a demonstration.
    old_config_data_string = config_data_bytes.decode('utf-8')
    new_config_data_string = old_config_data_string.replace('hello', 'hello!')
    new_config_data_bytes = new_config_data_string.encode('utf-8')

    # The lambda initially received the configuration data as a base64-encoded string
    # and must return it in the same format.
    new_config_data_base64string =
 base64.b64encode(new_config_data_bytes).decode('ascii')

    return {
        'statusCode': 200,
        # If you want to modify the contents of the configuration, you must include the
 new contents in the
        # Lambda response. If you don't want to modify the contents, you can omit the
 'Content' field shown here.
        'Content': new_config_data_base64string
    }


def write_backup_to_s3(bucket_name, config_data_bytes):
```

```
    s3 = boto3.resource('s3')
    new_object = s3.Object(bucket_name,
 f"config_backup_{datetime.now().isoformat()}.txt")
    new_object.put(Body=config_data_bytes)
```

Jika Anda ingin menggunakan sampel ini selama panduan ini, simpan dengan nama **`MyS3ConfigurationBackUpExtension`** dan salin Nama Sumber Daya Amazon (ARN) untuk fungsi tersebut. Anda menentukan ARN saat Anda membuat peran AWS Identity and Access Management (IAM) di bagian berikutnya. Anda menentukan ARN dan nama saat Anda membuat ekstensi.

Referensi muatan

Bagian ini mencakup permintaan payload dan rincian referensi respons untuk bekerja dengan AWS AppConfig ekstensi kustom.

Struktur permintaan

PreCreateHostedConfigurationVersion

```
{
    'InvocationId': 'vlns753', // id for specific invocation
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh', // Base64 encoded content
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'Description': '',
    'Type': 'PreCreateHostedConfigurationVersion',
    'PreviousContent': {
        'ContentType': 'text/plain',
        'ContentVersion': '1',
```

```
        'Content': 'SGVsbG8gd29ybGQh'
    }
}
```

PreStartDeployment

```
{
    'InvocationId': '765ahdm',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh',
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
    'Environment': {
        'Id': 'ibpnqlq',
        'Name': 'EnvironmentName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'Type': 'PreStartDeployment'
}
```

Peristiwa asinkron

OnStartDeployment, OnDeploymentStep, OnDeployment

```
{
    'InvocationId': 'o2xbtm7',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
```

```
    'Type': 'OnDeploymentStart',
    'Application': {
        'Id': 'abcd123'
    },
    'Environment': {
        'Id': 'efgh456'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'ConfigurationVersion': '2'
}
```

Struktur respons

Contoh berikut menunjukkan apa yang dikembalikan oleh fungsi Lambda Anda sebagai tanggapan atas permintaan dari ekstensi kustom. AWS AppConfig

Peristiwa sinkron - respons yang berhasil

Jika Anda ingin mengubah konten, gunakan yang berikut ini:

```
"Content": "SomeBase64EncodedByteArray"
```

Jika Anda tidak ingin mengubah konten, jangan kembalikan apa pun.

Peristiwa asinkron - respons yang berhasil

Tidak mengembalikan apa-apa.

Semua peristiwa kesalahan

```
{
        "Error": "BadRequestError",
        "Message": "There was malformed stuff in here",
        "Details": [{
            "Type": "Malformed",
            "Name": "S3 pointer",
            "Reason": "S3 bucket did not exist"
```

```
        }]
    }
```

Mengonfigurasi izin untuk ekstensi kustom AWS AppConfig

Gunakan prosedur berikut untuk membuat dan mengonfigurasi peran layanan AWS Identity and Access Management (IAM) (atau mengambil peran). AWS AppConfig menggunakan peran ini untuk memanggil fungsi Lambda.

Untuk membuat peran layanan IAM dan memungkinkan AWS AppConfig untuk mengasumsikan itu

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Di bawah Pilih jenis entitas tepercaya, pilih Kebijakan kepercayaan khusus.
4. Tempelkan kebijakan JSON berikut ke bidang Kebijakan kepercayaan kustom.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "appconfig.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   Pilih Berikutnya.
5. Pada halaman Tambahkan izin, pilih Buat kebijakan. Halaman Buat kebijakan terbuka di tab baru.
6. Pilih tab JSON, lalu tempelkan kebijakan izin berikut ke editor. `lambda:InvokeFunction`Tindakan ini digunakan untuk titik `PRE_*` tindakan. `lambda:InvokeAsync`Tindakan ini digunakan untuk titik `ON_*` tindakan. Ganti *`ARN Lambda Anda dengan Nama Sumber`* Daya Amazon (ARN) Lambda Anda.

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

```
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "Your Lambda ARN"
    }
  ]
}
```

7. Pilih Berikutnya: Tanda.
8. Pada halaman Tambahkan tag (Opsional), tambahkan satu atau beberapa pasangan nilai kunci dan kemudian pilih Berikutnya: Tinjau.
9. Pada halaman Kebijakan ulasan, masukkan nama dan deskripsi, lalu pilih Buat kebijakan.
10. Pada tab browser untuk kebijakan kepercayaan khusus Anda, pilih ikon Segarkan, lalu cari kebijakan izin yang baru saja Anda buat.
11. Pilih kotak centang untuk kebijakan izin Anda, lalu pilih Berikutnya.
12. Pada halaman Nama, tinjau, dan buat, masukkan nama di kotak Nama peran, lalu masukkan deskripsi.
13. Pilih Buat peran. Sistem mengembalikan Anda ke halaman Peran. Pilih Lihat peran di spanduk.
14. Salin ARN. Anda menentukan ARN ini saat Anda membuat ekstensi.

Membuat AWS AppConfig ekstensi khusus

Ekstensi mendefinisikan satu atau beberapa tindakan yang dilakukannya selama AWS AppConfig alur kerja. Misalnya, `AWS AppConfig deployment events to Amazon SNS` ekstensi yang AWS ditulis menyertakan tindakan untuk mengirim pemberitahuan ke topik Amazon SNS. Setiap tindakan dipanggil baik ketika Anda berinteraksi dengan AWS AppConfig atau ketika AWS AppConfig melakukan proses atas nama Anda. Ini disebut titik tindakan. AWS AppConfig ekstensi mendukung poin tindakan berikut:

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

Tindakan ekstensi yang dikonfigurasi pada titik PRE_* tindakan diterapkan setelah validasi permintaan, tetapi sebelum AWS AppConfig melakukan aktivitas yang sesuai dengan nama titik tindakan. Pemanggilan tindakan ini diproses pada saat yang sama sebagai permintaan. Jika lebih dari satu permintaan dibuat, pemanggilan tindakan berjalan secara berurutan. Perhatikan juga bahwa titik PRE_* tindakan menerima dan dapat mengubah konten konfigurasi. PRE_*titik tindakan juga dapat merespons kesalahan dan mencegah tindakan terjadi.

Ekstensi juga dapat berjalan secara paralel dengan AWS AppConfig alur kerja dengan menggunakan titik ON_* tindakan. ON_*titik tindakan dipanggil secara asinkron. ON_*titik tindakan tidak menerima isi konfigurasi. Jika ekstensi mengalami kesalahan selama titik ON_* tindakan, layanan mengabaikan kesalahan dan melanjutkan alur kerja.

Ekstensi contoh berikut mendefinisikan satu tindakan yang memanggil titik PRE_CREATE_HOSTED_CONFIGURATION_VERSION tindakan. Di Uri bidang, tindakan menentukan Nama Sumber Daya Amazon (ARN) dari fungsi MyS3ConfigurationBackUpExtension Lambda yang dibuat sebelumnya dalam panduan ini. Tindakan ini juga menentukan peran asumsi AWS Identity and Access Management (IAM) ARN yang dibuat sebelumnya dalam panduan ini.

AWS AppConfig Ekstensi sampel

```
{
    "Name": "MySampleExtension",
    "Description": "A sample extension that backs up configurations to an S3 bucket.",
    "Actions": {
        "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
            {
                "Name": "PreCreateHostedConfigVersionActionForS3Backup",
                "Uri": "arn:aws:lambda:aws-
region:111122223333:function:MyS3ConfigurationBackUpExtension",
                "RoleArn": "arn:aws:iam::111122223333:role/ExtensionsTestRole"
            }
        ]
    },
```

