aws

Panduan Pengguna

AWS AppConfig

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS AppConfig: Panduan Pengguna

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

Apa itu AWS AppConfig? 1
Memulai dengan AWS AppConfig 1
AWS AppConfig kasus penggunaan 1
Ikhtisar manfaat 2
Bagaimana cara AWS AppConfig kerja 3
Harga untuk AWS AppConfig 5
AWS AppConfig kuota 5
Sumber daya tambahan 5
Blog 5
SDKs 6
Menyiapkan AWS AppConfig 7
Mendaftar untuk Akun AWS 7
Buat pengguna dengan akses administratif 7
Memberikan akses programatis 9
Memahami IPv6 dukungan 11
Konfigurasikan izin untuk rollback otomatis 11
Langkah 1: Buat kebijakan izin untuk rollback berdasarkan alarm CloudWatch 12
Langkah 2: Buat peran IAM untuk rollback berdasarkan alarm CloudWatch 13
Langkah 3: Tambahkan hubungan kepercayaan 14
Membuat 15
Memahami peran IAM profil konfigurasi 16
Membuat namespace 19
Membuat AWS AppConfig aplikasi (konsol) 19
Membuat AWS AppConfig aplikasi (baris perintah) 20
Membuat lingkungan 22
Menciptakan AWS AppConfig lingkungan (konsol) 22
Menciptakan AWS AppConfig lingkungan (baris perintah) 23
Membuat profil konfigurasi di AWS AppConfig 25
Membuat profil konfigurasi bendera fitur 29
Membuat profil konfigurasi formulir gratis 61
Membuat profil konfigurasi untuk sumber data non-native 76
Men-deploy 79
Bekerja dengan strategi penyebaran 80
Menggunakan strategi penyebaran yang telah ditentukan 83

Buat strategi deployment 84
Men-deploy konfigurasi 89
Menerapkan konfigurasi (konsol) 90
Menyebarkan konfigurasi (baris perintah) 90
Menyebarkan dengan CodePipeline 94
Bagaimana integrasi bekerja 95
Mengembalikan konfigurasi 95
Mengambil 98
Apa itu AWS AppConfig Agen? 99
Cara menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi 101
Menggunakan AWS AppConfig agen dengan AWS Lambda 101
Menggunakan AWS AppConfig Agen dengan Amazon EC2 dan mesin lokal 212
Menggunakan AWS AppConfig Agen dengan Amazon ECS dan Amazon EKS 231
Mengambil bendera fitur 252
Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan 255
Menghasilkan klien menggunakan spesifikasi OpenAPI 266
Bekerja dengan mode pengembangan lokal AWS AppConfig Agen 269
Pertimbangan penggunaan browser dan seluler 273
Data konfigurasi dan pengambilan bendera 274
Otentikasi dan Amazon Cognito 274
Pembuatan cache 275
Segmentasi 276
Bandwidth (kasus penggunaan seluler) 276
Kasus penggunaan bendera tambahan 277
Mengambil data konfigurasi tanpa Agen AWS AppConfig 277
(Contoh) Mengambil konfigurasi dengan memanggil AWS AppConfig APIs 279
Memperluas alur kerja AWS AppConfig 281
Memahami AWS AppConfig ekstensi 281
Langkah 1: Tentukan apa yang ingin Anda lakukan dengan ekstensi 282
Langkah 2: Tentukan kapan Anda ingin ekstensi berjalan 283
Langkah 3: Buat asosiasi ekstensi 284
Langkah 4: Menyebarkan konfigurasi dan memverifikasi tindakan ekstensi dilakukan 285
Bekerja dengan ekstensi yang AWS ditulis 285
Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon EventBridge 286
Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon SNS 288
Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon SQS 291

Menggunakan ekstensi Jira 293
Walkthrough: Membuat ekstensi khusus AWS AppConfig 298
Langkah 1: Buat fungsi Lambda untuk ekstensi khusus AWS AppConfig 300
Langkah 2: Konfigurasikan izin untuk ekstensi khusus AWS AppConfig 306
Langkah 3: Buat AWS AppConfig ekstensi khusus 308
Langkah 4: Buat asosiasi ekstensi untuk AWS AppConfig ekstensi khusus 312
Sampel Kode 314
Membuat atau memperbarui konfigurasi bentuk bebas yang disimpan di toko konfigurasi yang dihosting 314
Membuat profil konfigurasi untuk rahasia yang disimpan di Secrets Manager 317
Menerapkan profil konfigurasi 318
Menggunakan AWS AppConfig Agen untuk membaca profil konfigurasi bentuk bebas 323
Menggunakan AWS AppConfig Agen untuk membaca bendera fitur tertentu 325
Menggunakan AWS AppConfig Agen untuk mengambil bendera fitur dengan varian 326
Menggunakan tindakan GetLatestConfiguration API untuk membaca profil konfigurasi bentuk bebas 328
Membersihkan lingkungan Anda 339
Perlindungan penghapusan 345
Melewati atau memaksa pemeriksaan perlindungan penghapusan 346
Keamanan 349
Terapkan akses hak akses paling rendah 349
Enkripsi data saat istirahat untuk AWS AppConfig 350
AWS PrivateLink 355
Pertimbangan-pertimbangan 355
Membuat sebuah titik akhir antarmuka 356
Membuat kebijakan titik akhir 356
Rotasi kunci Secrets Manager 357
Menyiapkan rotasi otomatis rahasia Secrets Manager yang digunakan oleh AWS AppConfig 357
Memantau 360
CloudTrail log 361
AWS AppConfig peristiwa data di CloudTrail 362
AWS AppConfig acara manajemen di CloudTrail 364
AWS AppConfig contoh acara 364
Metrik pencatatan untuk panggilan pesawat AWS AppConfig data 366
Membuat alarm untuk CloudWatch metrik 368

Memantau penyebaran untuk rollback otomatis 368
Metrik yang disarankan untuk memantau rollback otomatis 370
Riwayat dokumen 377
..... cdviii

Apa itu AWS AppConfig?

AWS AppConfig flag fitur dan konfigurasi dinamis membantu pembangun perangkat lunak dengan cepat dan aman menyesuaikan perilaku aplikasi di lingkungan produksi tanpa penerapan kode penuh. AWS AppConfig mempercepat 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.

Memulai dengan AWS AppConfig

Video berikut dapat membantu Anda memahami kemampuan AWS AppConfig.

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

AWS AppConfig kasus penggunaan

AWS AppConfig mendukung spektrum kasus penggunaan yang luas:

- 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.

Ikhtisar manfaat

Ikhtisar singkat berikut menguraikan manfaat penggunaan AWS AppConfig.

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 CI/CD pipeline 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 AppConfig menawarkan fitur keamanan berikut untuk membantu Anda menghindari mengaktifkan flag fitur atau memperbarui data konfigurasi yang dapat menyebabkan kegagalan aplikasi.

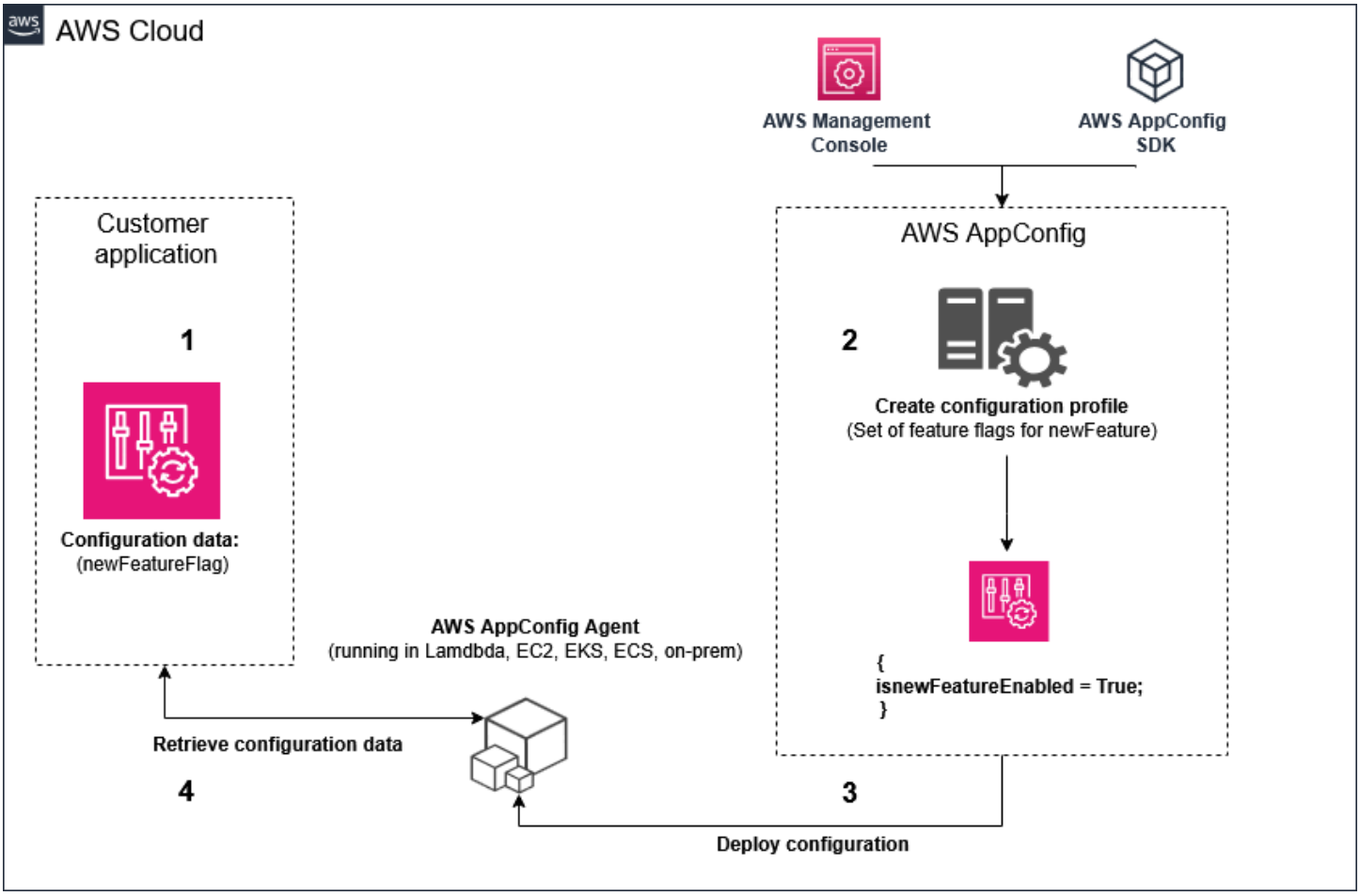
- Validator: Validator memastikan bahwa data konfigurasi Anda benar secara sintaksis dan semantik sebelum menerapkan perubahan ke lingkungan produksi.
- Strategi penyebaran: Strategi penerapan memungkinkan Anda untuk secara perlahan melepaskan perubahan ke lingkungan produksi selama beberapa menit atau 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 AppConfig terintegrasi dengan AWS Identity and Access Management (IAM) untuk menyediakan akses berbasis peran halus ke layanan. AWS AppConfig juga 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.

Bagaimana cara AWS AppConfig kerja

Bagian ini memberikan deskripsi tingkat tinggi tentang cara AWS AppConfig kerja.



1. Identifikasi nilai konfigurasi dalam kode yang ingin Anda kelola AWS AppConfig

   Sebelum Anda membuat profil konfigurasi di AWS AppConfig, 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. Jenis konfigurasi ini sering berubah dan dapat menyebabkan masalah jika tidak benar.

   Jika data konfigurasi Anda sudah ada di cloud, misalnya di Parameter Store atau Amazon S3, Anda dapat memanfaatkan fitur AWS AppConfig validasi, penyebaran, dan ekstensi untuk lebih merampingkan pengelolaan data konfigurasi.

2. Buat 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 profil. AWS AppConfig mendukung dua jenis profil konfigurasi: bendera fitur dan konfigurasi bentuk bebas. Kedua jenis dapat mengurangi risiko dan kompleksitas pengembangan dan penyebaran perangkat lunak dengan memisahkan rilis fitur dari penerapan kode. Mereka juga memungkinkan pengiriman berkelanjutan dan mitigasi risiko melalui peluncuran bertahap. Selain itu, flag fitur memungkinkan pengujian dalam produksi dengan pengguna nyata, sementara konfigurasi bentuk bebas memungkinkan Anda untuk mengambil data konfigurasi dari layanan lain. AWS Kedua jenis profil memungkinkan iterasi, eksperimen, personalisasi, dan manajemen siklus hidup perangkat lunak yang lebih cepat. Untuk informasi selengkapnya tentang membuat profil konfigurasi, lihatMembuat profil konfigurasi 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 akan kembali ke data konfigurasi sebelumnya.