```
    "Parameters" : {
        "S3_BUCKET": {
            "Required": false
        }
    }
}
```

> **Note**
>
> Untuk melihat sintaks permintaan dan deskripsi bidang saat membuat ekstensi, lihat CreateExtensiontopik di Referensi AWS AppConfig API.

Untuk membuat ekstensi (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih AWS AppConfig.
3. Pada tab Ekstensi, pilih Buat ekstensi.
4. Untuk nama Ekstensi, masukkan nama unik. Untuk keperluan panduan ini, masukkan. **`MyS3ConfigurationBackUpExtension`** Secara opsional, masukkan deskripsi.
5. Di bagian Tindakan, pilih Tambahkan tindakan baru.
6. Untuk nama Action, masukkan nama unik. Untuk keperluan panduan ini, masukkan. **`PreCreateHostedConfigVersionActionForS3Backup`** Nama ini menjelaskan titik tindakan yang digunakan oleh tindakan dan tujuan ekstensi.
7. Dalam daftar Action point, pilih PRE_CREATE_HOSTED_CONFIGURATION_VERSION.
8. Untuk Uri, pilih fungsi Lambda dan kemudian pilih fungsi dalam daftar fungsi Lambda. Jika Anda tidak melihat fungsi Anda, verifikasi bahwa Anda berada di tempat yang sama Wilayah AWS di mana Anda membuat fungsi.
9. Untuk Peran IAM, pilih peran yang Anda buat sebelumnya dalam panduan ini.
10. Di bagian Parameter ekstensi (opsional), pilih Tambahkan parameter baru.
11. Untuk nama Parameter, masukkan nama. Untuk keperluan panduan ini, masukkan. **`S3_BUCKET`**
12. Ulangi langkah 5—11 untuk membuat tindakan kedua untuk titik `PRE_START_DEPLOYMENT` tindakan.
13. Pilih Buat ekstensi.

Menyesuaikan ekstensi AWS notifikasi yang ditulis

Anda tidak perlu membuat Lambda atau ekstensi untuk menggunakan ekstensi notifikasi yang AWS ditulis. Anda cukup membuat asosiasi ekstensi dan kemudian melakukan operasi yang memanggil salah satu titik tindakan yang didukung. Secara default, ekstensi notifikasi yang AWS ditulis mendukung poin tindakan berikut:

- `ON_DEPLOYMENT_START`
- `ON_DEPLOYMENT_COMPLETE`
- `ON_DEPLOYMENT_ROLLED_BACK`

Jika Anda membuat versi khusus dari `AWS AppConfig deployment events to Amazon SNS` ekstensi dan `AWS AppConfig deployment events to Amazon SQS` ekstensi, Anda dapat menentukan titik tindakan yang ingin Anda terima notifikasi.

> **Note**
>
> `AWS AppConfig deployment events to EventBridge`Ekstensi tidak mendukung titik `PRE_*` tindakan. Anda dapat membuat versi kustom jika Anda ingin menghapus beberapa titik tindakan default yang ditetapkan ke versi yang AWS ditulis.

Anda tidak perlu membuat fungsi Lambda jika Anda membuat versi kustom dari ekstensi notifikasi yang AWS ditulis. Anda hanya perlu menentukan Nama Sumber Daya Amazon (ARN) di `Uri` bidang untuk versi ekstensi baru.

- Untuk ekstensi EventBridge notifikasi kustom, masukkan ARN dari peristiwa EventBridge default di bidang. `Uri`
- Untuk ekstensi notifikasi Amazon SNS khusus, masukkan ARN topik Amazon SNS di bidang tersebut. `Uri`
- Untuk ekstensi notifikasi Amazon SQS kustom, masukkan ARN antrean pesan Amazon SQS di bidang. `Uri`

Membuat asosiasi ekstensi untuk AWS AppConfig ekstensi kustom

Untuk membuat ekstensi, atau mengonfigurasi ekstensi yang AWS ditulis, Anda menentukan titik tindakan yang memanggil ekstensi saat AWS AppConfig sumber daya tertentu digunakan. Misalnya,

Anda dapat memilih untuk menjalankan `AWS AppConfig deployment events to Amazon SNS` ekstensi dan menerima pemberitahuan tentang topik Amazon SNS kapan saja penerapan konfigurasi dimulai untuk aplikasi tertentu. Mendefinisikan titik tindakan mana yang memanggil ekstensi untuk AWS AppConfig sumber daya tertentu disebut asosiasi ekstensi. Asosiasi ekstensi adalah hubungan tertentu antara ekstensi dan AWS AppConfig sumber daya, seperti aplikasi atau profil konfigurasi.

Satu AWS AppConfig aplikasi dapat mencakup beberapa lingkungan dan profil konfigurasi. Jika Anda mengaitkan ekstensi ke aplikasi atau lingkungan, AWS AppConfig memanggil ekstensi untuk alur kerja apa pun yang terkait dengan sumber daya aplikasi atau lingkungan, jika berlaku.

Misalnya, Anda memiliki AWS AppConfig aplikasi bernama MobileApps yang menyertakan profil konfigurasi yang disebut AccessList. Dan katakanlah MobileApps aplikasi tersebut mencakup lingkungan Beta, Integrasi, dan Produksi. Anda membuat asosiasi ekstensi untuk ekstensi notifikasi Amazon SNS yang AWS ditulis dan mengaitkan ekstensi ke aplikasi. MobileApps Ekstensi notifikasi Amazon SNS dipanggil kapan saja konfigurasi diterapkan untuk aplikasi ke salah satu dari tiga lingkungan.

Gunakan prosedur berikut untuk membuat asosiasi AWS AppConfig ekstensi dengan menggunakan AWS AppConfig konsol.

Untuk membuat asosiasi ekstensi (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih AWS AppConfig.
3. Pada tab Ekstensi, pilih tombol opsi untuk ekstensi dan kemudian pilih Tambahkan ke sumber daya. Untuk keperluan panduan ini, pilih mys3. ConfigurationBackUpExtension
4. Di bagian Detail sumber daya ekstensi, untuk jenis sumber daya, pilih jenis AWS AppConfig sumber daya. Bergantung pada sumber daya yang Anda pilih, AWS AppConfig meminta Anda untuk memilih sumber daya lain. Untuk keperluan panduan ini, pilih Aplikasi.
5. Pilih aplikasi dalam daftar.
6. Di bagian Parameter, verifikasi bahwa S3_BUCKET terdaftar di bidang Kunci. Di bidang Nilai, tempel ARN dari ekstensi Lambda. Misalnya: `arn:aws:lambda:`*`aws-region`*`:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Pilih Buat asosiasi ke sumber daya.

Melakukan tindakan yang memanggil ekstensi kustom AWS AppConfig

Setelah Anda membuat asosiasi, Anda dapat memanggil `MyS3ConfigurationBackUpExtension` ekstensi dengan membuat profil konfigurasi baru yang menentukan `hosted` untuk itu. `SourceUri` Sebagai bagian dari alur kerja untuk membuat konfigurasi baru, AWS AppConfig temui titik `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` tindakan. Menghadapi titik tindakan ini akan memanggil `MyS3ConfigurationBackUpExtension` ekstensi, yang secara otomatis mencadangkan konfigurasi yang baru dibuat ke bucket S3 yang ditentukan di `Parameter` bagian asosiasi ekstensi.

AWS AppConfig integrasi ekstensi dengan Atlassian Jira

AWS AppConfig terintegrasi dengan Atlassian Jira. Integrasi memungkinkan AWS AppConfig untuk membuat dan memperbarui masalah di konsol Atlassian setiap kali Anda membuat perubahan pada tanda fitur di Anda Akun AWS untuk yang ditentukan. Wilayah AWS Setiap masalah Jira mencakup nama bendera, ID aplikasi, ID profil konfigurasi, dan nilai bendera. Setelah Anda memperbarui, menyimpan, dan menerapkan perubahan bendera Anda, Jira memperbarui masalah yang ada dengan detail perubahan. Untuk informasi selengkapnya, lihat Bekerja dengan ekstensi Atlassian Jira untuk AWS AppConfig.

AWS AppConfigcontoh kode

Bagian ini mencakup contoh kode untuk melakukan tindakan umum AWS AppConfig secara terprogram. Kami menyarankan Anda menggunakan sampel ini dengan Java, Python, dan JavaScriptSDK untuk melakukan tindakan di lingkungan pengujian. Bagian ini mencakup contoh kode untuk membersihkan lingkungan pengujian Anda setelah Anda selesai.

Topik

Membuat atau memperbarui konfigurasi bentuk bebas yang disimpan di toko konfigurasi yang dihosting

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode. Sampel di bagian ini memanggil API berikut:

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion

Java

```
public CreateHostedConfigurationVersionResponse createHostedConfigVersion() {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
```

```
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));

        return hcv;
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a hosted, freeform configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
    Content=b'my config data',
```

```
    ContentType='text/plain')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a hosted, freeform configuration profile
const profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: profile.Id,
    ContentType: "text/plain",
    Content: "my config data",
  })
);
```

Membuat profil konfigurasi untuk rahasia yang disimpan di Secrets Manager

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode. Sampel di bagian ini memanggil API berikut:

- CreateApplication
- CreateConfigurationProfile

Java

```
private void createSecretsManagerConfigProfile() {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a configuration profile for Secrets Manager Secret
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("secretsmanager://MySecret")
            .retrievalRoleArn("arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret")
            .type("AWS.Freeform"));
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a configuration profile for Secrets Manager Secret
config_profile = appconfig.create_configuration_profile(
```

```
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='secretsmanager://MySecret',
    RetrievalRoleArn='arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret',
    Type='AWS.Freeform')
```

JavaScript

```
import {
  AppConfigClient,
  CreateConfigurationProfileCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a configuration profile for Secrets Manager Secret
await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "secretsmanager://MySecret",
    RetrievalRoleArn: "arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret",
    Type: "AWS.Freeform",
  })
);
```

Menerapkan profil konfigurasi

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode. Sampel di bagian ini memanggil API berikut:

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion

- CreateEnvironment
- StartDeployment
- GetDeployment

Java

```
private void createDeployment() throws InterruptedException {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));


        // Create an environment
        CreateEnvironmentResponse env = appconfig.createEnvironment(req -> req
            .applicationId(app.id())
            .name("Beta")
            // If you have CloudWatch alarms that monitor the health of your
 service, you can add them here and they
            // will trigger a rollback if they fire during an appconfig deployment
            //.monitors(Monitor.builder().alarmArn("arn:aws:cloudwatch:us-
east-1:520900602629:alarm:MyAlarm")
            //
  .alarmRoleArn("arn:aws:iam::520900602629:role/MyAppConfigAlarmRole").build())
        );
```

```
        // Start a deployment
        StartDeploymentResponse deploymentResponse = appconfig.startDeployment(req -
> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .environmentId(env.id())
            .configurationVersion(hcv.versionNumber().toString())
            .deploymentStrategyId("AppConfig.Linear50PercentEvery30Seconds")
        );

        // Wait for deployment to complete
        List<DeploymentState> nonFinalDeploymentStates = Arrays.asList(
            DeploymentState.DEPLOYING,
            DeploymentState.BAKING,
            DeploymentState.ROLLING_BACK,
            DeploymentState.VALIDATING);
        GetDeploymentRequest getDeploymentRequest =
 GetDeploymentRequest.builder().applicationId(app.id())

 .environmentId(env.id())

 .deploymentNumber(deploymentResponse.deploymentNumber()).build();
        GetDeploymentResponse deployment =
 appconfig.getDeployment(getDeploymentRequest);
        while (nonFinalDeploymentStates.contains(deployment.state())) {
            System.out.println("Waiting for deployment to complete: " + deployment);
            Thread.sleep(1000L);
            deployment = appconfig.getDeployment(getDeploymentRequest);
        }

        System.out.println("Deployment complete: " + deployment);
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')
```

```
# create an environment
environment = appconfig.create_environment(
    ApplicationId=application['Id'],
    Name='MyEnvironment')

# create a configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
    Content=b'my config data',
    ContentType='text/plain')

# start a deployment
deployment = appconfig.start_deployment(
    ApplicationId=application['Id'],
    EnvironmentId=environment['Id'],
    ConfigurationProfileId=config_profile['Id'],
    ConfigurationVersion=str(hcv['VersionNumber']),
    DeploymentStrategyId='AppConfig.Linear20PercentEvery6Minutes')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
  CreateEnvironmentCommand,
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
  StartDeploymentCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
```

```
);

// create an environment
const environment = await appconfig.send(
  new CreateEnvironmentCommand({
    ApplicationId: application.Id,
    Name: "MyEnvironment",
  })
);

// create a configuration profile
const config_profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
const hcv = await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: config_profile.Id,
    Content: "my config data",
    ContentType: "text/plain",
  })
);

// start a deployment
await appconfig.send(
  new StartDeploymentCommand({
    ApplicationId: application.Id,
    EnvironmentId: environment.Id,
    ConfigurationProfileId: config_profile.Id,
    ConfigurationVersion: hcv.VersionNumber.toString(),
    DeploymentStrategyId: "AppConfig.Linear20PercentEvery6Minutes",
  })
);
```

Menggunakan AWS AppConfig Agen untuk membaca profil konfigurasi bentuk bebas

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode.

Java

```
public void retrieveConfigFromAgent() throws Exception {
        /*
        In this sample, we will retrieve configuration data from the AWS AppConfig
 Agent.
        The agent is a sidecar process that handles retrieving configuration data
 from AppConfig
        for you in a way that implements best practices like configuration caching.

        For more information about the agent, see Simplified retrieval methods
        */

        // The agent runs a local HTTP server that serves configuration data
        // Make a GET request to the agent's local server to retrieve the
 configuration data
        URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyConfigProfile");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        StringBuilder content;
        try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
            content = new StringBuilder();
            int ch;
            while ((ch = in.read()) != -1) {
                content.append((char) ch);
            }
        }
        con.disconnect();
        System.out.println("Configuration from agent via HTTP: " + content);
    }
```

Python

```
# in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
```

```
# the agent is a sidecar process that handles retrieving configuration data from AWS
 AppConfig
# for you in a way that implements best practices like configuration caching.
#
# for more information about the agent, see
# Simplified retrieval methods
#

import requests

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

# the agent runs a local HTTP server that serves configuration data
# make a GET request to the agent's local server to retrieve the configuration data
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}")
config = response.content
```

JavaScript

```
// in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
// the agent is a sidecar process that handles retrieving configuration data from
 AppConfig
// for you in a way that implements best practices like configuration caching.

// for more information about the agent, see
// Simplified retrieval methods

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// the agent runs a local HTTP server that serves configuration data
// make a GET request to the agent's local server to retrieve the configuration data
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}`;
const response = await fetch(url);
const config = await response.text(); // (use `await response.json()` if your config
 is json)
```

Menggunakan AWS AppConfig Agen untuk membaca bendera fitur tertentu

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode.

Java

```
public void retrieveSingleFlagFromAgent() throws Exception {
        /*
          You can retrieve a single flag's data from the agent by providing the
 "flag" query string parameter.
          Note: the configuration's type must be AWS.AppConfig.FeatureFlags
        */

        URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyFlagsProfile?flag=myFlagKey");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        StringBuilder content;
        try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
            content = new StringBuilder();
            int ch;
            while ((ch = in.read()) != -1) {
                content.append((char) ch);
            }
        }
        con.disconnect();
        System.out.println("MyFlagName from agent: " + content);
    }
```

Python

```
import requests

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'
flag_key = 'MyFlag'

# retrieve a single flag's data by providing the "flag" query string parameter
# note: the configuration's type must be AWS.AppConfig.FeatureFlags
```

```
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}?
flag={flag_key}")
config = response.content
```

JavaScript

```
const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";
const flag_name = "MyFlag";

// retrieve a single flag's data by providing the "flag" query string parameter
// note: the configuration's type must be AWS.AppConfig.FeatureFlags
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}?flag=${flag_name}`;
const response = await fetch(url);
const flag = await response.json(); // { "enabled": true/false }
```

Menggunakan tindakan GetLatestConfig API untuk membaca profil konfigurasi bentuk bebas

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode. Sampel di bagian ini memanggil API berikut:

- GetLatestConfiguration
- StartConfigurationSession

Java

```
public void retrieveConfigFromApi() {
        /*
        The example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.
        For more information on these APIs, see AWS AppConfig Data         */
        AppConfigDataClient appConfigData = AppConfigDataClient.create();

        /*
```

```
        Start a new configuration session using the StartConfigurationSession API.
This operation does not return configuration data.
        Rather, it returns an initial configuration token that should be passed to
GetLatestConfiguration.
        IMPORTANT: This operation should only be performed once (per configuration),
prior to the first GetLatestConfiguration
        call you preform. Each GetLatestConfiguration will return a new
configuration token that you should then use in the
        next GetLatestConfiguration call.
        */
        StartConfigurationSessionResponse session =
            appConfigData.startConfigurationSession(req -> req
                .applicationIdentifier("MyDemoApp")
                .configurationProfileIdentifier("MyConfigProfile")
                .environmentIdentifier("Beta"));

        /*
        Retrieve configuration data using the GetLatestConfiguration API. The first
time you call this API your configuration
        data will be returned. You should cache that data (and the configuration
token) and update that cache asynchronously
        by regularly polling the GetLatestConfiguration API in a background thread.
If you already have the latest configuration
        data, subsequent GetLatestConfiguration calls will return an empty response.
If you then deploy updated configuration
        data the next time you call GetLatestConfiguration it will return that
updated data.

        You can also avoid all the complexity around writing this code yourself by
leveraging our agent instead.
        For more information about the agent, see Simplified retrieval methods
        */

        // The first getLatestConfiguration call uses the token from
StartConfigurationSession
        String configurationToken = session.initialConfigurationToken();
        GetLatestConfigurationResponse configuration =

appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

        System.out.println("Configuration retrieved via API: " +
configuration.configuration().asUtf8String());
```

```
        // You'll want to hold on to the token in the getLatestConfiguration
 response because you'll need to use it
        // the next time you call
        configurationToken = configuration.nextPollConfigurationToken();
        configuration =

 appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

        // Try creating a new deployment at this point to see how the output below
 changes.
        if (configuration.configuration().asByteArray().length != 0) {
            System.out.println("Configuration contents have changed
 since the last GetLatestConfiguration call, new contents = " +
 configuration.configuration().asUtf8String());
        } else {
            System.out.println("GetLatestConfiguration returned an empty response
 because we already have the latest configuration");
        }
    }
```

Python

```
# the example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.
#
# for more information on these APIs, see
# AWS AppConfig Data
#

import boto3

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

appconfigdata = boto3.client('appconfigdata')

# start a new configuration session.
# this operation does not return configuration data.
# rather, it returns an initial configuration token that should be passed to
 GetLatestConfiguration.
#
# note: this operation should only be performed once (per configuration).
```

```
#   all subsequent calls to AppConfigData should be via GetLatestConfiguration.
scs = appconfigdata.start_configuration_session(
    ApplicationIdentifier=application_name,
    EnvironmentIdentifier=environment_name,
    ConfigurationProfileIdentifier=config_profile_name)
initial_token = scs['InitialConfigurationToken']

# retrieve configuration data from the session.
# this operation returns your configuration data.
# each invocation of this operation returns a unique token that should be passed to
 the subsequent invocation.
#
# note: this operation does not always return configuration data after the first
 invocation.
#   data is only returned if the configuration has changed within AWS AppConfig
 (i.e. a deployment occurred).
#   therefore, you should cache the data returned by this call so that you can use
 it later.
glc = appconfigdata.get_latest_configuration(ConfigurationToken=initial_token)
config = glc['Configuration'].read()
```

JavaScript

```
// the example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.

// for more information on these APIs, see
// AWS AppConfig Data

import {
  AppConfigDataClient,
  GetLatestConfigurationCommand,
  StartConfigurationSessionCommand,
} from "@aws-sdk/client-appconfigdata";

const appconfigdata = new AppConfigDataClient();

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// start a new configuration session.
// this operation does not return configuration data.
```

```
// rather, it returns an initial configuration token that should be passed to
 GetLatestConfiguration.
//
// note: this operation should only be performed once (per configuration).
//   all subsequent calls to AppConfigData should be via GetLatestConfiguration.
const scs = await appconfigdata.send(
  new StartConfigurationSessionCommand({
    ApplicationIdentifier: application_name,
    EnvironmentIdentifier: environment_name,
    ConfigurationProfileIdentifier: config_profile_name,
  })
);
const { InitialConfigurationToken } = scs;

// retrieve configuration data from the session.
// this operation returns your configuration data.
// each invocation of this operation returns a unique token that should be passed to
 the subsequent invocation.
//
// note: this operation does not always return configuration data after the first
 invocation.
//   data is only returned if the configuration has changed within AWS AppConfig
 (i.e. a deployment occurred).
//   therefore, you should cache the data returned by this call so that you can use
 it later.
const glc = await appconfigdata.send(
  new GetLatestConfigurationCommand({
    ConfigurationToken: InitialConfigurationToken,
  })
);
const config = glc.Configuration.transformToString();
```

Membersihkan lingkungan Anda

Jika Anda menjalankan satu atau beberapa contoh kode di bagian ini, kami sarankan Anda menggunakan salah satu sampel berikut untuk mencari dan menghapus AWS AppConfig sumber daya yang dibuat oleh sampel kode tersebut. Sampel di bagian ini memanggil API berikut:

- ListApplications
- DeleteApplication
- ListEnvironments

- DeleteEnvironments
- ListConfigurationProfiles
- DeleteConfigurationProfile
- ListHostedConfigurationVersions
- DeleteHostedConfigurationVersion

Java

```
/*
    This sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.

   WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
   an application that you may need in the future.
   */

   public void cleanUpDemoResources() {
       AppConfigClient appconfig = AppConfigClient.create();

       // The name of the application to delete
       // IMPORTANT: verify this name corresponds to the application you wish to
 delete
       String applicationToDelete = "MyDemoApp";


 appconfig.listApplicationsPaginator(ListApplicationsRequest.builder().build()).items().forE
 -> {
           if (app.name().equals(applicationToDelete)) {
               System.out.println("Deleting App: " + app);
               appconfig.listConfigurationProfilesPaginator(req ->
 req.applicationId(app.id())).items().forEach(cp -> {
                   System.out.println("Deleting Profile: " + cp);
                   appconfig
                       .listHostedConfigurationVersionsPaginator(req -> req
                           .applicationId(app.id())
                           .configurationProfileId(cp.id()))
                       .items()
                       .forEach(hcv -> {
```

```
                            System.out.println("Deleting HCV: " + hcv);
                            appconfig.deleteHostedConfigurationVersion(req -> req
                                .applicationId(app.id())
                                .configurationProfileId(cp.id())
                                .versionNumber(hcv.versionNumber()));
                        });
                    appconfig.deleteConfigurationProfile(req -> req
                        .applicationId(app.id())
                        .configurationProfileId(cp.id()));
                });

                appconfig.listEnvironmentsPaginator(req-
>req.applicationId(app.id())).items().forEach(env -> {
                    System.out.println("Deleting Environment: " + env);
                    appconfig.deleteEnvironment(req-
>req.applicationId(app.id()).environmentId(env.id()));
                });

                appconfig.deleteApplication(req -> req.applicationId(app.id()));
            }
        });
    }
```

Python

```
# this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.
#
# WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
#   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
#   an application that you may need in the future.
#

import boto3

# the name of the application to delete
# IMPORTANT: verify this name corresponds to the application you wish to delete
application_name = 'MyDemoApp'

# create and iterate over a list paginator such that we end up with a list of pages,
 which are themselves lists of applications
```

```
# e.g. [ [{'Name':'MyApp1',...},{'Name':'MyApp2',...}], [{'Name':'MyApp3',...}] ]
list_of_app_lists = [page['Items'] for page in
 appconfig.get_paginator('list_applications').paginate()]
# retrieve the target application from the list of lists
application = [app for apps in list_of_app_lists for app in apps if app['Name'] ==
 application_name][0]
print(f"deleting application {application['Name']} (id={application['Id']})")

# delete all configuration profiles
list_of_config_lists = [page['Items'] for page in
 appconfig.get_paginator('list_configuration_profiles').paginate(ApplicationId=application['
for config_profile in [config for configs in list_of_config_lists for config in
 configs]:
    print(f"\tdeleting configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

    # delete all hosted configuration versions
    list_of_hcv_lists = [page['Items'] for page in
 appconfig.get_paginator('list_hosted_configuration_versions').paginate(ApplicationId=applic
 ConfigurationProfileId=config_profile['Id'])]
    for hcv in [hcv for hcvs in list_of_hcv_lists for hcv in hcvs]:

 appconfig.delete_hosted_configuration_version(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'], VersionNumber=hcv['VersionNumber'])
        print(f"\t\tdeleted hosted configuration version {hcv['VersionNumber']}")

    # delete the config profile itself
    appconfig.delete_configuration_profile(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'])
    print(f"\tdeleted configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

# delete all environments
list_of_env_lists = [page['Items'] for page in
 appconfig.get_paginator('list_environments').paginate(ApplicationId=application['Id'])]
for environment in [env for envs in list_of_env_lists for env in envs]:
    appconfig.delete_environment(ApplicationId=application['Id'],
 EnvironmentId=environment['Id'])
    print(f"\tdeleted environment {environment['Name']} (Id={environment['Id']})")

# delete the application itself
appconfig.delete_application(ApplicationId=application['Id'])
print(f"deleted application {application['Name']} (id={application['Id']})")
```

JavaScript

```
// this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.

// WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
//   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
//   an application that you may need in the future.

import {
  AppConfigClient,
  paginateListApplications,
  DeleteApplicationCommand,
  paginateListConfigurationProfiles,
  DeleteConfigurationProfileCommand,
  paginateListHostedConfigurationVersions,
  DeleteHostedConfigurationVersionCommand,
  paginateListEnvironments,
  DeleteEnvironmentCommand,
} from "@aws-sdk/client-appconfig";

const client = new AppConfigClient();

// the name of the application to delete
// IMPORTANT: verify this name corresponds to the application you wish to delete
const application_name = "MyDemoApp";

// iterate over all applications, deleting ones that have the name defined above
for await (const app_page of paginateListApplications({ client }, {})) {
  for (const application of app_page.Items) {

    // skip applications that dont have the name thats set
    if (application.Name !== application_name) continue;

    console.log( `deleting application ${application.Name} (id=${application.Id})`);

    // delete all configuration profiles
    for await (const config_page of paginateListConfigurationProfiles({ client },
 { ApplicationId: application.Id })) {
      for (const config_profile of config_page.Items) {
        console.log(`\tdeleting configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`);
```

```
        // delete all hosted configuration versions
        for await (const hosted_page of
 paginateListHostedConfigurationVersions({ client },
          { ApplicationId: application.Id, ConfigurationProfileId:
 config_profile.Id }
        )) {
          for (const hosted_config_version of hosted_page.Items) {
            await client.send(
              new DeleteHostedConfigurationVersionCommand({
                ApplicationId: application.Id,
                ConfigurationProfileId: config_profile.Id,
                VersionNumber: hosted_config_version.VersionNumber,
              })
            );
            console.log(`\t\tdeleted hosted configuration version
 ${hosted_config_version.VersionNumber}`);
          }
        }

        // delete the config profile itself
        await client.send(
          new DeleteConfigurationProfileCommand({
            ApplicationId: application.Id,
            ConfigurationProfileId: config_profile.Id,
          })
        );
        console.log(`\tdeleted configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`)
      }

      // delete all environments
      for await (const env_page of paginateListEnvironments({ client },
 { ApplicationId: application.Id })) {
        for (const environment of env_page.Items) {
          await client.send(
            new DeleteEnvironmentCommand({
              ApplicationId: application.Id,
              EnvironmentId: environment.Id,
            })
          );
          console.log(`\tdeleted environment ${environment.Name} (Id=
${environment.Id})`)
        }
```

```
      }
    }

    // delete the application itself
    await client.send(
      new DeleteApplicationCommand({ ApplicationId: application.Id })
    );
    console.log(`deleted application ${application.Name} (id=${application.Id})`)
  }
}
```

Keamanan di AWS AppConfig

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan dari organisasi yang paling sensitif terhadap keamanan.

Keamanan menjadi tanggung jawab bersama antara 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 menjalankan layanan-layanan AWS di AWS Cloud Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga menguji dan memverifikasi secara berkala efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlakuAWS Systems Manager, lihat AWSLayanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- Keamanan di cloud – Tanggung jawab Anda ditentukan oleh layanan AWS yang digunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

AWS AppConfigadalah kemampuanAWS Systems Manager. Untuk memahami cara menerapkan model tanggung jawab bersama saat menggunakanAWS AppConfig, lihat Keamanan di AWS Systems Manager. Bagian tersebut menjelaskan cara mengonfigurasi Systems Manager untuk memenuhi tujuan keamanan dan kepatuhanAWS AppConfig.

Terapkan akses hak akses paling rendah

Sebagai praktik keamanan terbaik, berikan izin minimum yang diperlukan yang diperlukan identitas untuk melakukan tindakan spesifik pada sumber daya tertentu dalam kondisi tertentu. AWS AppConfig Agen menawarkan dua fitur yang memungkinkan agen untuk mengakses filesystem dari sebuah instance atau container: backup dan write to disk. Jika Anda mengaktifkan fitur ini, verifikasi bahwa hanya AWS AppConfig Agen yang memiliki izin untuk menulis ke file konfigurasi yang ditunjuk pada sistem file. Juga verifikasi bahwa hanya proses yang diperlukan untuk membaca dari file konfigurasi ini yang memiliki kemampuan untuk melakukannya. Menerapkan akses hak akses paling rendah adalah hal mendasar dalam mengurangi risiko keamanan dan dampak yang dapat diakibatkan oleh kesalahan atau niat jahat.

Untuk informasi selengkapnya tentang penerapan akses hak istimewa terkecil, lihat SEC03-BP02 Memberikan akses hak istimewa terkecil di Panduan Pengguna. AWS Well-Architected Tool Untuk informasi selengkapnya tentang fitur AWS AppConfig Agen yang disebutkan di bagian ini, lihatFitur pengambilan tambahan.

Enkripsi data saat istirahat untuk AWS AppConfig

AWS AppConfigmenyediakan enkripsi secara default untuk melindungi data pelanggan saat istirahat menggunakanKunci milik AWS.

Kunci milik AWS— AWS AppConfig menggunakan kunci ini secara default untuk secara otomatis mengenkripsi data yang digunakan oleh layanan dan dihosting di penyimpanan AWS AppConfig data. Anda tidak dapat melihat, mengelola, atau menggunakanKunci milik AWS, atau mengaudit penggunaannya. Namun, Anda tidak perlu mengambil tindakan apa pun atau mengubah program apa pun untuk melindungi kunci yang mengenkripsi data Anda. Untuk informasi lebih lanjut, lihat Kunci milik AWS dalam Panduan PengembangAWS Key Management Service.

Meskipun Anda tidak dapat menonaktifkan lapisan enkripsi ini atau memilih jenis enkripsi alternatif, Anda dapat menentukan kunci yang dikelola pelanggan yang akan digunakan saat menyimpan data konfigurasi yang dihosting di penyimpanan AWS AppConfig data dan saat Anda menerapkan data konfigurasi.

Kunci terkelola pelanggan — AWS AppConfig mendukung penggunaan kunci terkelola pelanggan simetris yang Anda buat, miliki, dan kelola untuk menambahkan lapisan enkripsi kedua di atas yang adaKunci milik AWS. Karena Anda memiliki kontrol penuh atas lapisan enkripsi ini, Anda dapat melakukan tugas-tugas seperti:

- Menetapkan dan memelihara kebijakan dan hibah utama
- Menetapkan dan memelihara kebijakan IAM
- Mengaktifkan dan menonaktifkan kebijakan utama
- Memutar bahan kriptografi kunci
- Menambahkan tanda
- Membuat alias kunci
- Kunci penjadwalan untuk penghapusan

Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

AWS AppConfigmendukung kunci yang dikelola pelanggan

AWS AppConfigmenawarkan dukungan untuk enkripsi kunci yang dikelola pelanggan untuk data konfigurasi. Untuk versi konfigurasi yang disimpan ke penyimpanan data yang AWS AppConfig dihosting, pelanggan dapat mengatur profil konfigurasi yang sesuai. `KmsKeyIdentifier` Setiap kali versi baru data konfigurasi dibuat menggunakan operasi `CreateHostedConfigurationVersion` API, AWS AppConfig menghasilkan kunci AWS KMS data dari `KmsKeyIdentifier` untuk mengenkripsi data sebelum menyimpannya. Ketika data kemudian diakses, baik selama operasi `GetHostedConfigurationVersion` atau `StartDeployment` API, AWS AppConfig mendekripsi data konfigurasi menggunakan informasi tentang kunci data yang dihasilkan.

AWS AppConfigjuga menawarkan dukungan untuk enkripsi kunci yang dikelola pelanggan untuk data konfigurasi yang digunakan. Untuk mengenkripsi data konfigurasi, pelanggan dapat menyediakan `KmsKeyIdentifier` untuk penyebaran mereka. AWS AppConfigmenghasilkan kunci AWS KMS data dengan ini `KmsKeyIdentifier` untuk mengenkripsi data pada operasi `StartDeployment` API.

AWS AppConfigakses enkripsi

Saat membuat kunci yang dikelola pelanggan, gunakan kebijakan kunci berikut untuk memastikan bahwa kunci tersebut dapat digunakan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "Allow use of the key",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::account_ID:role/role_name"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
            ],
        "Resource": "*"
        }
 ]
```