Saat Anda membuat profil konfigurasi, Anda juga membuat aplikasi di AWS AppConfig. Aplikasi hanyalah namespace atau konstruksi organisasi seperti folder.

3. Menyebarkan data konfigurasi

Saat Anda memulai penerapan, AWS AppConfig lakukan tugas-tugas berikut:

1. Mengambil data konfigurasi dari penyimpanan data yang mendasarinya dengan menggunakan nama jalur lokasi di profil konfigurasi.
2. Memverifikasi data konfigurasi secara sintaksis dan semantik benar dengan menggunakan validator yang Anda tentukan saat membuat profil konfigurasi.
3. Mengirim salinan data ke AWS AppConfig Agen untuk dibaca oleh aplikasi Anda. Salinan ini disebut data yang digunakan.

Untuk informasi selengkapnya tentang penerapan konfigurasi, lihatMenerapkan flag fitur dan data konfigurasi di AWS AppConfig.

4. Ambil konfigurasi

Untuk mengambil data, aplikasi Anda membuat panggilan HTTP ke server localhost tempat AWS AppConfig Agen telah menyimpan salinan lokal dari data konfigurasi yang Anda gunakan.

Mengambil data adalah peristiwa terukur. AWS AppConfig Agen mendukung beberapa kasus penggunaan, seperti yang dijelaskan dalamCara menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi.

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.

Untuk informasi selengkapnya tentang mengambil konfigurasi, lihatMengambil flag fitur dan data konfigurasi di AWS AppConfig.

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, silakan lihat Harga AWS Systems Manager.

AWS AppConfig kuota

Anda dapat melihat informasi tentang AWS AppConfig titik akhir dan kuota layanan di. Referensi Umum Amazon Web

> **Note**
>
> AWS AppConfig adalah kemampuan AWS Systems Manager.

Untuk informasi tentang kuota untuk layanan yang menyimpan AWS AppConfig konfigurasi, lihat. Memahami kuota dan batasan penyimpanan konfigurasi

Sumber daya tambahan

Sumber daya berikut dapat membantu Anda mempelajari lebih lanjut AWS AppConfig.

Blog

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

- Mengapa Anda harus menggunakan AWS AppConfig
- Lepaskan kekuatan bendera fitur dengan AWS AppConfig
- Menggunakan bendera AWS AppConfig fitur
- Praktik terbaik untuk memvalidasi flag AWS AppConfig fitur dan data konfigurasi

SDKs

Untuk informasi tentang AWS AppConfig bahasa tertentu SDKs, lihat sumber daya berikut:

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

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/pendaftaran.
2. Ikuti petunjuk online.

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

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

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

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

1. Masuk ke Konsol Manajemen AWSsebagai 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. Konsol Manajemen AWS 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
IAM	(Disarankan) Gunakan kredenal konsol sebagai kredensil sementara untuk menandatangani permintaan terprogram ke,, atau. AWS CLI AWS SDKs AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk itu AWS CLI, lihat Login untuk pengembangan AWS lokal di Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDKs, lihat Login untuk pengembangan AWS lokal di Panduan Referensi Alat AWS SDKs dan Alat.
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	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.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		• Untuk AWS SDKs, alat, dan AWS APIs, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi Alat AWS SDKs dan Alat.
IAM	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensi jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDKs dan alat, lihat Mengautentikasi menggunakan kredensil jangka panjang di Panduan Referensi Alat AWS SDKs dan Alat. • Untuk AWS APIs, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Memahami IPv6 dukungan

Semua dukungan AWS AppConfig APIs penuh IPv4 dan IPv6 panggilan.

Bidang kontrol APIs

Gunakan titik akhir berikut untuk IPv4 dan IPv6 dual-stack panggilan ke bidang kontrol:

```
appconfig.Region.api.aws
```

Misalnya: appconfig.us-east-1.api.aws

IPv4 Hanya untuk, gunakan URL berikut:

```
appconfig.Region.amazonaws.com
```

Pesawat data APIs

Untuk panggilan dual-stack ke bidang data, gunakan titik akhir berikut:

```
appconfigdata.Region.api.aws
```

Misalnya: appconfig.us-east-1.api.aws

IPv4 Hanya untuk, gunakan URL berikut:

```
appconfigdata.Region.amazonaws.com
```

> **Note**
>
> Untuk informasi selengkapnya, lihat AWS AppConfig titik akhir dan kuota di. Referensi Umum AWS

Konfigurasikan izin untuk rollback otomatis

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. Prosedur ini opsional, tetapi sangat dianjurkan.

> **Note**
>
> Perhatikan informasi berikut.
>
> - Peran IAM harus milik akun saat ini. Secara default, hanya AWS AppConfig dapat memantau alarm yang dimiliki oleh akun saat ini.
> - Untuk informasi tentang metrik yang akan dipantau dan cara mengonfigurasi AWS AppConfig untuk rollback otomatis, lihat. Memantau penyebaran untuk rollback otomatis

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.

JSON

```
{
        "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 SSMCloudWatchAlarmDiscoveryPolicy.
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:

   JSON

```
{
  "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. Ciptakan 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, seperti`AWS Lambda functions`,,`Containers`, `WebMobile`, dan`Back-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

   Data konfigurasi adalah kumpulan pengaturan yang memengaruhi perilaku aplikasi Anda. 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 jenis profil konfigurasi berikut:

   - Bendera fitur: 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. URI untuk konfigurasi flag fitur sederhana`hosted`.
   - Konfigurasi bentuk bebas: Konfigurasi bentuk bebas dapat menyimpan data di salah satu alat berikut dan Systems Layanan AWS Manager:
     - AWS AppConfig toko konfigurasi yang dihosting
     - Amazon Simple Storage Service

- AWS CodePipeline
- AWS Secrets Manager
- AWS Systems Manager (SSM) Parameter Menyimpan
- Toko Dokumen SSM

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

4. (Opsional, tetapi disarankan) Buat bendera fitur multi-varian

   AWS AppConfig menawarkan flag fitur dasar, yang (jika diaktifkan) mengembalikan kumpulan data konfigurasi tertentu per permintaan. Untuk mendukung segmentasi pengguna dan kasus penggunaan pemisahan lalu lintas dengan lebih baik, AWS AppConfig juga menawarkan tanda fitur multi-varian, yang memungkinkan Anda menentukan serangkaian nilai tanda yang mungkin untuk dikembalikan untuk permintaan. Anda juga dapat mengonfigurasi status yang berbeda (diaktifkan atau dinonaktifkan) untuk bendera multi-varian. Saat meminta flag yang dikonfigurasi dengan varian, aplikasi Anda menyediakan konteks yang AWS AppConfig mengevaluasi terhadap serangkaian aturan yang ditetapkan pengguna. Bergantung pada konteks yang ditentukan dalam permintaan dan aturan yang ditetapkan untuk varian, AWS AppConfig mengembalikan nilai flag yang berbeda ke aplikasi.

Topik

- Memahami peran IAM profil konfigurasi
- Membuat namespace untuk aplikasi Anda di AWS AppConfig
- Membuat lingkungan untuk aplikasi Anda di AWS AppConfig
- Membuat profil konfigurasi di AWS AppConfig

Memahami 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

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
             ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret_name-a1b2c3"
            ]
        }
    ]
}
```

Sumber konfigurasi adalah parameter Parameter Store

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:parameter/parameter_name"
            ]
        }
    ]
```

```
    }
```

Sumber konfigurasi adalah dokumen SSM

JSON

```
{

    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/document_name"
            ]
        }
    ]
}
```

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

JSON

```
{

  "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.

Prosedur berikut 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 Memahami AWS AppConfig ekstensi.

> **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)
- Membuat AWS AppConfig aplikasi (baris perintah)

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 Memahami 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 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.

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 Konfigurasikan izin untuk rollback otomatis.

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 memanggil `cloudwatch:DescribeAlarms` metrik yang ingin Anda pantau untuk alarm.

7. Dalam daftar CloudWatch alarm, masukkan Nama Sumber Daya Amazon (ARNs) satu atau beberapa metrik untuk dipantau. AWS AppConfig memutar kembali penerapan konfigurasi Anda jika salah satu metrik ini masuk ke status. `ALARM` Untuk informasi tentang metrik yang direkomendasikan, lihat Memantau penyebaran untuk rollback otomatis
8. (Opsional) Di bagian Ekstensi asosiasi, pilih ekstensi dari daftar. Untuk informasi selengkapnya, lihat Memahami 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

Data konfigurasi adalah kumpulan pengaturan yang memengaruhi perilaku aplikasi Anda. 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 jenis profil konfigurasi berikut:

- Bendera fitur: 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. URI untuk konfigurasi flag fitur sederhana`hosted`.

- Konfigurasi bentuk bebas: Konfigurasi bentuk bebas dapat menyimpan data di salah satu alat berikut dan Systems Layanan AWS Manager:
  - AWS AppConfig toko konfigurasi yang dihosting
  - Amazon Simple Storage Service
  - AWS CodePipeline
  - AWS Secrets Manager
  - AWS Systems Manager (SSM) Parameter Menyimpan
  - Toko Dokumen SSM

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

Berikut adalah beberapa contoh data konfigurasi untuk membantu Anda lebih memahami berbagai jenis data konfigurasi dan bagaimana mereka dapat digunakan baik dalam flag fitur atau bebas dari profil konfigurasi.

Data konfigurasi bendera fitur

Data 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
```

Data konfigurasi operasional

Data 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
```

Akses data konfigurasi daftar kontrol

Data 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
```

Topik

- Membuat profil konfigurasi bendera fitur di AWS AppConfig
- Membuat profil konfigurasi formulir gratis di AWS AppConfig
- Membuat profil konfigurasi untuk sumber data non-native

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.

> **Note**
>
> Saat membuat profil konfigurasi flag fitur, Anda dapat membuat flag fitur dasar sebagai bagian dari alur kerja profil konfigurasi. AWS AppConfig juga mendukung bendera fitur multi-varian. Bendera fitur multi-varian memungkinkan Anda untuk menentukan satu set nilai bendera yang mungkin untuk dikembalikan untuk permintaan. Saat meminta flag yang dikonfigurasi dengan varian, aplikasi Anda menyediakan konteks yang AWS AppConfig mengevaluasi terhadap serangkaian aturan yang ditetapkan pengguna. Bergantung pada konteks yang ditentukan dalam permintaan dan aturan yang ditetapkan untuk varian, AWS AppConfig mengembalikan nilai flag yang berbeda ke aplikasi.
> Untuk membuat flag fitur multi-varian, buat profil konfigurasi terlebih dahulu, lalu edit flag apa pun di dalam profil konfigurasi untuk menambahkan varian. Untuk informasi selengkapnya, lihat Membuat bendera fitur multi-varian.