Untuk mengenkripsi data konfigurasi yang dihosting dengan kunci terkelola pelanggan, panggilan identitas `CreateHostedConfigurationVersion` memerlukan pernyataan kebijakan berikut yang dapat ditetapkan ke pengguna, grup, atau peran:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Jika Anda menggunakan rahasia Secrets Manager atau data konfigurasi lainnya yang dienkripsi dengan kunci yang dikelola pelanggan, Anda `retrievalRoleArn` harus `kms:Decrypt` mendekripsi dan mengambil data.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:account_ID:configuration source/object"
    }
  ]
}
```

Saat memanggil operasi AWS AppConfig StartDeploymentAPI, pemanggilan identitas `StartDeployment` memerlukan kebijakan IAM berikut yang dapat ditetapkan ke pengguna, grup, atau peran:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
```

```
      ],
      "Resource": "arn:aws:kms:Region:account_ID:key_ID"
    }
  ]
}
```

Saat memanggil operasi AWS AppConfig GetLatestConfigurationAPI, pemanggilan identitas `GetLatestConfiguration` memerlukan kebijakan berikut yang dapat ditetapkan ke pengguna, grup, atau peran:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Konteks enkripsi

Konteks enkripsi adalah kumpulan opsional pasangan kunci-nilai yang berisi informasi kontekstual tambahan tentang data.

AWS KMSmenggunakan konteks enkripsi sebagai data otentikasi tambahan untuk mendukung enkripsi yang diautentikasi. Bila Anda menyertakan konteks enkripsi dalam permintaan untuk mengenkripsi data, AWS KMS mengikat konteks enkripsi ke data terenkripsi. Untuk mendekripsi data, Anda menyertakan konteks enkripsi yang sama dalam permintaan.

AWS AppConfigkonteks enkripsi: AWS AppConfig menggunakan konteks enkripsi di semua operasi AWS KMS kriptografi untuk data dan penerapan konfigurasi host terenkripsi. Konteks berisi kunci yang sesuai dengan jenis data dan nilai yang mengidentifikasi item data tertentu.

Memantau kunci enkripsi Anda untuk AWS

Saat menggunakan kunci terkelola AWS KMS pelanggan AWS AppConfig, Anda dapat menggunakan AWS CloudTrail atau Amazon CloudWatch Logs untuk melacak permintaan yang AWS AppConfig dikirimAWS KMS.

Contoh berikut adalah CloudTrail peristiwa untuk memantau AWS KMS operasi yang dipanggil oleh `Decrypt` AWS AppConfig untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "appconfig.amazonaws.com"
    },
    "eventTime": "2023-01-03T02:22:28z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "Region",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:appconfig:deployment:arn":
 "arn:aws:appconfig:Region:account_ID:application/application_ID/
environment/environment_ID/deployment/deployment_ID"
        },
        "keyId": "arn:aws:kms:Region:account_ID:key/key_ID",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account_ID",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "account_ID",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

Akses AWS AppConfig menggunakan endpoint antarmuka () AWS PrivateLink

Anda dapat menggunakan AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan. AWS AppConfig Anda dapat mengakses AWS AppConfig 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. AWS AppConfig

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka, yang didukung olehAWS 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. AWS AppConfig

Untuk informasi selengkapnya, lihat Mengakses Layanan AWS melalui AWS PrivateLink di Panduan AWS PrivateLink.

Pertimbangan untuk AWS AppConfig

Sebelum Anda menyiapkan titik akhir antarmukaAWS AppConfig, tinjau Pertimbangan dalam Panduan. AWS PrivateLink

AWS AppConfigmendukung membuat panggilan ke `appconfig`dan `appconfigdata`layanan melalui titik akhir antarmuka.

Buat titik akhir antarmuka untuk AWS AppConfig

Anda dapat membuat titik akhir antarmuka untuk AWS AppConfig menggunakan konsol VPC Amazon atau ()AWS Command Line Interface. AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLinkPanduan.

Buat titik akhir antarmuka untuk AWS AppConfig menggunakan nama layanan berikut:

```
com.amazonaws.region.appconfig
```

```
com.amazonaws.region.appconfigdata
```

Jika Anda mengaktifkan DNS pribadi untuk titik akhir antarmuka, Anda dapat membuat permintaan API untuk AWS AppConfig menggunakan nama DNS Regional default. Misalnya, `appconfig.us-east-1.amazonaws.com` dan `appconfigdata.us-east-1.amazonaws.com`.

Buat kebijakan titik akhir untuk titik akhir antarmuka Anda

Kebijakan endpoint adalah sumber daya IAM yang dapat Anda lampirkan ke titik akhir antarmuka. Kebijakan endpoint default memungkinkan akses penuh AWS AppConfig melalui titik akhir antarmuka. Untuk mengontrol akses yang diizinkan AWS AppConfig dari VPC Anda, lampirkan kebijakan titik akhir kustom ke titik akhir antarmuka.

kebijakan titik akhir mencantumkan informasi berikut:

- Prinsipal yang dapat melakukan tindakan (Akun AWS, pengguna IAM, dan peran IAM).
- Tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan menggunakan kebijakan titik akhir di Panduan AWS PrivateLink.

Contoh: Kebijakan VPC endpoint untuk tindakan AWS AppConfig

Berikut ini adalah contoh kebijakan endpoint kustom. Saat Anda melampirkan kebijakan ini ke titik akhir antarmuka Anda, kebijakan ini akan memberikan akses ke AWS AppConfig tindakan yang tercantum untuk semua prinsip di semua sumber daya.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "appconfig:CreateApplication",
            "appconfig:CreateEnvironment",
            "appconfig:CreateConfigurationProfile",
            "appconfig:StartDeployment",
            "appconfig:GetLatestConfiguration"
            "appconfig:StartConfigurationSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Rotasi kunci Secrets Manager

Bagian ini menjelaskan informasi keamanan penting tentang AWS AppConfig integrasi dengan Secrets Manager. Untuk informasi tentang Secrets Manager, lihat Apa ituAWS Secrets Manager? dalam AWS Secrets ManagerUser Guide.

Menyiapkan rotasi otomatis rahasia Secrets Manager yang digunakan oleh AWS AppConfig

Rotasi adalah proses memperbarui rahasia yang disimpan di Secrets Manager secara berkala. Ketika Anda memutar rahasia, Anda memperbarui kredensil di kedua rahasia dan database atau layanan. Anda dapat mengonfigurasi rotasi rahasia otomatis di Secrets Manager dengan menggunakan AWS Lambda fungsi untuk memperbarui rahasia dan database. Untuk informasi selengkapnya, lihat Memutar AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Untuk mengaktifkan rotasi kunci rahasia Secrets Manager yang digunakan olehAWS AppConfig, perbarui fungsi Lambda rotasi Anda dan gunakan rahasia yang diputar.

> **Note**
>
> Terapkan profil AWS AppConfig konfigurasi Anda setelah rahasia Anda diputar dan diperbarui sepenuhnya ke versi baru. Anda dapat menentukan apakah rahasia diputar karena status `VersionStage` perubahan dari `AWSPENDING` ke`AWSCURRENT`. Penyelesaian rotasi rahasia terjadi dalam `finish_secret` fungsi Secrets Manager Rotation Templates.

Berikut adalah contoh fungsi yang memulai AWS AppConfig penerapan setelah rahasia diputar.

```
import time
import boto3
client = boto3.client('appconfig')

def finish_secret(service_client, arn, new_version):
    """Finish the rotation by marking the pending secret as current
    This method finishes the secret rotation by staging the secret staged AWSPENDING
 with the AWSCURRENT stage.
    Args:
        service_client (client): The secrets manager service client
        arn (string): The secret ARN or other identifier
```

```
        new_version (string): The new version to be associated with the secret
    """
    # First describe the secret to get the current version
    metadata = service_client.describe_secret(SecretId=arn)
    current_version = None
    for version in metadata["VersionIdsToStages"]:
        if "AWSCURRENT" in metadata["VersionIdsToStages"][version]:
            if version == new_version:
                # The correct version is already marked as current, return
                logger.info("finishSecret: Version %s already marked as AWSCURRENT for
%s" % (version, arn))
                return
            current_version = version
            break

    # Finalize by staging the secret version current
    service_client.update_secret_version_stage(SecretId=arn, VersionStage="AWSCURRENT",
MoveToVersionId=new_version, RemoveFromVersionId=current_version)

    # Deploy rotated secret
    response = client.start_deployment(
            ApplicationId='TestApp',
            EnvironmentId='TestEnvironment',
            DeploymentStrategyId='TestStrategy',
            ConfigurationProfileId='ConfigurationProfileId',
            ConfigurationVersion=new_version,
            KmsKeyIdentifier=key,
            Description='Deploy secret rotated at ' + str(time.time())
        )

    logger.info("finishSecret: Successfully set AWSCURRENT stage to version %s for
secret %s." % (new_version, arn))
```

AWS AppConfig Pemantauan

Pemantauan adalah bagian penting dari pemeliharaan keandalan, ketersediaan, dan kinerja AWS AppConfig dan solusi AWS lain Anda. AWS menyediakan alat pemantauan berikut untuk memantau AWS AppConfig, melaporkan jika ada yang salah, dan mengambil tindakan otomatis jika diperlukan:

- AWS CloudTrail merekam panggilan API dan peristiwa terkait yang dilakukan oleh atau atas nama akun AWS Anda dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang memanggil AWS, alamat IP sumber yang melakukan panggilan, dan kapan panggilan tersebut terjadi. Untuk mengetahui informasi selengkapnya, lihat Panduan Pengguna AWS CloudTrail.
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari instans Amazon EC2, 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 tahan lama. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.

Topik

Mencatat panggilan API AWS AppConfig menggunakan AWS CloudTrail

AWS AppConfigterintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan diAWS AppConfig. CloudTrail menangkap semua panggilan API untuk AWS AppConfig sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari AWS AppConfig konsol dan panggilan kode ke operasi API AWS AppConfig ini. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. AWS AppConfig Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuatAWS AppConfig, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWS AppConfiginformasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi diAWS AppConfig, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di Akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

Untuk catatan berkelanjutan tentang peristiwa di Akun AWS, termasuk peristiwa untuk AWS AppConfig, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua AWS AppConfig tindakan dicatat oleh CloudTrail dan didokumentasikan dalam Referensi AWS AppConfig API. Misalnya, panggilan ke`CreateApplication`, `GetApplication` dan `ListApplications` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Bahwa permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna (IAM) AWS Identity and Access Management.
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

AWS AppConfigperistiwa data di CloudTrail

Peristiwa data memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya (misalnya, mengambil konfigurasi terbaru yang diterapkan dengan memanggil GetLatestConfiguration). 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 CloudTrailHarga.

Anda dapat mencatat peristiwa data untuk jenis AWS AppConfig sumber daya menggunakan CloudTrail konsolAWS CLI, atau operasi CloudTrail API. Tabel di bagian ini menunjukkan jenis sumber daya yang tersedia untukAWS AppConfig.

- Untuk mencatat peristiwa data menggunakan CloudTrail konsol, buat penyimpanan data jejak atau peristiwa untuk mencatat peristiwa data, atau perbarui penyimpanan data jejak atau peristiwa yang ada untuk mencatat peristiwa data.

  1. Pilih Peristiwa data untuk mencatat peristiwa data.
  2. Dari daftar tipe peristiwa Data, pilih AWS AppConfig.
  3. Pilih template pemilih log yang ingin Anda gunakan. Anda dapat mencatat semua peristiwa data untuk jenis sumber daya, mencatat semua `readOnly` peristiwa, mencatat semua `writeOnly` peristiwa, atau membuat templat pemilih log khusus untuk memfilter pada `readOnly`eventName, dan `resources.`ARN bidang.
  4. Untuk nama Selector, masukkan AppConfigDataEvents. Untuk informasi tentang mengaktifkan CloudWatch Log Amazon untuk jejak peristiwa data Anda, lihatMetrik pencatatan untuk panggilan pesawat AWS AppConfig data.
- Untuk mencatat peristiwa data menggunakanAWS CLI, konfigurasikan `--advanced-event-selectors` parameter untuk mengatur `eventCategory` bidang sama dengan `Data` dan `resources.type` bidang sama dengan nilai tipe sumber daya (lihat tabel). Anda dapat menambahkan kondisi untuk memfilter nilai`readOnly`,`eventName`, dan `resources.`ARN bidang.
  - Untuk mengonfigurasi jejak untuk mencatat peristiwa data, jalankan put-event-selectorsperintah. Untuk informasi selengkapnya, lihat Mencatat peristiwa data untuk jejak dengan. AWS CLI

- Untuk mengonfigurasi penyimpanan data peristiwa untuk mencatat peristiwa data, jalankan create-event-data-storeperintah untuk membuat penyimpanan data peristiwa baru untuk mencatat peristiwa data, atau jalankan update-event-data-storeperintah untuk memperbarui penyimpanan data peristiwa yang ada. Untuk informasi selengkapnya, lihat Mencatat peristiwa data untuk menyimpan data peristiwa dengan AWS CLI.

Tabel berikut mencantumkan jenis AWS AppConfig sumber daya. 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 API atau. AWS CLI CloudTrail CloudTrailKolom API Data yang dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	API data masuk CloudTrail ke*
AWS AppConfig	`AWS::AppConfig::Configuration`	• GetLatestConfiguration • StartConfigurationSession

* Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada`eventName`,`readOnly`, dan `resources.ARN` bidang untuk mencatat hanya peristiwa yang penting bagi Anda. Untuk informasi lebih lanjut tentang bidang ini, lihat AdvancedFieldSelector.

AWS AppConfigacara manajemen di CloudTrail

Acara manajemen memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di AWS akun Anda. Ini juga dikenal sebagai operasi pesawat kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

AWS AppConfigmencatat semua operasi pesawat AWS AppConfig kontrol sebagai peristiwa manajemen. Untuk daftar operasi bidang AWS AppConfig kontrol yang AWS AppConfig masuk ke log CloudTrail, lihat Referensi AWS AppConfig API.

Memahami entri file log AWS AppConfig

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta,

tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan StartConfigurationSessiontindakan.


```
{
      "eventVersion": "1.09",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Administrator",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {},
          "attributes": {
            "creationDate": "2024-01-11T14:37:02Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2024-01-11T14:45:15Z",
      "eventSource": "appconfig.amazonaws.com",
      "eventName": "StartConfigurationSession",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "Boto3/1.34.11 md/Botocore#1.34.11 ua/2.0 os/macos#22.6.0
 md/arch#x86_64 lang/python#3.11.4 md/pyimpl#CPython cfg/retry-mode#legacy
 Botocore/1.34.11",
      "requestParameters": {
        "applicationIdentifier": "rrfexample",
        "environmentIdentifier": "mexampleqe0",
        "configurationProfileIdentifier": "3eexampleu1"
      },
      "responseElements": null,
      "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
      "readOnly": false,
      "resources": [
        {
          "accountId": "123456789012",
```

```
          "type": "AWS::AppConfig::Configuration",
          "ARN": "arn:aws:appconfig:us-east-1:123456789012:application/rrfexample/
environment/mexampleqe0/configuration/3eexampleu1"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "123456789012",
      "eventCategory": "Data",
      "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "appconfigdata.us-east-1.amazonaws.com"
      }
    }
```

Metrik pencatatan untuk panggilan pesawat AWS AppConfig data

Jika Anda mengonfigurasi AWS CloudTrail untuk mencatat peristiwa AWS AppConfig data, Anda dapat mengaktifkan CloudWatch Log Amazon untuk mencatat metrik panggilan ke bidang AWS AppConfig data. Anda kemudian dapat mencari dan memfilter data CloudWatch log di Log dengan membuat satu atau beberapa filter metrik. Filter metrik menentukan istilah dan pola yang harus dicari dalam data log saat dikirim ke CloudWatch Log. CloudWatch Log menggunakan filter metrik untuk mengubah data log menjadi CloudWatch metrik numerik. Anda dapat membuat grafik metrik atau mengonfigurasinya dengan alarm.

Sebelum Anda memulai

Aktifkan pencatatan peristiwa AWS AppConfig data diAWS CloudTrail. Prosedur berikut menjelaskan cara mengaktifkan pencatatan metrik untuk AWS AppConfigjejak yang ada di CloudTrail. Untuk informasi tentang cara mengaktifkan CloudTrail pencatatan untuk panggilan paket AWS AppConfig data, lihatAWS AppConfigperistiwa data di CloudTrail.

Gunakan prosedur berikut untuk mengaktifkan CloudWatch Log untuk mencatat metrik untuk panggilan ke bidang AWS AppConfig data.

Untuk mengaktifkan CloudWatch Log untuk mencatat metrik untuk panggilan ke bidang AWS AppConfig data

1. Buka CloudTrail konsol di https://console.aws.amazon.com/cloudtrail/.

2. Di dasbor, pilih AWS AppConfig jejak Anda.
3. Di bagian CloudWatch Log, pilih Edit.
4. Pilih Diaktifkan.
5. Untuk nama grup Log, tinggalkan nama default atau masukkan nama. Catat namanya. Anda akan memilih grup log di konsol CloudWatch Log nanti.
6. Untuk nama Peran, masukkan nama.
7. Pilih Simpan perubahan.

Gunakan prosedur berikut untuk membuat metrik dan filter metrik untuk AWS AppConfig di CloudWatch Log. Prosedur ini menjelaskan cara membuat filter metrik untuk panggilan oleh `operation` dan (opsional) panggilan oleh `operation` dan`Amazon Resource Name (ARN)`.

Untuk membuat metrik dan filter metrik untuk AWS AppConfig di CloudWatch Log

1. Buka CloudWatch konsol dihttps://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, pilih Log, lalu pilih Grup log.
3. Pilih kotak centang di samping grup AWS AppConfig log.
4. Pilih Tindakan, lalu pilih Buat filter metrik.
5. Untuk nama Filter, masukkan nama.
6. Untuk pola Filter, masukkan yang berikut ini:

```
{ $.eventSource = "appconfig.amazonaws.com" }
```

7. (Opsional) Di bagian Pola uji, pilih grup log Anda dari daftar Pilih data log untuk diuji. Jika CloudTrail belum mencatat panggilan apa pun, Anda dapat melewati langkah ini.
8. Pilih Berikutnya.
9. Untuk namespace Metric, masukkan. **`AWS AppConfig`**
10. Untuk Nama metrik, masukkan **`Calls`**.
11. Untuk Metric value (Nilai metrik), masukkan **`1`**.
12. Lewati nilai Default dan Unit.
13. Untuk nama Dimensi, masukkan**`operation`**.
14. Untuk nilai Dimensi, masukkan**`$.eventName`**.