Topik

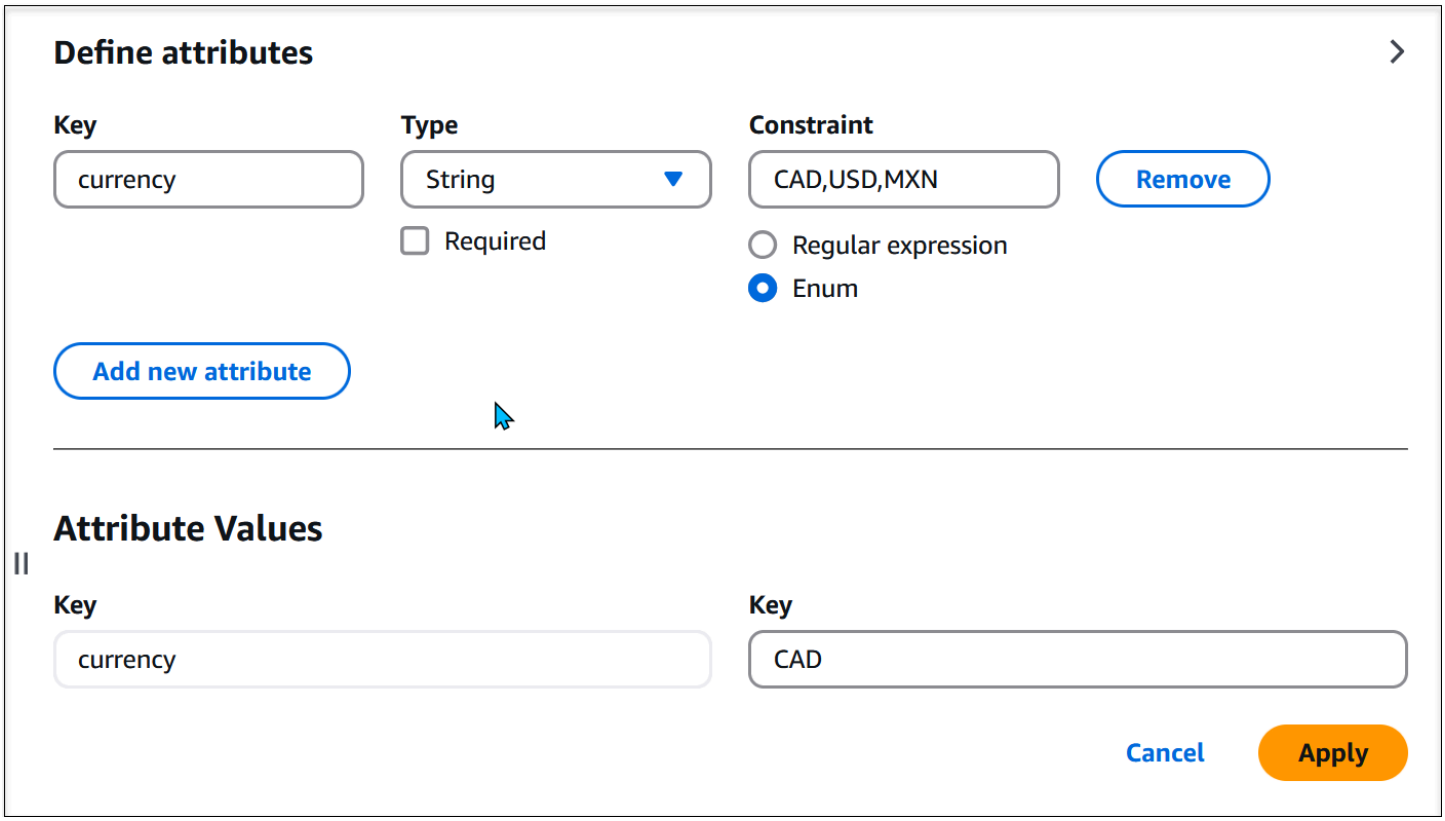
- Memahami atribut bendera fitur
- Membuat profil konfigurasi bendera fitur (konsol)
- Membuat profil konfigurasi bendera fitur (baris perintah)
- Membuat bendera fitur multi-varian

- Memahami referensi tipe untuk AWS.AppConfig.FeatureFlags
- Menyimpan versi bendera fitur sebelumnya ke versi baru

Memahami atribut bendera fitur

Saat membuat profil konfigurasi flag fitur—atau membuat flag baru dalam profil konfigurasi yang ada—Anda dapat menentukan atribut dan batasan yang sesuai untuk flag tersebut. Atribut adalah bidang yang Anda kaitkan dengan flag fitur untuk mengekspresikan properti yang terkait dengan flag fitur Anda. Atribut dikirimkan ke aplikasi Anda dengan kunci bendera dan `enable` atau `disable` nilai bendera.

Batasan memastikan bahwa nilai atribut yang tidak terduga tidak diterapkan ke aplikasi Anda. Gambar berikut menunjukkan sebuah contoh.



> **Note**
>
> Perhatikan informasi berikut tentang atribut bendera.
>
> - 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.

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. Pada saat Anda membuat profil konfigurasi, Anda juga dapat membuat bendera fitur dasar.

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. Pada tab Profil konfigurasi dan bendera fitur, pilih Buat konfigurasi.
4. Di bagian Opsi konfigurasi, pilih Bendera fitur.
5. 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. 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.
   b. Di bagian Tag, pilih Tambahkan tag baru, lalu tentukan kunci dan nilai opsional.
8. Pilih Berikutnya.
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. AWS AppConfig tidak menonaktifkan bendera pada tanggal penghentian.

13. (Opsional) Di bagian atribut bendera fitur, pilih Tentukan atribut. Atribut memungkinkan Anda untuk memberikan nilai tambahan dalam flag Anda. Untuk informasi selengkapnya tentang atribut dan kendala, lihat. Memahami atribut bendera fitur

    a. Untuk Key, tentukan tombol bendera dan pilih jenisnya dari daftar Type. Untuk informasi tentang opsi yang didukung untuk bidang Nilai dan Batasan, lihat bagian yang direferensikan sebelumnya tentang atribut.

    b. Pilih Nilai yang diperlukan untuk menentukan apakah nilai atribut diperlukan.

    c. Pilih Tentukan atribut untuk menambahkan atribut tambahan.

14. 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.
15. Pilih Berikutnya.
16. Pada halaman Tinjau dan simpan, verifikasi detail bendera dan kemudian Simpan dan lanjutkan untuk menerapkan.

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

Membuat 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. Pada saat Anda membuat profil konfigurasi, Anda juga dapat membuat bendera fitur dasar.

Untuk membuat konfigurasi flag fitur

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 APPLICATION_ID \
  --name CONFIGURATION_PROFILE_NAME \
  --location-uri hosted \
```

```
  --type AWS.AppConfig.FeatureFlags
```

Windows

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

PowerShell

```
New-APPCConfigurationProfile `
  -Name CONFIGURATION_PROFILE_NAME `
  -ApplicationId 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, lihatMemahami referensi tipe 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 APPLICATION_ID \
  --configuration-profile-id CONFIGURATION_PROFILE_ID \
  --content-type "application/json" \
  --content file://path/to/feature_flag_configuration_data.json \
  --cli-binary-format raw-in-base64-out
```

   Windows

```
aws appconfig create-hosted-configuration-version ^
  --application-id APPLICATION_ID ^
  --configuration-profile-id CONFIGURATION_PROFILE_ID ^
  --content-type "application/json" ^
```

```
  --content file://path/to/feature_flag_configuration_data.json ^
  --cli-binary-format raw-in-base64-out
```

PowerShell

```
New-APPCHostedConfigurationVersion `
  -ApplicationId APPLICATION_ID `
  -ConfigurationProfileId CONFIGURATION_PROFILE_ID `
  -ContentType "application/json" `
  -Content file://path/to/feature_flag_configuration_data.json
```

Perintah memuat konten yang ditentukan untuk `Content` parameter dari disk. Konten harus mirip dengan contoh berikut.

```
{
    "flags": {
        "ui_refresh": {
            "name": "UI Refresh"
        }
    },
    "values": {
        "ui_refresh": {
            "enabled": false,
            "attributeValues": {
                "dark_mode_support": true
            }
        }
    },
    "version": "1"
}
```

Sistem mengembalikan informasi seperti berikut ini.

Linux

```
{
   "ApplicationId"          : "ui_refresh",
   "ConfigurationProfileId" : "UI Refresh",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
```

```
}
```

Windows

```
{
   "ApplicationId"          : "ui_refresh",
   "ConfigurationProfileId" : "UI Refresh",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
}
```

PowerShell