(Opsional) Anda dapat memasukkan dimensi kedua yang menyertakan Nama Sumber Daya Amazon (ARN) yang melakukan panggilan. Untuk menambahkan dimensi kedua, untuk nama Dimensi, masukkan**`resource`**. Untuk nilai Dimensi, masukkan**`$.resources[0].ARN`**.

Pilih Berikutnya.

15. Tinjau detail filter dan Buat filter metrik.

(Opsional) Anda dapat mengulangi prosedur ini untuk membuat filter metrik baru untuk kode kesalahan tertentu seperti AccessDenied. Jika Anda melakukannya, masukkan detail berikut:

1. Untuk nama Filter, masukkan nama.
2. Untuk pola Filter, masukkan yang berikut ini:

```
{ $.errorCode = "codename" }
```

Sebagai contoh

```
{ $.errorCode = "AccessDenied" }
```

3. Untuk namespace Metric, masukkan. **`AWS AppConfig`**
4. Untuk Nama metrik, masukkan **`Errors`**.
5. Untuk Metric value (Nilai metrik), masukkan **`1`**.
6. Untuk nilai Default, masukkan nol (0).
7. Lewati Unit, Dimensi, dan Alarm.

Setelah CloudTrail mencatat panggilan API, Anda dapat melihat metrik di CloudWatch. Untuk informasi selengkapnya, lihat Melihat metrik dan log Anda di konsol di Panduan CloudWatch Pengguna Amazon. Untuk informasi tentang cara menemukan metrik yang Anda buat, lihat Mencari metrik yang tersedia.

> **Note**
>
> Jika Anda mengatur metrik kesalahan tanpa dimensi, seperti yang dijelaskan di sini, Anda dapat melihat metrik tersebut di halaman Metrik tanpa dimensi.

Membuat alarm untuk CloudWatch metrik

Setelah membuat metrik, Anda dapat membuat alarm metrik di. CloudWatch Misalnya, Anda dapat membuat alarm untuk metrik AWS AppConfigpanggilan yang Anda buat di prosedur sebelumnya. Secara khusus, Anda dapat membuat alarm untuk panggilan ke tindakan AWS AppConfig `StartConfigurationSession` API yang melampaui ambang batas. Untuk informasi tentang cara membuat alarm untuk metrik, lihat Membuat CloudWatch alarm berdasarkan ambang batas statis di Panduan CloudWatch Pengguna Amazon. Untuk informasi tentang batas default untuk panggilan ke bidang AWS AppConfig data, lihat Batas default bidang data di Referensi Umum Amazon Web Services.

AWS AppConfig Riwayat dokumen Panduan Pengguna

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir AWS AppConfig.

Versi API saat ini: 2019-10-09

Perubahan	Deskripsi	Tanggal
AWS AppConfig sampel ekstensi kustom	Panduan: Membuat topik AWS AppConfig ekstensi kustom sekarang mencakup tautan ke ekstensi contoh berikut di: GitHub • Contoh ekstensi yang mencegah penerapan dengan kalender `blocked day` moratorium menggunakan Systems Manager Change Calendar • Contoh ekstensi yang mencegah rahasia bocor ke data konfigurasi menggunakan git-secret • Ekstensi sampel yang mencegah informasi identitas pribadi (PII) bocor ke data konfigurasi menggunakan Amazon Comprehend	Februari 28, 2024
Topik baru: Logging panggilan AWS AppConfig API menggunakan AWS CloudTrail	AWS AppConfig terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di	Januari 18, 2024

	AWS AppConfig. CloudTrail menangkap semua panggilan API untuk AWS AppConfig sebagai peristiwa. Topik baru ini menyediakan konten AWS AppConfig khusus daripada menautkan ke konten yang sesuai di AWS Systems Manager Panduan Pengguna. Untuk informasi selengkapnya, lihat Logging panggilan AWS AppConfig API menggunakan AWS CloudTrail.	
AWS AppConfig sekarang mendukung AWS PrivateLink	Anda dapat menggunak an AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan. AWS AppConfig Anda dapat mengakses AWS AppConfig 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. AWS AppConfig Untuk informasi selengkapnya, lihat Akses AWS AppConfig menggunak an titik akhir antarmuka (AWS PrivateLink).	6 Desember 2023

Fitur pengambilan AWS AppConfig Agen tambahan dan mode pengembangan lokal baru	AWS AppConfig Agen menawarkan fitur tambahan berikut untuk membantu Anda mengambil konfigurasi untuk aplikasi Anda. Fitur pengambilan tambahan • Pengambilan multi-akun: Gunakan AWS AppConfig Agen dari primer atau pengambilan Akun AWS untuk mengambil data konfigurasi dari beberapa akun vendor. • Tulis salinan konfigurasi ke disk: Gunakan AWS AppConfig Agen untuk menulis data konfigurasi ke disk. Fitur ini memungkin kan pelanggan dengan aplikasi yang membaca data konfigurasi dari disk untuk diintegrasikan AWS AppConfig. Note Konfigurasi tulis ke disk tidak dirancang sebagai fitur cadangan konfigurasi. AWS AppConfig Agen tidak membaca dari file konfigurasi yang disalin ke disk. Jika	1 Desember 2023

Anda ingin mencadangkan konfigurasi ke disk, lihat variabel `BACKUP_DIRECTORY` dan `PRELOAD_BACKUP` lingkungan untuk Menggunakan AWS AppConfig Agen dengan Amazon EC2 atau AWS AppConfig Menggunakan Agen dengan Amazon ECS dan Amazon EKS.

Mode pengembangan lokal

AWS AppConfig Agen mendukung mode pengembangan lokal. Jika Anda mengaktifkan mode pengembangan lokal, agen membaca data konfigurasi dari direktori tertentu pada disk. Itu tidak mengambil data konfigurasi dari AWS AppConfig. Anda dapat mensimulasikan penerapan konfigurasi dengan memperbarui file di direktori yang ditentukan. Kami merekomendasikan mode pengembangan lokal untuk kasus penggunaan berikut:

- Uji versi konfigurasi yang berbeda sebelum

	menerapkannya menggunakan AWS AppConfig. • Uji opsi konfigurasi yang berbeda untuk fitur baru sebelum melakukan perubahan ke repositori kode Anda. • Uji skenario konfigurasi yang berbeda untuk memverifikasi bahwa mereka berfungsi seperti yang diharapkan.	
Topik sampel kode baru	Menambahkan topik sampel kode baru ke panduan ini. Topik ini mencakup contoh di Java, Python, dan JavaScript untuk melakukan enam tindakan umum secara terprogram. AWS AppConfig	17 November 2023
Daftar isi yang direvisi untuk mencerminkan AWS AppConfig alur kerja dengan lebih baik	Konten dalam panduan pengguna ini sekarang dikelompokkan di bawah judul Membuat, Menerapkan, Mengambil, dan Memperluas alur kerja. Organisasi ini lebih mencerminkan alur kerja untuk digunakan AWS AppConfig dan bertujuan untuk membantu membuat konten lebih mudah ditemukan.	7 November 2023

Referensi muatan ditambahkan	Fungsi Membuat Lambda untuk topik AWS AppConfig ekstensi kustom sekarang menyertakan referensi payload permintaan dan respons.	7 November 2023
Strategi penyebaran baru AWS yang telah ditentukan	AWS AppConfig sekarang menawarkan dan merekomendasikan strategi penyebaran yang `AppConfig.Linear20PercentEvery6Minutes` telah ditentukan. Untuk informasi selengkapnya, lihat Strategi penerapan yang telah ditentukan sebelumnya.	11 Agustus 2023
AWS AppConfig integrasi dengan Amazon EC2	Anda dapat berintegrasi AWS AppConfig dengan aplikasi yang berjalan di instans Amazon Elastic Compute Cloud (Amazon EC2) Linux dengan menggunakan Agen. AWS AppConfig Agen mendukung arsitektur x86_64 dan ARM64 untuk Amazon EC2. Untuk informasi selengkapnya, lihat AWS AppConfig integrasi dengan Amazon EC2.	Juli 20, 2023

AWS CloudFormation dukungan untuk AWS AppConfig sumber daya baru dan contoh tanda fitur	AWS CloudFormation sekarang mendukung AWS::AppConfig::Extensiondan AWS::AppConfig::ExtensionAssociation sumber daya untuk membantu Anda memulai dengan AWS AppConfig ekstensi. Sumber daya AWS::AppConfig::ConfigurationProfiledan AWS::AppConfig::HostedConfigurationVersi sekarang menyertakan contoh untuk membuat profil konfigurasi flag fitur di toko konfigurasi yang AWS AppConfig dihosting.	12 April 2023

AWS AppConfig Integrasi dengan AWS Secrets Manager	AWS AppConfig terintegrasi dengan AWS Secrets Manager. Secrets Manager membantu Anda mengenkripsi, menyimpan, dan mengambil kredensyal dengan aman untuk database dan layanan lainnya. Alih-alih membuat hardcoding kredensyal di aplikasi Anda, Anda dapat melakukan panggilan ke Secrets Manager untuk mengambil kredensyal Anda kapan pun diperlukan. Secrets Manager membantu Anda melindungi akses ke sumber daya dan data TI Anda dengan memungkinkan Anda untuk memutar dan mengelola akses ke rahasia Anda. Saat Anda membuat profil konfigurasi bentuk bebas, Anda dapat memilih Secrets Manager sebagai sumber data konfigurasi Anda. Anda harus bergabung dengan Secrets Manager dan membuat rahasia sebelum membuat profil konfigurasi. Untuk informasi selengkapnya tentang Secrets Manager, lihat Apa itu AWS Secrets Manager? dalam AWS Secrets Manager User Guide. Untuk informasi tentang	2 Februari 2023

membuat profil konfigurasi, lihat Membuat profil konfigurasi bentuk bebas.