```
ApplicationId          : ui_refresh
ConfigurationProfileId : UI Refresh
VersionNumber          : 1
ContentType            : application/json
```

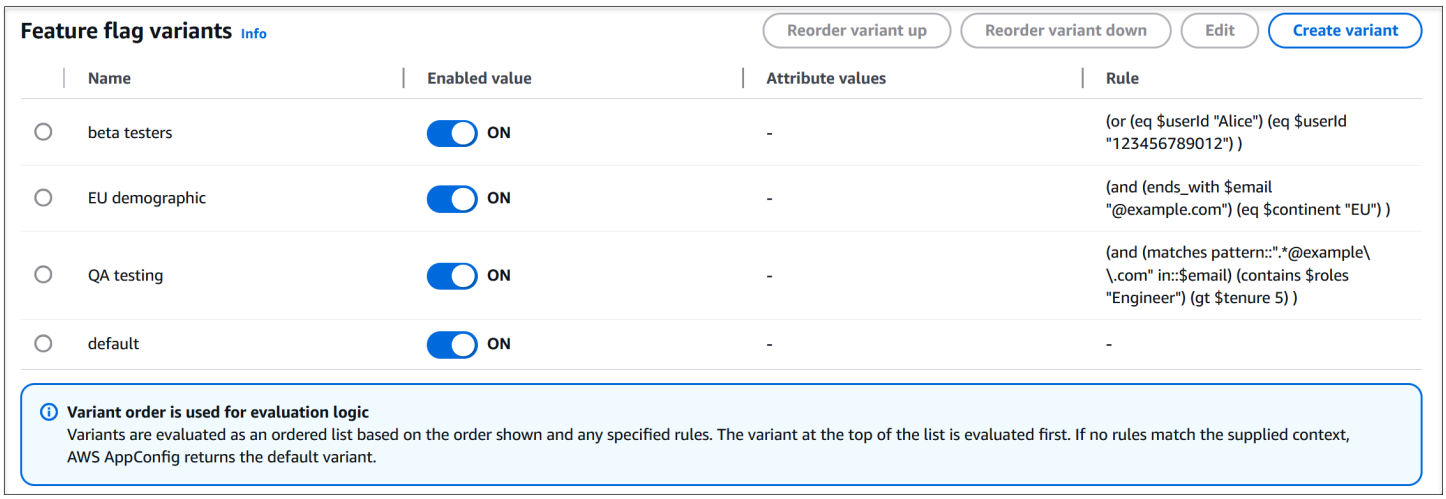
`service_returned_content_file`Berisi data konfigurasi Anda yang mencakup beberapa AWS AppConfig metadata yang dihasilkan.

> **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.

Membuat bendera fitur multi-varian

Varian bendera fitur memungkinkan Anda untuk menentukan satu set nilai bendera yang mungkin untuk kembali untuk permintaan. Anda juga dapat mengonfigurasi status yang berbeda (diaktifkan atau dinonaktifkan) untuk bendera multi-varian. Saat meminta flag yang dikonfigurasi dengan varian, aplikasi Anda menyediakan konteks yang AWS AppConfig mengevaluasi terhadap serangkaian aturan yang ditetapkan pengguna. Bergantung pada konteks yang ditentukan dalam permintaan dan aturan yang ditetapkan untuk varian, AWS AppConfig mengembalikan nilai flag yang berbeda ke aplikasi.

Tangkapan layar berikut menunjukkan contoh bendera fitur dengan tiga varian yang ditentukan pengguna dan varian default.



Topik

- Memahami konsep bendera fitur multi-varian dan kasus penggunaan umum
- Memahami aturan bendera fitur multi-varian
- Membuat bendera fitur multi-varian

Memahami konsep bendera fitur multi-varian dan kasus penggunaan umum

Untuk membantu Anda lebih memahami varian bendera fitur, bagian ini menjelaskan konsep varian bendera dan kasus penggunaan umum.

Konsep

- Bendera fitur: Jenis AWS AppConfig konfigurasi yang digunakan untuk mengontrol perilaku fitur dalam aplikasi. Bendera memiliki status (diaktifkan atau dinonaktifkan) dan kumpulan atribut opsional yang berisi nilai string, numerik, boolean, atau array arbitrer.
- Varian bendera fitur: Kombinasi spesifik dari nilai status dan atribut milik bendera fitur. Bendera fitur mungkin memiliki beberapa varian.
- Aturan varian: Ekspresi yang ditentukan pengguna yang digunakan untuk memilih varian bendera fitur. Setiap varian memiliki aturannya sendiri yang AWS AppConfig mengevaluasi untuk menentukan apakah akan mengembalikannya atau tidak.
- Varian default: Varian khusus yang dikembalikan ketika tidak ada varian lain yang dipilih. Semua flag fitur multi-varian memiliki varian default.

Catatan, varian default harus terakhir dalam urutan varian Anda, dan tidak dapat memiliki aturan yang terkait dengannya. Jika tidak didefinisikan terakhir, AWS AppConfig mengembalikan `BadRequestException` ketika Anda mencoba untuk membuat bendera multi-varian.

- Konteks: Kunci dan nilai yang ditentukan pengguna diteruskan AWS AppConfig pada waktu pengambilan konfigurasi. Nilai konteks digunakan selama evaluasi aturan untuk memilih varian bendera fitur yang akan dikembalikan.

> **Note**
>
> AWS AppConfig agen mengevaluasi aturan varian dan menentukan aturan mana yang berlaku untuk permintaan berdasarkan konteks yang disediakan. Untuk informasi selengkapnya tentang mengambil flag fitur multi-varient, lihat. Mengambil flag fitur dasar dan multi-varian

Kasus penggunaan umum

Bagian ini menjelaskan dua kasus penggunaan umum untuk varian bendera fitur.

Segmentasi pengguna

Segmentasi pengguna adalah proses membagi pengguna berdasarkan atribut tertentu. Sebagai contoh, Anda dapat menggunakan varian bendera untuk mengekspos fitur ke beberapa pengguna tetapi tidak yang lain berdasarkan ID pengguna, lokasi geografis, jenis perangkat, atau frekuensi pembelian mereka.

Dengan menggunakan contoh frekuensi pembelian, misalkan aplikasi perdagangan Anda mendukung fitur untuk meningkatkan loyalitas pelanggan. Anda dapat menggunakan varian bendera untuk mengonfigurasi jenis insentif yang berbeda untuk ditampilkan kepada pengguna berdasarkan kapan mereka terakhir membeli sesuatu. Pengguna baru mungkin ditawari diskon kecil untuk mendorong mereka menjadi pelanggan, sedangkan pelanggan tetap mungkin diberi diskon yang lebih besar jika mereka membeli sesuatu dari kategori baru.

Pemisahan lalu lintas

Pemisahan lalu lintas adalah proses memilih varian bendera acak, tetapi konsisten, berdasarkan nilai konteks yang Anda tentukan. Misalnya, Anda mungkin ingin melakukan eksperimen di mana sebagian kecil pengguna Anda (diidentifikasi oleh ID pengguna mereka) melihat varian tertentu. Atau,

Anda mungkin ingin menjalankan peluncuran fitur bertahap di mana fitur pertama kali diekspos ke 5% pengguna Anda, kemudian 15%, kemudian 40%, lalu 100%, sambil mempertahankan pengalaman pengguna yang konsisten selama peluncuran.

Menggunakan contoh eksperimen, Anda dapat menggunakan varian bendera untuk menguji gaya tombol baru untuk tindakan utama di beranda aplikasi Anda untuk melihat apakah itu mendorong lebih banyak klik. Untuk eksperimen, Anda dapat membuat varian bendera dengan aturan pemisahan lalu lintas yang memilih 5% pengguna untuk melihat gaya baru, sedangkan varian default menunjukkan pengguna yang harus terus melihat gaya yang ada. Jika percobaan berhasil, Anda dapat meningkatkan nilai persentase, atau bahkan mengubah varian itu menjadi default.

Memahami aturan bendera fitur multi-varian

Saat Anda membuat varian bendera fitur, Anda menentukan aturan untuk itu. Aturan adalah ekspresi yang mengambil nilai konteks sebagai input dan menghasilkan hasil boolean sebagai output. Misalnya, Anda dapat menentukan aturan untuk memilih varian bendera untuk pengguna beta, yang diidentifikasi oleh ID akun mereka, menguji penyegaran antarmuka pengguna. Untuk skenario ini, Anda melakukan hal berikut:

1. Buat profil konfigurasi flag fitur baru yang disebut UI Refresh.
2. Buat flag fitur baru yang disebut ui_refresh.
3. Edit bendera fitur setelah Anda membuatnya untuk menambahkan varian.
4. Buat dan aktifkan varian baru yang disebut BetaUsers.
5. Tentukan aturan untuk BetaUsersmemilih varian jika ID akun dari konteks permintaan ada dalam daftar akun yang IDs disetujui untuk melihat pengalaman beta baru.
6. Konfirmasikan bahwa status varian default disetel ke Dinonaktifkan.

> **Note**
>
> Varian dievaluasi sebagai daftar berurutan berdasarkan urutan yang ditentukan di konsol. Varian di bagian atas daftar dievaluasi terlebih dahulu. Jika tidak ada aturan yang cocok dengan konteks yang disediakan, AWS AppConfig mengembalikan varian Default.

Saat AWS AppConfig memproses permintaan flag fitur, ia membandingkan konteks yang disediakan, yang menyertakan accountID (untuk contoh ini) dengan varian terlebih dahulu. BetaUsers Jika konteksnya cocok dengan aturan untuk BetaUsers, AWS AppConfig mengembalikan data konfigurasi

untuk pengalaman beta. Jika konteks tidak menyertakan ID akun atau jika ID akun berakhir dengan apa pun selain 123, AWS AppConfig mengembalikan data konfigurasi untuk aturan Default, yang berarti pengguna melihat pengalaman saat ini dalam produksi.

> **Note**
>
> Untuk informasi tentang mengambil flag fitur multi-varian, lihat. Mengambil flag fitur dasar dan multi-varian

Mendefinisikan aturan untuk flag fitur multi-varian

Aturan varian adalah ekspresi yang terdiri dari satu atau lebih operan dan operator. Operan adalah nilai spesifik yang digunakan selama evaluasi aturan. Nilai operan dapat berupa statis, seperti angka literal atau string, atau variabel, seperti nilai yang ditemukan dalam konteks atau hasil ekspresi lain. Operator, seperti “lebih besar dari”, adalah tes atau tindakan yang diterapkan pada operannya yang menghasilkan nilai. Ekspresi aturan varian harus menghasilkan “true” atau “false” agar valid.

Operan

Tipe	Deskripsi	Contoh
String	Urutan karakter UTF-8, terlampir dalam tanda kutip ganda.	`"apple", "###ë# ##š##"`
Bilangan Bulat	Nilai integer 64-bit.	`-7, 42`
Desimal	Nilai floating-point 64-bit IEEE-754.	`3.14, 1.234e-5`
Stempel waktu	Momen tertentu dalam waktu seperti yang dijelaskan oleh catatan W3C pada format tanggal dan waktu.	`2012-03-04T05:06:07-08:00, 2024-01`
Boolean	Nilai benar atau salah.	`true, false`

Tipe	Deskripsi	Contoh
Nilai konteks	Nilai parameter dalam bentuk $ *key* yang diambil dari konteks selama evaluasi aturan.	`$country, $userId`

Operator perbandingan

Operator	Deskripsi	Contoh
persamaan	Menentukan apakah nilai konteks sama dengan nilai yang diberikan.	`(eq $state "Virginia")`
gt	Menentukan apakah nilai konteks lebih besar dari nilai yang diberikan.	`(gt $age 65)`
gte	Menentukan apakah nilai konteks lebih besar dari atau sama dengan nilai yang diberikan.	`(gte $age 65)`
lt	Menentukan apakah nilai konteks kurang dari nilai yang diberikan.	`(lt $age 65)`
lte	Menentukan apakah nilai konteks kurang dari atau sama dengan nilai yang diberikan.	`(lte $age 65)`

Operator logis

Operator	Deskripsi	Contoh
and	Menentukan apakah kedua operan benar.	`(and (eq $state "Virginia") (gt $age 65))`
atau	Menentukan apakah setidakny a salah satu operan benar.	`(or (eq $state "Virginia") (gt $age 65))`
tidak	Membalikkan nilai ekspresi.	`(not (eq $state "Virginia"))`

Operator kustom

Operator	Deskripsi	Contoh
mulai_dengan	Menentukan apakah nilai konteks dimulai dengan awalan yang diberikan.	`(begins_with $state "A")`
berakhir_dengan	Menentukan apakah nilai konteks berakhir dengan awalan yang diberikan.	`(ends_with $email "amazon.com")`
mengandung	Menentukan apakah nilai konteks berisi substring yang diberikan.	`(contains $promoCode "WIN")`
in	Menentukan apakah nilai konteks terkandung dalam daftar konstanta.	`(in $userId ["123", "456"])`

Operator	Deskripsi	Contoh
korek api	Menentukan apakah nilai konteks cocok dengan pola regex yang diberikan.	`(matches in::$greeting pattern::"h.*y")`
exists	Menentukan apakah ada nilai yang disediakan untuk kunci konteks.	`(exists key::"country")`
terbelah	Mengevaluasi `true` untuk persentase lalu lintas tertentu berdasarkan hash yang konsisten dari nilai konteks yang disediakan. Untuk penjelasan rinci tentang cara `split` kerja, lihat bagian selanjutnya dalam topik ini,Memahami operator split. Perhatikan bahwa `seed` ini adalah properti opsional. Jika Anda tidak menentukan`seed`, hash konsisten secara lokal, yang berarti lalu lintas akan dibagi secara konsisten untuk flag itu, tetapi flag lain yang menerima nilai konteks yang sama dapat membagi lalu lintas secara berbeda. Jika `seed` disediakan, setiap nilai unik dijamin untuk membagi lalu lintas secara konsisten di seluruh bendera fitur, profil konfigurasi, dan Akun AWS.	`(split pct::10 by::$userId seed::"abc")`

Memahami operator split

Bagian berikut menjelaskan bagaimana `split` operator berperilaku ketika digunakan dalam skenario yang berbeda. Sebagai pengingat, `split` evaluasi `true` untuk persentase lalu lintas tertentu berdasarkan hash yang konsisten dari nilai konteks yang disediakan. Untuk memahami hal ini dengan lebih baik, pertimbangkan skenario dasar berikut yang menggunakan split dengan dua varian:

```
A: (split by::$uniqueId pct::20)
C: <no rule>
```

Seperti yang diharapkan, memberikan serangkaian `uniqueId` nilai acak menghasilkan distribusi yang kira-kira:

```
A: 20%
C: 80%
```

Jika Anda menambahkan varian ketiga, tetapi gunakan persentase split yang sama seperti:

```


A: (split by::$uniqueId pct::20)
B: (split by::$uniqueId pct::20)
C: <default>
```

Anda berakhir dengan distribusi berikut:

```
A: 20%
B: 0%
C: 80%
```

Distribusi yang berpotensi tidak terduga ini terjadi karena setiap aturan varian dievaluasi secara berurutan dan kecocokan pertama menentukan varian yang dikembalikan. Ketika aturan A dievaluasi, 20% `uniqueId` nilai cocok dengannya, jadi varian pertama dikembalikan. Selanjutnya, aturan B dievaluasi. Namun, semua `uniqueId` nilai yang akan cocok dengan pernyataan split kedua sudah dicocokkan dengan aturan varian A, jadi tidak ada nilai yang cocok B. Varian default dikembalikan sebagai gantinya.

Sekarang perhatikan contoh ketiga.

```
A: (split by::$uniqueId pct::20)
B: (split by::$uniqueId pct::25)
C: <default>
```

Seperti contoh sebelumnya, 20% `uniqueId` nilai pertama cocok dengan aturan A. Untuk aturan varian B, 25% dari semua `uniqueId` nilai akan cocok, tetapi sebagian besar dari aturan yang sebelumnya cocok A. Itu menyisakan 5% dari total untuk varian B, dengan sisanya menerima varian C. Distribusi akan terlihat seperti berikut:

```
A: 20%
B: 5%
C: 75%
```

Menggunakan **`seed`** properti

Anda dapat menggunakan `seed` properti untuk memastikan lalu lintas dibagi secara konsisten untuk nilai konteks tertentu terlepas dari di mana operator split digunakan. Jika Anda tidak menentukan`seed`, hash konsisten secara lokal, yang berarti lalu lintas akan dibagi secara konsisten untuk flag itu, tetapi flag lain yang menerima nilai konteks yang sama dapat membagi lalu lintas secara berbeda. Jika `seed` disediakan, setiap nilai unik dijamin untuk membagi lalu lintas secara konsisten di seluruh bendera fitur, profil konfigurasi, dan Akun AWS.

Biasanya, pelanggan menggunakan `seed` nilai yang sama di seluruh varian dalam bendera saat memisahkan lalu lintas pada properti konteks yang sama. Namun, terkadang masuk akal untuk menggunakan nilai benih yang berbeda. Berikut adalah contoh yang menggunakan benih berbeda untuk aturan A dan B:

```
A: (split by::$uniqueId pct::20 seed::"seed_one")
B: (split by::$uniqueId pct::25 seed::"seed_two")
C: <default>
```

Seperti sebelumnya, 20% dari `uniqueId` nilai yang cocok dengan aturan A. Itu berarti 80% nilai jatuh dan diuji terhadap aturan varian B. Karena benih berbeda, tidak ada korelasi antara nilai yang cocok dengan A dan nilai yang cocok B. Namun, hanya 80% `uniqueId` nilai yang dibagi dengan 25% dari aturan pencocokan angka B dan 75% tidak. Itu berhasil untuk distribusi berikut:

```
A: 20%
B: 20% (25% of what falls through from A, or 25% of 80%)
```

```
C: 60%
```

Membuat bendera fitur multi-varian

Gunakan prosedur di bagian ini untuk membuat varian bendera fitur.

Sebelum Anda mulai

Perhatikan informasi penting berikut.

- Anda dapat membuat varian bendera fitur yang ada dengan mengeditnya. Anda tidak dapat membuat varian tanda fitur baru saat membuat profil konfigurasi baru. Anda harus menyelesaikan alur kerja membuat profil konfigurasi baru terlebih dahulu. Setelah Anda membuat profil konfigurasi, Anda dapat menambahkan varian ke bendera apa pun dalam profil konfigurasi. Untuk informasi tentang cara membuat profil konfigurasi baru, lihatMembuat profil konfigurasi bendera fitur di AWS AppConfig.
- Untuk mengambil data varian tanda fitur untuk platform komputasi Amazon EC2, Amazon ECS, dan Amazon EKS, Anda harus AWS AppConfig menggunakan Agen versi 2.0.4416 atau yang lebih baru.
- Untuk alasan kinerja, AWS CLI dan panggilan SDK untuk AWS AppConfig tidak mengambil data varian. Untuk informasi lebih lanjut tentang AWS AppConfig Agen, lihatCara menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi.
- Saat Anda membuat varian bendera fitur, Anda menentukan aturan untuk itu. Aturan adalah ekspresi yang mengambil konteks permintaan sebagai input dan menghasilkan hasil boolean sebagai output. Sebelum Anda membuat varian, tinjau operan dan operator yang didukung untuk menandai aturan varian. Anda dapat membuat aturan sebelum membuat varian. Untuk informasi selengkapnya, lihat Memahami aturan bendera fitur multi-varian.

Topik

- Membuat bendera fitur multi-varian (konsol)
- Membuat bendera fitur multi-varian (baris perintah)

Membuat bendera fitur multi-varian (konsol)

Prosedur berikut menjelaskan cara membuat flag fitur multi-varian untuk profil konfigurasi yang ada dengan menggunakan AWS AppConfig konsol. Anda juga dapat mengedit bendera fitur yang ada untuk membuat varian.

Untuk membuat bendera fitur multi-varian

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi.
3. Pada tab Profil konfigurasi dan bendera fitur, pilih profil konfigurasi bendera fitur yang ada.
4. Di bagian Bendera, pilih Tambahkan bendera baru.
5. Di bagian Definisi bendera fitur, untuk nama Bendera, masukkan nama.
6. 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.
7. (Opsional) Di bidang Deskripsi, masukkan informasi tentang bendera ini.
8. Di bagian Varian, pilih Bendera Multi-varian.
9. (Opsional) Di bagian atribut bendera fitur, pilih Tentukan atribut. Atribut memungkinkan Anda untuk memberikan nilai tambahan dalam flag Anda. Untuk informasi selengkapnya tentang atribut dan kendala, lihat. Memahami atribut bendera fitur

    a. Untuk Key, tentukan tombol bendera dan pilih jenisnya dari daftar Type. Untuk informasi tentang opsi yang didukung untuk bidang Nilai dan Batasan, lihat bagian yang direferensikan sebelumnya tentang atribut.

    b. Pilih Nilai yang diperlukan untuk menentukan apakah nilai atribut diperlukan.

    c. Pilih Tentukan atribut untuk menambahkan atribut tambahan.

    d. Pilih Terapkan untuk menyimpan perubahan atribut.
10. Di bagian varian bendera fitur, pilih Buat varian.

    a. Untuk nama Variant, masukkan nama.

    b. Gunakan sakelar nilai Diaktifkan untuk mengaktifkan varian.

    c. Di kotak teks Aturan, masukkan aturan.

    d. Gunakan opsi Buat varian> Buat varian di atas atau Buat varian di bawah ini untuk membuat varian tambahan untuk bendera ini.

    e. Di bagian varian Default, gunakan sakelar Nilai diaktifkan untuk mengaktifkan varian default. Secara opsional, berikan nilai untuk atribut yang ditentukan pada langkah 10.

    f. Pilih Terapkan.

11. Verifikasi detail bendera dan variannya dan pilih Buat bendera.

Untuk informasi tentang penerapan flag fitur baru Anda dengan varian, lihatMenerapkan flag fitur dan data konfigurasi di AWS AppConfig.

Membuat bendera fitur multi-varian (baris perintah)

Prosedur berikut menjelaskan cara menggunakan AWS Command Line Interface (di Linux atau Windows) atau Alat untuk Windows PowerShell untuk membuat bendera fitur multi-varian untuk profil konfigurasi yang ada. Anda juga dapat mengedit bendera fitur yang ada untuk membuat varian.

Sebelum Anda mulai

Selesaikan tugas-tugas berikut sebelum Anda membuat flag fitur multi-varian dengan menggunakan. AWS CLI

- Buat profil konfigurasi bendera fitur. Untuk informasi selengkapnya, lihat Membuat profil konfigurasi bendera fitur di AWS AppConfig.
- Perbarui ke versi terbaru dari AWS CLI. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru dari Panduan AWS Command Line Interface Pengguna. AWS CLI

Untuk membuat bendera fitur multi-varian

1. Buat file konfigurasi di mesin lokal Anda yang menentukan detail bendera multi-varian yang ingin Anda buat. Simpan file dengan ekstensi `.json` file. File harus mematuhi skema `AWS.AppConfig.FeatureFlags`JSON. Isi skema file konfigurasi Anda akan mirip dengan yang berikut ini.

```
{
  "flags": {
    "FLAG_NAME": {
      "attributes": {
          "ATTRIBUTE_NAME": {
          "constraints": {
            "type": "CONSTRAINT_TYPE"
          }
        }
      },
      "description": "FLAG_DESCRIPTION",
      "name": "VARIANT_NAME"
```

```
    }
  },
  "values": {
    "VARIANT_VALUE_NAME": {
      "_variants": [
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        }
      ]
    }
  },
  "version": "VERSION_NUMBER"
}
```

Berikut adalah contoh dengan tiga varian dan varian default.

```
{
  "flags": {
    "ui_refresh": {
      "attributes": {
        "dark_mode_support": {
          "constraints": {
            "type": "boolean"
          }
        }
      },
      "description": "A release flag used to release a new UI",
      "name": "UI Refresh"
    }
  },
  "values": {
    "ui_refresh": {
      "_variants": [
        {
          "attributeValues": {
            "dark_mode_support": true
          },
          "enabled": true,
          "name": "QA",
          "rule": "(ends_with $email \"qa-testers.mycompany.com\")"
        },
        {
          "attributeValues": {
            "dark_mode_support": true
          },
          "enabled": true,
          "name": "Beta Testers",
          "rule": "(exists key::\"opted_in_to_beta\")"
        },
        {
          "attributeValues": {
            "dark_mode_support": false
          },
          "enabled": true,
          "name": "Sample Population",
          "rule": "(split pct::10 by::$email)"
        },
```

```
        {
          "attributeValues": {
            "dark_mode_support": false
          },
          "enabled": false,
          "name": "Default Variant"
        }
      ]
    }
  },
  "version": "1"
}
```

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

   Linux

   ```
   aws appconfig create-hosted-configuration-version \
     --application-id APPLICATION_ID \
     --configuration-profile-id CONFIGURATION_PROFILE_ID \
     --content-type "application/json" \
     --content file://path/to/feature_flag_configuration_data.json \
     --cli-binary-format raw-in-base64-out \
     outfile
   ```

   Windows

   ```
   aws appconfig create-hosted-configuration-version ^
     --application-id APPLICATION_ID ^
     --configuration-profile-id CONFIGURATION_PROFILE_ID ^
     --content-type "application/json" ^
     --content file://path/to/feature_flag_configuration_data.json ^
     --cli-binary-format raw-in-base64-out ^
     outfile
   ```

   PowerShell

   ```
   New-APPCHostedConfigurationVersion `
     -ApplicationId APPLICATION_ID `
     -ConfigurationProfileId CONFIGURATION_PROFILE_ID `
     -ContentType "application/json" `
   ```

```
  -Content file://path/to/feature_flag_configuration_data.json `
  -Raw
```

`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.

Memahami referensi tipe 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"],
      "additionalProperties": false
    },
    "flagDefinitions": {
      "type": "object",
```

```
      "patternProperties": {
        "^[a-z][a-zA-Z\\d_-]{0,63}$": {
          "$ref": "#/definitions/flagDefinition"
        }
      },
      "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"]
            },
            "date": {
              "type": "string",
              "format": "date"
            }
          },
         "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"
        }
      },
      "additionalProperties": false
    },
    "flagValue": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
        },
        "_createdAt": {
          "type": "string"
        },
        "_updatedAt": {
```

```
          "type": "string"
        },
        "_variants": {
          "type": "array",
          "maxLength": 32,
          "items": {
            "$ref": "#/definitions/variant"
          }
        }
      },
      "patternProperties": {
        "^[a-z][a-zA-Z\\d_-]{0,63}$": {
          "$ref": "#/definitions/attributeValue",
          "maxProperties": 25
        }
      },
      "additionalProperties": false
    },
    "attributeValue": {
      "oneOf": [
        { "type": "string", "maxLength": 1024 },
        { "type": "number" },
        { "type": "boolean" },
        {
          "type": "array",
          "oneOf": [
            {
              "items": {
                "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" },
```

```
        { "$ref": "#/definitions/stringConstraints" }
      ]
    },
    "variant": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
        },
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "rule": {
          "type": "string",
          "maxLength": 16384
        },
        "attributeValues": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d_-]{0,63}$": {
              "$ref": "#/definitions/attributeValue"
            }
          },
          "maxProperties": 25,
          "additionalProperties": false
        }
      },
      "required": ["name", "enabled"],
      "additionalProperties": false
    },
    "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 GetLatestConfiguration di dalam Referensi API AWS AppConfig .

Saat aplikasi Anda memanggil GetLatestConfigurationdan 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.

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

Menyimpan versi bendera fitur sebelumnya ke versi baru

Saat Anda memperbarui bendera fitur, AWS AppConfig secara otomatis menyimpan perubahan Anda ke versi baru. Jika Anda ingin menggunakan versi bendera fitur sebelumnya, Anda harus

menyalinnya ke versi draf dan kemudian menyimpannya. Anda tidak dapat mengedit dan menyimpan perubahan ke versi bendera sebelumnya tanpa menyimpannya ke versi baru.

Untuk mengedit versi bendera fitur sebelumnya dan menyimpannya ke versi baru

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi dengan bendera fitur yang ingin Anda edit dan simpan ke versi baru.
3. Pada tab Profil konfigurasi dan bendera fitur, pilih profil konfigurasi dengan tanda fitur yang ingin Anda edit dan simpan ke versi baru.
4. Pada tab Feature flags, gunakan daftar Versi untuk memilih versi yang ingin Anda edit dan simpan ke versi baru.
5. Pilih Salin ke versi draf.
6. Di bidang label Versi, masukkan label baru (opsional, tetapi disarankan).
7. Di bidang Deskripsi versi, masukkan deskripsi baru (opsional, tetapi disarankan).
8. Pilih Simpan versi.
9. Pilih Mulai penerapan untuk menerapkan versi baru.

Membuat profil konfigurasi formulir gratis di AWS AppConfig

Data konfigurasi adalah kumpulan pengaturan yang memengaruhi perilaku aplikasi Anda. Profil konfigurasi mencakup, antara lain, URI yang memungkinkan AWS AppConfig untuk menemukan data konfigurasi Anda di lokasi yang disimpan dan jenis konfigurasi. Dengan profil konfigurasi bentuk bebas, Anda dapat menyimpan data Anda di toko konfigurasi yang AWS AppConfig dihosting atau salah satu alat berikut Layanan AWS dan Systems Manager:

Lokasi	Tipe file yang didukung
AWS AppConfig toko konfigurasi yang dihosting	YAMM, JSON, dan teks jika ditambahkan menggunakan file. Konsol Manajemen AWS Tipe file apa pun jika ditambahkan menggunakan tindakan AWS AppConfig CreateHostedConfigurationVersionAPI.

Lokasi	Tipe file yang didukung
Amazon Simple Storage Service (Amazon S3)	Setiap
AWS CodePipeline	Pipeline (seperti yang didefinisikan oleh layanan)
AWS Secrets Manager	Rahasia (seperti yang 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, sebaiknya hosting 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

- Memahami validator
- Memahami kuota dan batasan penyimpanan konfigurasi
- Memahami penyimpanan konfigurasi yang AWS AppConfig dihosting
- Memahami konfigurasi yang disimpan di Amazon S3
- Membuat profil konfigurasi AWS AppConfig bentuk bebas (konsol)
- Membuat profil konfigurasi AWS AppConfig bentuk bebas (baris perintah)

Memahami 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.

Memahami 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, maksimum 4 MB	2 MB Ditegakkan oleh AWS AppConfig, bukan S3	4 KB (tingkat gratis)/8 KB (parameter lanjutan)	64 KB	64 KB	2 MB Dipaksakan oleh AWS AppConfig, tidak CodePipeline
Batas penyimpanan sumber daya	1 GB	Tidak terbatas.	10.000 parameter (tingkat gratis) /100.0 parameter	500.000	500 dokumen	Dibatasi oleh jumlah profil konfigurasi per aplikasi

	AWS AppConfig toko konfigurasi yang dihosting	Amazon S3	Penyimpanan Parameter Systems Manager	AWS Secrets Manager	Toko dokumen Systems Manager	AWS CodePipeline
			(parameter lanjutan)			(100 profil per aplikasi)
Enkripsi sisi server	Ya	SSE-S3, SSE-KM	Ya	Ya	Tidak	Ya
CloudFormation dukungan	Ya	Bukan untuk membuat atau memperbarui data	Ya	Ya	Tidak	Ya
Harga	Kosong	Lihat harga Amazon S3	Lihat AWS Systems Manager harga	Lihat AWS Secrets Manager harga	Kosong	Lihat AWS CodePipeline harga

Memahami penyimpanan 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.

Memahami 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 sebuah 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`
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: GetObjectVersion

Izin untuk mencantumkan bucket S3

s3: ListAllMyBuckets

Izin untuk mengakses bucket S3 tempat objek disimpan

- s3: GetBucketLocation
- s3: GetBucketVersioning
- s3: ListBucket
- s3: ListBucketVersions

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 *placeholder values* dengan informasi Anda sendiri.

   JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/my-configurations/my-
configuration.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-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.

   JSON

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

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Pilih Perbarui Kebijakan Kepercayaan.

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 Berikutnya.
9. Pada halaman Tentukan data konfigurasi, di bagian Definisi 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 YAMM, dan masukkan konfigurasi Anda di bidang. Pergi ke Langkah 12 dalam prosedur ini.

Opsi dipilih	Detail
Objek Amazon S3	Masukkan URI objek di bidang sumber objek S3 dan pergi ke Langkah 11 dalam prosedur ini.
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 Berikutnya.
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 tampilMenerapkan flag fitur dan data konfigurasi di AWS AppConfig. Namun, Anda harus mengonfigurasi klien untuk menerima pembaruan konfigurasi aplikasi seperti yang dijelaskan dalamMengambil data konfigurasi tanpa Agen AWS AppConfig. 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 Menerapkan 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 Panduan Pengguna AWS CloudShell .

> **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 APPLICATION_ID \
     --name NAME \
     --description CONFIGURATION_PROFILE_DESCRIPTION \
     --location-uri CONFIGURATION_URI or hosted \
     --retrieval-role-arn IAM_ROLE_ARN \
     --tags TAGS \
     --validators "Content=SCHEMA_CONTENT or LAMBDA_FUNCTION_ARN,Type=JSON_SCHEMA
    or LAMBDA"
   ```