AWS AppConfig integrasi dengan Amazon ECS dan Amazon EKS	Anda dapat berintegrasi AWS AppConfig dengan Amazon Elastic Container Service (Amazon ECS) dan Amazon Elastic Kubernetes Service (Amazon EKS) dengan menggunakan agen. AWS AppConfig Agen berfungsi sebagai wadah sespan yang berjalan bersama aplikasi penampung Amazon ECS dan Amazon EKS Anda. Agen meningkatkan pemrosesa n dan manajemen aplikasi kontainer dengan cara berikut: • Agen memanggil AWS AppConfig atas nama Anda dengan menggunakan peran AWS Identity and Access Management (IAM) dan mengelola cache lokal data konfigurasi. Dengan menarik data konfigurasi dari cache lokal, aplikasi Anda memerlukan lebih sedikit pembaruan kode untuk mengelola data konfigurasi, mengambil data konfigurasi dalam milidetik , dan tidak terpengaruh oleh masalah jaringan yang dapat menggangg u panggilan untuk data tersebut.	Desember 2, 2022

- Agen menawarkan pengalaman asli untuk mengambil dan menyelesaikan flag AWS AppConfig fitur.
- Di luar kotak, agen menyediakan praktik terbaik untuk strategi caching, interval polling, dan ketersediaan data konfigurasi lokal sambil melacak token konfigurasi yang diperlukan untuk panggilan layanan berikutnya.
- Saat berjalan di latar belakang, agen secara berkala melakukan polling bidang AWS AppConfig data untuk pembaruan data konfigurasi. Aplikasi kontainer Anda dapat mengambil data dengan menghubungkan ke localhost pada port 2772 (nilai port default yang dapat disesuaikan) dan memanggil HTTP GET untuk mengambil data.
- AWS AppConfig Agen memperbarui data konfigurasi dalam wadah Anda tanpa harus memulai ulang atau mendaur ulang kontainer tersebut.

Untuk informasi selengkapnya, lihat AWS AppConfig integrasi dengan Amazon ECS dan Amazon EKS.

Ekstensi baru: AWS AppConfig ekstensi untuk CloudWatch Evidently	Anda dapat menggunakan Amazon CloudWatch Terbukti untuk memvalidasi fitur baru dengan aman dengan menyajikannya ke persentase tertentu dari pengguna Anda saat Anda meluncurkan fitur tersebut. Anda dapat memantau performa fitur baru untuk membantu Anda memutuskan kapan harus menaikkan lalu lintas ke para pengguna Anda. Hal ini akan membantu Anda mengurangi risiko dan mengidentifikasi konsekuensi yang tidak Anda inginkan sebelum Anda meluncurkan fitur tersebut sepenuhnya. Anda juga dapat melakukan percobaan-percobaan A/B untuk membuat keputusan desain fitur berdasarkan bukti dan data. AWS AppConfig Ekstensi untuk CloudWatch Evidently memungkinkan aplikasi Anda untuk menetapkan variasi ke sesi pengguna secara lokal, bukan dengan memanggil operasi. EvaluateFeature Sesi lokal mengurangi risiko latensi dan ketersediaan yang menyertai panggilan API. Untuk informasi tentang	13 September 2022

	cara mengonfigurasi dan menggunakan ekstensi, lihat Melakukan peluncuran dan eksperimen A/B dengan CloudWatch Evidently in the Amazon CloudWatch User Guide.	
Pengakhiran tindakan API `GetConfiguration`	Pada 18 November 2021, AWS AppConfig merilis layanan pesawat data baru. Layanan ini mengganti kan proses pengambilan data konfigurasi sebelumny a dengan menggunak an tindakan `GetConfig uration` API. Layanan data plane menggunakan dua tindakan API baru, StartConf igurationSessiondan GetLatest Configuration. Layanan pesawat data juga menggunak an titik akhir baru. Untuk informasi selengkapnya, lihat Tentang layanan pesawat AWS AppConfig data.	13 September 2022
Versi baru dari ekstensi AWS AppConfig Agen Lambda	Versi 2.0.122 dari ekstensi Agen AWS AppConfig Lambda sekarang tersedia. Ekstensi baru menggunakan Nama Sumber Daya Amazon (ARN) yang berbeda. Untuk informasi selengkapnya, lihat AWS AppConfig Catatan rilis ekstensi Agen Lambda.	23 Agustus 2022

Peluncuran AWS AppConfig ekstensi	Ekstensi menambah kemampuan Anda untuk menyuntikkan logika atau perilaku pada titik yang berbeda selama AWS AppConfig alur kerja membuat atau menerapkan konfigura si. Anda dapat menggunak an ekstensi AWS-authored atau membuat sendiri. Untuk informasi selengkapnya, lihat Bekerja dengan AWS AppConfig ekstensi.	12 Juli 2022
Versi baru dari ekstensi AWS AppConfig Agen Lambda	Versi 2.0.58 dari ekstensi Agen AWS AppConfig Lambda sekarang tersedia. Ekstensi baru menggunakan Nama Sumber Daya Amazon (ARN) yang berbeda. Untuk informasi selengkapnya, lihat Versi ekstensi AWS AppConfig Lambda yang tersedia.	Mei 3, 2022

AWS AppConfig integrasi dengan Atlassian Jira	Mengintegrasikan dengan Atlassian Jira memungkinkan AWS AppConfig untuk membuat dan memperbarui masalah di konsol Atlassian setiap kali Anda membuat perubahan pada bendera fitur di Anda untuk yang ditentukan. Akun AWS Wilayah AWS Setiap masalah Jira mencakup nama bendera, ID aplikasi, ID profil konfigurasi, dan nilai bendera. Setelah Anda memperbarui, menyimpan, dan menerapkan perubahan bendera Anda, Jira memperbarui masalah yang ada dengan detail perubahan . Untuk informasi lebih lanjut, lihat AWS AppConfig integrasi dengan Atlassian Jira.	7 April 2022

Ketersediaan umum bendera fitur dan dukungan ekstensi Lambda untuk prosesor ARM64 (Graviton2)	Dengan flag AWS AppConfig fitur, Anda dapat mengembangkan fitur baru dan menyebarkannya ke produksi sambil menyembunyikan fitur dari pengguna. Anda mulai dengan menambahkan bendera AWS AppConfig sebagai data konfigurasi. Setelah fitur siap dirilis, Anda dapat memperbarui data konfigurasi bendera tanpa menerapkan kode apa pun. Fitur ini meningkatkan keamanan lingkungan dev-ops Anda karena Anda tidak perlu menerapkan kode baru untuk merilis fitur tersebut. Untuk informasi selengkapnya, lihat Membuat profil konfigurasi tanda fitur. Ketersediaan umum flag fitur AWS AppConfig termasuk penyempurnaan berikut: • Konsol menyertakan opsi untuk menunjuk bendera sebagai bendera jangka pendek. Anda dapat memfilter dan mengurutkan daftar bendera pada bendera jangka pendek. • Untuk pelanggan yang menggunakan flag fitur AWS Lambda, ekstensi Lambda baru memungkinkan Anda memanggil flag	15 Maret 2022

	fitur individual dengan menggunakan titik akhir HTTP. Untuk informasi selengkapnya, lihat Mengambil satu atau beberapa flag dari konfigurasi flag fitur.	
	Pembaruan ini juga menyediakan dukungan untuk AWS Lambda ekstensi yang dikembangkan untuk prosesor ARM64 (Graviton2). Untuk informasi selengkapnya, lihat Versi ekstensi AWS AppConfig Lambda yang tersedia.	
Tindakan GetConfiguration API tidak digunakan lagi	Tindakan `GetConfiguration` API tidak digunakan lagi. Panggilan untuk menerima data konfigurasi harus menggunakan `GetLatestConfiguration` API `StartConfigurationSession` dan sebagai gantinya. Untuk informasi selengkapnya tentang API ini dan cara menggunakannya, lihat Mengambil konfigurasi.	28 Januari 2022

ARN wilayah baru untuk ekstensi Lambda AWS AppConfig	AWS AppConfig Ekstensi Lambda tersedia di wilayah Asia Pasifik (Osaka) yang baru. Nama Sumber Daya Amazon (ARN) diperluka n untuk membuat Lambda di wilayah tersebut. Untuk informasi lebih lanjut tentang ARN wilayah Asia Pasifik (Osaka), lihat Menambahk an ekstensi Lambda AWS AppConfig.	4 Maret 2021
AWS AppConfig Ekstensi Lambda	Jika Anda menggunakan AWS AppConfig untuk mengelola konfigurasi untuk fungsi Lambda, maka kami sarankan Anda menambahkan ekstensi AWS AppConfig Lambda. Ekstensi ini mencakup praktik terbaik yang menyederh anakan penggunaan AWS AppConfig sekaligus mengurangi biaya. Pengurang an biaya dihasilkan dari lebih sedikit panggilan API ke AWS AppConfig layanan dan, secara terpisah, mengurang i biaya dari waktu pemrosesa n fungsi Lambda yang lebih pendek. Untuk informasi selengkapnya, lihat AWS AppConfig integrasi dengan ekstensi Lambda.	8 Oktober 2020

Bagian baru	Menambahkan bagian baru yang memberikan instruksi untuk pengaturan AWS AppConfig. Untuk informasi selengkapnya, lihat Menyiapkan AWS AppConfig.	30 September 2020
Ditambahkan prosedur commandline	Prosedur dalam panduan pengguna ini sekarang mencakup langkah-langkah baris perintah untuk AWS Command Line Interface (AWS CLI) dan Alat untuk Windows. PowerShell Untuk informasi selengkapnya, lihat Bekerja dengan AWS AppConfig.	30 September 2020
Peluncuran panduan AWS AppConfig pengguna	Gunakan AWS AppConfig, kemampuan AWS Systems Manager, untuk membuat, mengelola, dan dengan cepat menyebarkan konfigurasi aplikasi. AWS AppConfig mendukung penerapan terkontrol ke aplikasi dari berbagai ukuran dan mencakup pemeriksaan dan pemantauan validasi bawaan. Anda dapat menggunakan AWS AppConfig dengan aplikasi yang dihosting di instans EC2, wadah AWS Lambda, aplikasi seluler, atau perangkat IoT.	31 Juli 2020

Daftar istilah AWS

Untuk terminologi AWS terbaru, lihat Daftar istilah AWS di Referensi Glosarium AWS.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.