   Windows

   ```
   aws appconfig create-configuration-profile ^
     --application-id APPLICATION_ID ^
     --name NAME ^
     --description CONFIGURATION_PROFILE_DESCRIPTION ^
     --location-uri CONFIGURATION_URI or hosted  ^
     --retrieval-role-arn IAM_ROLE_ARN ^
     --tags TAGS ^
     --validators "Content=SCHEMA_CONTENT or LAMBDA_FUNCTION_ARN,Type=JSON_SCHEMA
    or LAMBDA"
   ```

   PowerShell

   ```
   New-APPCConfigurationProfile `
     -Name NAME `
     -ApplicationId APPLICATION_ID `
     -Description CONFIGURATION_PROFILE_DESCRIPTION `
   ```

```
  -LocationUri CONFIGURATION_URI or hosted `
  -RetrievalRoleArn IAM_ROLE_ARN `
  -Tag TAGS `
  -Validators "Content=SCHEMA_CONTENT or LAMBDA_FUNCTION_ARN,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 tampilMenerapkan flag fitur dan data konfigurasi di AWS AppConfig. Namun, Anda harus mengonfigurasi klien untuk menerima pembaruan konfigurasi aplikasi seperti yang dijelaskan dalamMengambil data konfigurasi tanpa Agen AWS AppConfig. 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-versiondi AWS CLI Command Reference.

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

Membuat profil konfigurasi untuk sumber data non-native

AWS AppConfig mendukung penyebaran data konfigurasi dari sebagian besar penyimpanan data apa pun. Secara native, AWS AppConfig mendukung penerapan data konfigurasi yang disimpan dalam layanan berikut:

- Toko konfigurasi yang AWS AppConfig dihosting
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager Toko Parameter

- Toko Dokumen Systems Manager
- AWS CodePipeline

Jika data konfigurasi disimpan di lokasi yang tidak didukung secara native AWS AppConfig, Anda dapat membuat AWS AppConfig ekstensi untuk mengambil data dari sumbernya. Misalnya, dengan menggunakan AWS AppConfig ekstensi, Anda dapat mengambil data konfigurasi yang disimpan di Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB (DynamoDB GitHub) GitLab,,, atau repo lokal, untuk beberapa nama. Dengan menerapkan ekstensi, Anda dapat memanfaatkan AWS AppConfig keamanan dan DevOps peningkatan untuk aplikasi dan lingkungan komputasi Anda. Anda juga dapat menggunakan metode ini saat memigrasikan data konfigurasi dari sistem lama ke dalam. AWS AppConfig

Membuat profil konfigurasi untuk sumber data yang tidak didukung secara native AWS AppConfig melibatkan proses atau tindakan berikut:

1. Buat AWS Lambda fungsi yang mengambil data dari sumber data Anda. Selama fungsi Lambda dapat mengakses sumber data, AWS AppConfig ekstensi Anda akan dapat mengambil data.
2. Buat AWS AppConfig ekstensi khusus yang memanggil fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Walkthrough: Membuat ekstensi khusus AWS AppConfig.
3. Buat profil konfigurasi AWS AppConfig bentuk bebas. Secara khusus, buat profil konfigurasi yang menggunakan definisi konfigurasi yang AWS AppConfig dihosting. Profil konfigurasi berfungsi sebagai penyimpanan data sementara setelah fungsi Lambda Anda mengambil konfigurasi Anda dari sumber Anda. Aplikasi Anda akan mengambil data konfigurasi dari toko konfigurasi yang AWS AppConfig dihosting. Untuk informasi selengkapnya, lihat Membuat profil konfigurasi formulir gratis di AWS AppConfig.
4. Buat asosiasi ekstensi yang memicu menggunakan titik `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` tindakan. Untuk informasi selengkapnya, lihat Langkah 4: Buat asosiasi ekstensi untuk AWS AppConfig ekstensi khusus.

Setelah dikonfigurasi, ketika aplikasi Anda meminta versi baru dari data konfigurasi, Lambda mengambil data konfigurasi Anda dan menariknya ke profil konfigurasi. AWS AppConfig kemudian menyimpan profil konfigurasi dan data pihak ketiga Anda.

Ketika Anda siap, Anda dapat menyebarkan profil konfigurasi ke aplikasi Anda, sama seperti jenis data konfigurasi lainnya.

> **Note**
>
> Anda dapat memilih untuk menyisipkan data pihak ketiga sesuai dengan data konfigurasi yang ada, atau seluruh konten data konfigurasi hanya berisi data pihak ketiga. Jika Anda ingin memiliki data sesuai dengan data lain yang ada, logika itu harus menjadi bagian dari fungsi Lambda yang mengimpor data dari sumber pihak ketiga.

Bermigrasi ke AWS AppConfig dari layanan konfigurasi lama dan buatan sendiri

Jika Anda sudah mulai menggunakan AWS AppConfig dan masih memiliki data konfigurasi lama atau tanda fitur di sistem lain, Anda dapat menggunakan proses yang dijelaskan sebelumnya dalam topik ini untuk bermigrasi dari sistem lama Anda dan ke sistem lama. AWS AppConfig Anda dapat membuat ekstensi yang menarik data dari sistem lama Anda dan menerapkannya. AWS AppConfig Menggunakan dengan AWS AppConfig cara ini memberi Anda semua kontrol dan manfaat pagar pembatas keamanan sambil tetap menggunakan penyimpanan data lama Anda.

Menerapkan 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 membuat profil konfigurasi.
3. Cache salinan data sehingga siap untuk diambil oleh aplikasi Anda. Salinan cache ini disebut data yang digunakan.

Anda dapat mengurangi situasi di mana penerapan data konfigurasi menyebabkan kesalahan dalam aplikasi Anda dengan menggunakan kombinasi strategi AWS AppConfig penerapan dan rollback otomatis berdasarkan alarm Amazon. CloudWatch Strategi penerapan memungkinkan Anda untuk secara perlahan melepaskan perubahan pada lingkungan produksi selama beberapa menit atau jam. Setelah dikonfigurasi, jika satu atau beberapa CloudWatch alarm masuk ke status alarm selama penerapan, AWS AppConfig secara otomatis memutar kembali data konfigurasi Anda ke versi sebelumnya. Untuk informasi selengkapnya tentang strategi penerapan, lihatBekerja dengan strategi penyebaran. Untuk informasi selengkapnya tentang rollback otomatis, lihat. Memantau penyebaran untuk rollback otomatis

Topik

- Bekerja dengan strategi penyebaran
- Men-deploy konfigurasi
- Menerapkan AWS AppConfig konfigurasi menggunakan CodePipeline
- Mengembalikan konfigurasi

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>Tipe 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 penambaha n 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>
</table>
</td></tr>
</table>

Pengaturan	Deskripsi

Waktu berlalu	Kemajuan penerapan
6 jam	60%
8 jam	80%
10 jam	100%

- Eksponensial: Untuk tipe ini, AWS AppConfig memproses deployment secara eksponensial menggunakan rumus berikut: `G*(2^N)`. 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.

Pengaturan	Deskripsi
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 deployment	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.
Waktu menanam	Pengaturan ini menentukan jumlah AWS AppConfig monitor waktu untuk CloudWatch alarm Amazon setelah konfigurasi dikerahka n ke 100% targetnya, sebelum mempertim bangkan penerapan selesai. Jika alarm dipicu selama waktu ini AWS AppConfig , putar kembali penerapan. Anda harus mengonfig urasi izin AWS AppConfig untuk memutar kembali berdasarkan CloudWatch alarm. Untuk informasi selengkapnya, lihat Konfigurasikan izin untuk rollback otomatis.

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

Topik

- Menggunakan strategi penyebaran yang telah ditentukan
- Buat strategi deployment

Menggunakan 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 deployment	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. 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

Strategi deployment	Deskripsi
	dengan praktik AWS terbaik untuk penerapan konfigurasi.
AppConfig.AllAtOnce	Cepat: Strategi ini segera menyebarkan konfigurasi ke semua target. Sistem memonitor CloudWatch 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.
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 data konfigurasi tanpa Agen AWS AppConfig. 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 Konsol Manajemen AWS, the AWS CLI, atau SDK. Memulai penerapan dalam AWS AppConfig panggilan operasi StartDeploymentAPI. Panggilan ini mencakup AWS AppConfig aplikasi, lingkungan, profil konfigurasi, dan (opsional) versi data konfigurasi yang akan digunakan. IDs 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.

Menerapkan 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
```

Menerapkan AWS AppConfig konfigurasi menggunakan 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

Mengembalikan konfigurasi

Selama penerapan, Anda dapat mengurangi situasi di mana data konfigurasi yang salah atau salah menyebabkan kesalahan dalam aplikasi Anda dengan menggunakan rollback otomatis (jika alarm dipicu selama penerapan) atau dengan mengembalikan data konfigurasi ke versi sebelumnya (jika penerapan berhasil diselesaikan).

Untuk rollback otomatis, Anda dapat menggunakan kombinasi strategi AWS AppConfig penyebaran dan alarm Amazon. CloudWatch Setelah dikonfigurasi, jika satu atau beberapa CloudWatch alarm masuk ke `ALARM` status selama penerapan, AWS AppConfig secara otomatis memutar kembali data konfigurasi Anda ke versi sebelumnya, sehingga mencegah pemadaman aplikasi atau kesalahan. Untuk memulai, lihat Konfigurasikan izin untuk rollback otomatis.

> **Note**
>
> Anda juga dapat memutar kembali konfigurasi dengan memanggil operasi StopDeploymentAPI saat penerapan masih berlangsung.

Untuk penerapan yang berhasil diselesaikan, AWS AppConfig juga mendukung pengembalian data konfigurasi ke versi sebelumnya dengan menggunakan `AllowRevert` parameter dengan operasi API. StopDeployment Untuk beberapa pelanggan, kembali ke konfigurasi sebelumnya setelah penerapan yang berhasil menjamin data akan sama seperti sebelum penerapan. Mengembalikan juga mengabaikan monitor alarm, yang dapat mencegah kemajuan maju selama keadaan darurat aplikasi.

> **Important**
>
> Jika Anda memanggil `StopDeployment` dengan `AllowRevert` parameter diaktifkan, AWS AppConfig akan mengembalikan penerapan hanya jika penerapan berhasil dalam 72 jam terakhir. Setelah 72 jam, penyebaran tidak dapat lagi dikembalikan. Anda harus membuat penerapan baru.

Berikut adalah rincian `StopDeployment` fungsionalitas berdasarkan situasi yang berbeda.

1. Jika `StopDeployment` dipanggil pada penerapan yang sedang berlangsung, status penerapan yang dihasilkan akan menjadi. `ROLLED_BACK`
2. Jika `StopDeployment` (with`AllowRevert`) dipanggil pada penerapan yang sedang berlangsung, status penerapan yang dihasilkan akan menjadi. `ROLLED_BACK`
3. Jika `StopDeployment` dipanggil pada penerapan yang selesai, a `BadRequestException` akan dilemparkan.
4. Jika `StopDeployment` (with`AllowRevert`) dipanggil pada penerapan yang selesai, status penerapan yang dihasilkan akan menjadi. `REVERTED`
5. Jika `StopDeployment` (with`AllowRevert`) dipanggil pada penerapan selesai setelah 72 jam, a `BadRequestException` akan dilemparkan.

Anda dapat menggunakan AWS CLI untuk memanggil StopDeploymentoperasi dengan `AllowRevert` parameter. Berikut adalah contoh AWS CLI perintah yang menyertakan `AllowRevert` parameter.

```
aws appconfig stop-deployment \
    --application-id 339ohji \
    --environment-id 54j1r29 \
    --deployment-number 2 \
    --allow-revert
```

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. Kami menyarankan Anda menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi. Agen (atau ekstensi AWS AppConfig Agen Lambda untuk lingkungan komputasi Lambda) mengelola serangkaian panggilan API dan token sesi atas nama Anda. Dari tingkat tinggi, prosesnya bekerja sebagai berikut:

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

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

1. 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.
2. Saat memanggil`StartConfigurationSession`, kode Anda mengirimkan pengenal (ID atau nama) dari AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi yang dilacak sesi.
3. Sebagai tanggapan, AWS AppConfig berikan `InitialConfigurationToken` untuk diberikan kepada klien sesi dan digunakan saat pertama kali memanggil `GetLatestConfiguration` sesi itu.
4. Saat meneleponGetLatestConfiguration, kode klien Anda mengirimkan `ConfigurationToken` nilai terbaru yang dimilikinya dan diterima sebagai tanggapan:
   - `NextPollConfigurationToken`: `ConfigurationToken` nilai yang akan digunakan pada panggilan berikutnya ke`GetLatestConfiguration`.
   - Konfigurasi: data terbaru yang ditujukan untuk sesi tersebut. Ini mungkin kosong jika klien sudah memiliki versi konfigurasi terbaru.

> **Note**
>
> Mengambil data konfigurasi dari terpisah Akun AWS tidak didukung.

Daftar Isi

- Apa itu AWS AppConfig Agen?
- Cara menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi
- AWS AppConfig pertimbangan penggunaan browser dan seluler
- Mengambil data konfigurasi tanpa Agen AWS AppConfig

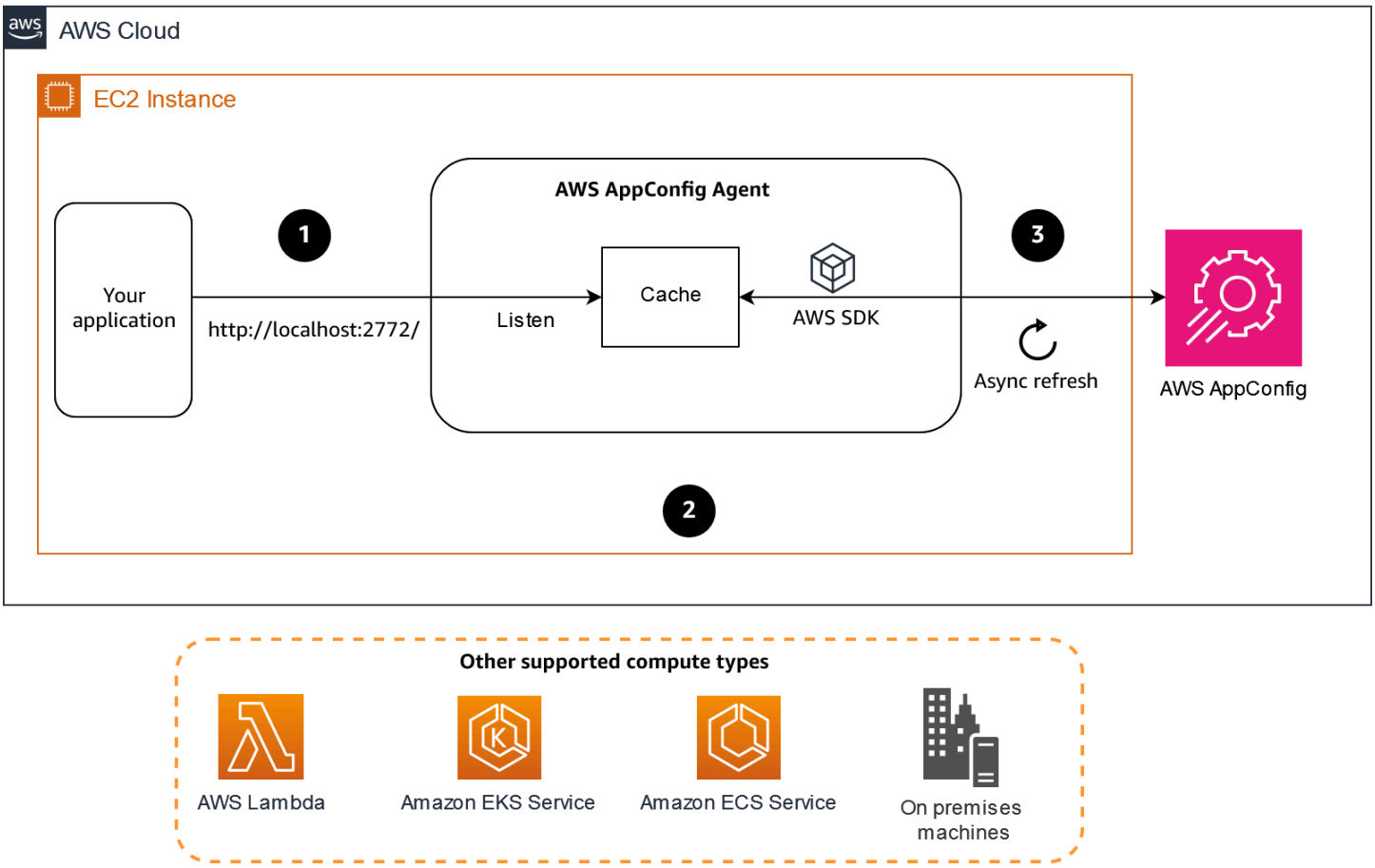
Apa itu AWS AppConfig Agen?

AWS AppConfig Agen adalah proses yang dikembangkan dan dikelola Amazon untuk mengambil data konfigurasi dari. AWS AppConfig Dengan agen, Anda dapat menyimpan data konfigurasi secara lokal dan secara asinkron melakukan polling layanan pesawat AWS AppConfig data untuk pembaruan. caching/polling Proses ini memastikan bahwa data konfigurasi Anda selalu tersedia untuk aplikasi Anda sambil meminimalkan latensi dan biaya. Agen bukan satu-satunya cara untuk mengambil data konfigurasi AWS AppConfig, tetapi ini adalah cara yang disarankan. Agen meningkatkan pemrosesan dan manajemen aplikasi dengan cara berikut:

- Agen memanggil AWS AppConfig atas nama Anda dengan menggunakan prinsipal AWS Identity and Access Management (IAM) dan mengelola cache data konfigurasi lokal. Dengan mengambil 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 layanan 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.

> **Note**
>
> 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.

Diagram berikut menunjukkan cara kerja AWS AppConfig Agen.



1. Aplikasi Anda meminta data konfigurasi dari agen.
2. Agen mengembalikan data dari cache dalam memori.
3. Agen secara asinkron melakukan polling AWS AppConfig layanan untuk data konfigurasi terbaru pada irama yang telah ditentukan sebelumnya. Data konfigurasi terbaru selalu disimpan dalam cache di memori.

Cara menggunakan AWS AppConfig Agen untuk mengambil data konfigurasi

AWS AppConfig Agen adalah metode yang direkomendasikan untuk mengambil flag AWS AppConfig fitur atau data konfigurasi formulir bebas. Agen didukung pada semua bentuk AWS Compute termasuk Amazon EC2, Amazon ECS, Amazon EKS, dan Lambda. Setelah Anda menyelesaikan penyiapan agen awal, menggunakan agen untuk mengambil data konfigurasi lebih sederhana daripada menelepon AWS AppConfig APIs langsung. Agen secara otomatis menerapkan praktik terbaik dan dapat menurunkan biaya penggunaan Anda AWS AppConfig sebagai akibat dari lebih sedikit panggilan API untuk mengambil konfigurasi.

> **Note**
>
> Mengambil data konfigurasi dari terpisah Akun AWS tidak didukung.

Topik

- Menggunakan AWS AppConfig agen dengan AWS Lambda
- Menggunakan AWS AppConfig Agen dengan Amazon EC2 dan mesin lokal
- Menggunakan AWS AppConfig Agen dengan Amazon ECS dan Amazon EKS
- Mengambil flag fitur dasar dan multi-varian
- Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan
  - Mengkonfigurasi AWS AppConfig Agen untuk mengambil konfigurasi dari beberapa akun
  - Mengkonfigurasi AWS AppConfig Agen untuk menulis salinan konfigurasi ke disk
- Menghasilkan klien menggunakan spesifikasi OpenAPI
- Bekerja dengan mode pengembangan lokal AWS AppConfig Agen

Menggunakan AWS AppConfig agen dengan AWS 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 adalah kemampuan AWS Systems Manager. 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.

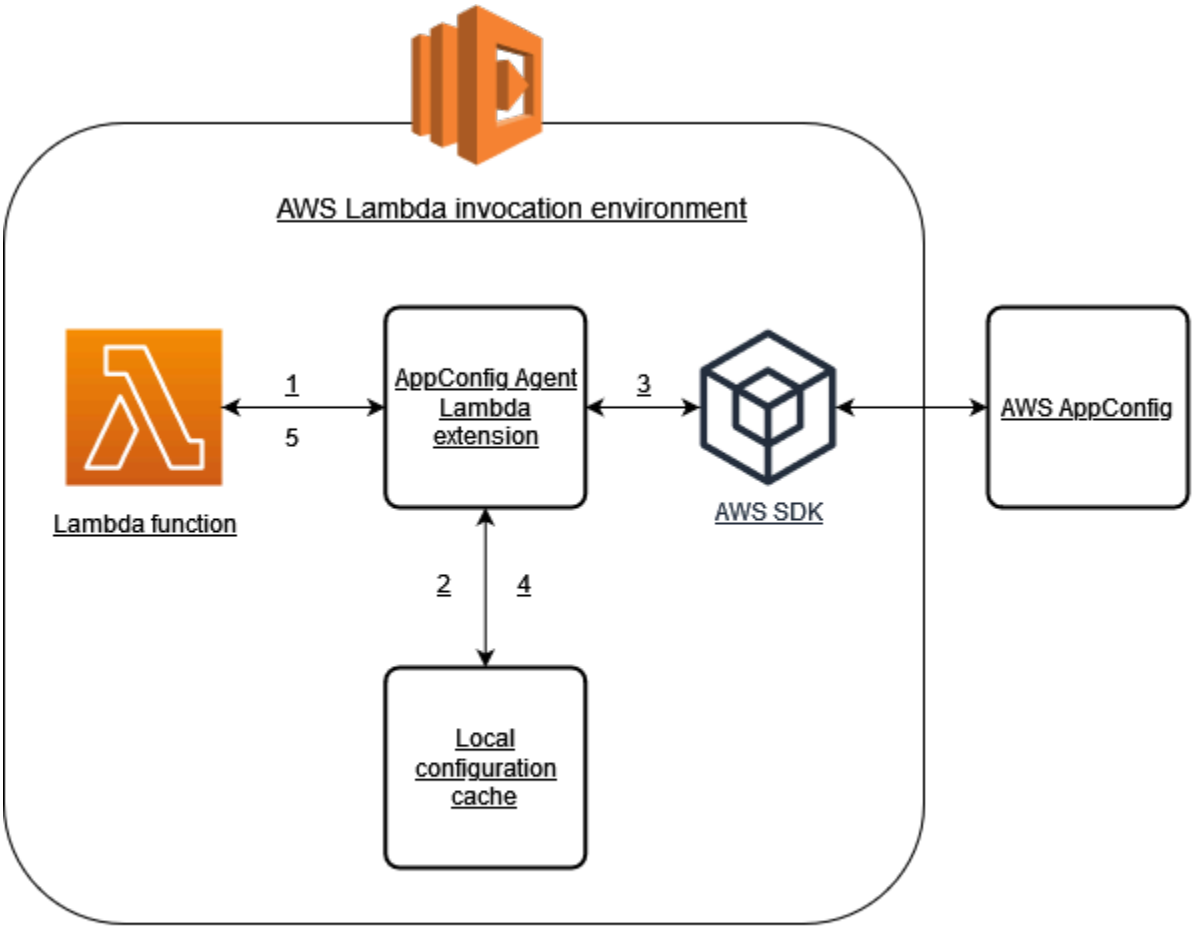
Topik

- Memahami cara kerja AWS AppConfig ekstensi Agen Lambda
- Menambahkan ekstensi AWS AppConfig Agen Lambda
- Mengkonfigurasi ekstensi AWS AppConfig Agen Lambda
- Memahami versi ekstensi AWS AppConfig Agen Lambda yang tersedia

Memahami cara kerja AWS AppConfig ekstensi Agen Lambda

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 dan. StartConfigurationSessionGetLatestConfiguration

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.

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.

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 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 AWS Region tempat Anda membuat Lambda.

- platform x86-64
- ARM64 platform

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

Important

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)
```

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 sebagian atau ARNs 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
> - Tabel berikut mencakup kolom Nilai sampel. Tergantung pada resolusi monitor Anda, Anda mungkin perlu menggulir ke bagian bawah tabel dan kemudian gulir ke kanan untuk melihat kolom.

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`AWS_APPCO NFIG_EXTE`	Variabel lingkungan ini menentukan port di mana server HTTP	2772	2772

Variabel lingkungan	Detail	Nilai default	Nilai sampel
NSION_HTT P_PORT	lokal yang menghosti ng ekstensi berjalan.		
AWS_APPCO NFIG_EXTE NSION_LOG _LEVEL	Variabel lingkunga n ini menentukan tingkat detail yang dicatat agen. Setiap level mencakup level saat ini dan semua level yang lebih tinggi. Nilainya tidak peka huruf besar/ kecil. Dari sebagian besar hingga yang paling tidak detail, level log adalah: tracedebug,info,,wa: dannone. traceLog mencakup informasi terperinci, termasuk informasi waktu, tentang agen.	info	jejak debug info menegur kesalahan mematikan none
AWS_APPCO NFIG_EXTE NSION_MAX _CONNECTIONS	Variabel lingkunga n ini mengonfigurasi jumlah maksimum koneksi yang digunakan ekstensi untuk mengambil konfigurasi dari. AWS AppConfig	3	3

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS`	Variabel lingkungan ini mengontrol seberapa sering agen melakukan polling AWS AppConfig untuk data konfigurasi yang diperbarui. 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	45 45-an 5m 1 jam

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`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.	3000ms	3000 300ms 5s

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`AWS_APPCONFIG_EXTENSION_PREFETCH_LIST`	Variabel lingkungan ini menentukan data konfigurasi permintaan agen AWS AppConfig segera setelah dimulai. Beberapa pengidentifikasi konfigurasi dapat disediakan dalam daftar yang dipisahkan koma. Mengambil data konfigurasi dari AWS AppConfig dapat mengurangi waktu mulai dingin fungsi Anda secara signifikan.	Tidak ada	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Config1, MyEnv ::Config2 MyApp MyEnv
`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.	Tidak ada	header: nilai h1: v1, h2: v2

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`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` URLs didukung.	Tidak ada	http://localhost:7474 https://my-proxy.example.com
`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	arn:aws:iam: :123456789012:peran/MyRole
`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`	Variabel lingkungan ini menentukan id eksternal untuk digunakan bersama dengan peran ARN yang diasumsikan.	Tidak ada	MyExternalId
`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada	AWSAppConfigAgentSession

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`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	us-east-1 eu-west-1
`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. Untuk informasi lebih lanjut tentang fitur ini, lihat Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan.	Tidak ada	Saat menggunakan AWS AppConfig konfigurasi sebagai manifes:`MyApp:MyEnv:MyManifestConfig` . Saat memuat manifes dari disk: `file:/path/to/manifest.json`

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesaikan startup.	true	true SALAH

Memahami 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 (Graviton2). ARM64 Agar berfungsi dengan baik, fungsi Lambda Anda harus dikonfigurasi untuk menggunakan Nama Sumber Daya Amazon (ARN) tertentu untuk AWS Region tempat saat ini di-host. Anda dapat melihat AWS Region 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 `StartConfigurationSession` dan `GetLatestConfiguration` APIs sebagai gantinya. Jika Anda menggunakan versi ekstensi AWS AppConfig Agen Lambda yang dibuat setelah 28 Januari 2022, Anda perlu mengonfigurasi izin ke yang baru. APIs Untuk informasi selengkapnya, lihat Mengambil data konfigurasi tanpa Agen AWS AppConfig.
- 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

- Menemukan nomor versi ekstensi Lambda Anda
- platform x86-64
- ARM64 platform
- Versi ekstensi yang lebih lama

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
2.0.11962.0	02/20/2026	Peningkatan dukungan lingkungan, penyempurnaan kecil, dan perbaikan bug.
2.0.8693	11/20/2025	Peningkatan dukungan lingkungan, penyempurnaan kecil, dan perbaikan bug. Ditambahkan dukungan untuk berikut Wilayah AWS • Asia Pasifik (Taipei), ap-timur-2 • Asia Pasifik (Selandia Baru), ap-tenggara - 6 • Asia Pasifik (Thailand), ap-tenggara - 7 • Meksiko (Tengah), mx-central-1
2.0.2037	05/12/2025	`/ping`Jalur yang ditambahkan, yang memperlihatkan pemeriksaan kesehatan sederhana yang mengembalikan versi agen itu. Juga

Versi	Tanggal peluncuran	Catatan
		termasuk perangkat tambahan kecil dan perbaikan bug.
2.0.1079	12/12/2024	Penyempurnaan kecil dan perbaikan bug.
2.0.719	08/08/2024	Perangkat tambahan kecil dan perbaikan bug.
2.0.678	07/23/2024	Penyempurnaan untuk mendukung target, varian, dan split flag fitur. Untuk informasi selengkapnya, lihat Membuat bendera fitur multi-varian.
2.0.501	07/01/2024	Perangkat tambahan kecil dan perbaikan bug.

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) AWS Region il-central-1.

Versi	Tanggal peluncuran	Catatan
2.0.165	02/21/2023	Perbaikan bug minor. Tidak lagi membatasi penggunaan ekstensi ke versi runtime tertentu melalui konsol. AWS Lambda Ditambahkan dukungan untuk yang berikut Wilayah AWS: • Timur Tengah (UEA), me-central-1 • 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.

Versi	Tanggal peluncuran	Catatan
2.0.58	05/03/2022	Peningkatan dukungan untuk prosesor Graviton2 (ARM64) di Lambda.
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 AWS Region tempat saat ini di-host.

1. Masuk ke Konsol Manajemen AWS 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 yang diperlukan URLs untuk versi itu.

platform x86-64

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

Versi 2.0.11962.0

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:296`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:252`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:359`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:348`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:239`

Region	ARN
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:147
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:270
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:195
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:278
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:217
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:248
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:342
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:226
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:189

Region	ARN
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:219`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:221`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:228`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:245`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:248`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:247`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:233`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:288`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:231`

Region	ARN
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:163
Asia Pasifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:136
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:264
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:192
Asia Pasifik (Selandia Baru)	arn:aws:lambda:ap-southeast-6:381491832265:layer:AWS-AppConfig-Extension:58
Asia Pasifik (Thailand)	arn:aws:lambda:ap-southeast-7:851725616657:layer:AWS-AppConfig-Extension:109
Asia Pasifik (Taipei)	arn:aws:lambda:ap-east-2:730335625313:layer:AWS-AppConfig-Extension:118
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:302
Meksiko (Tengah)	arn:aws:lambda:mx-central-1:891376990304:layer:AWS-AppConfig-Extension:115

Region	ARN
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:234`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:168`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:206`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:244`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:184`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:182`

ARM64 platform

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

Versi 2.0.11962.0

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:229`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:204`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:236`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:250`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:159`
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:137`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:213`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:153`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:216`

Region	ARN
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:169
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:167
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:201
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:154
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:150
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:156
Asia Pasifik (Taipei)	arn:aws:lambda:ap-east-2:730335625313:layer:AWS-AppConfig-Extension-Arm64:92
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:198
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:156

Region	ARN
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:162`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:185`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:231`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:168`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:148`
Asia Pasifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:111`
Asia Pasifik (Selandia Baru)	`arn:aws:lambda:ap-southeast-6:381491832265:layer:AWS-AppConfig-Extension-Arm64:48`
Asia Pasifik (Thailand)	`arn:aws:lambda:ap-southeast-7:851725616657:layer:AWS-AppConfig-Extension-Arm64:108`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:206`

Region	ARN
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:150
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:190
Meksiko (Tengah)	arn:aws:lambda:mx-central-1:891376990304:layer:AWS-AppConfig-Extension-Arm64:114
Africa (Cape Town)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:162
Timur Tengah (UAE)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:162
Timur Tengah (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:172
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:151
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:141
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:143

Region	ARN
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:130`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:128`

Versi ekstensi yang lebih lama

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

Topik

- Versi ekstensi yang lebih lama (platform x86-64)
- Versi ekstensi yang lebih lama (ARM64 platform)

Versi ekstensi yang lebih lama (platform x86-64)

Daftar tabel berikut ARNs dan Wilayah AWS untuk versi lama dari ekstensi AWS AppConfig Agen Lambda dikembangkan untuk platform x86-64.

Tanggal diganti dengan ekstensi yang lebih baru: 02/17/2026

Versi 2.0.8693

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:279`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:235`

Region	ARN
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:348`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:335`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:228`
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:130`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:261`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:178`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:261`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:207`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:235`

Region	ARN
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:333`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:215`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:176`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:205`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:203`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:217`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:228`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:239`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:234`

Region	ARN
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:224
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:272
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:222
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:152
Asia Pasifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:127
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:248
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:179
Asia Pasifik (Selandia Baru)	arn:aws:lambda:ap-southeast-6:381491832265:layer:AWS-AppConfig-Extension:41
Asia Pasifik (Thailand)	arn:aws:lambda:ap-southeast-7:851725616657:layer:AWS-AppConfig-Extension:98

Region	ARN
Asia Pasifik (Taipei)	`arn:aws:lambda:ap-east-2:730335625313:layer:AWS-AppConfig-Extension:100`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:288`
Meksiko (Tengah)	`arn:aws:lambda:mx-central-1:891376990304:layer:AWS-AppConfig-Extension:98`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:225`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:155`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:195`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:227`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:184`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:182`

Tanggal diganti dengan ekstensi yang lebih baru: 11/20/2025

Versi 2.0.2037

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:207`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:162`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:258`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:262`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:152`
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:57`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:189`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:106`

Region	ARN
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:189`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:133`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:162`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:259`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:140`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:102`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:133`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:131`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:142`

Region	ARN
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:155`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:165`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:159`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:156`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:199`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:150`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:78`
Asia Pasifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:55`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:175`

Region	ARN
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:104`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:215`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:152`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:81`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:120`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:154`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:110`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:109`

Tanggal diganti dengan ekstensi yang lebih baru: 05/20/2025

Versi 2.0.1079

Region	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:174
AS Timur (Ohio)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:133
AS Barat (California Utara)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:223
AS Barat (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:230
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:123
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:159
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:77
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:160

Region	ARN
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:121
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:133
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:225
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:111
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:74
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:106
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:104
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:113
Asia Pacific (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:126

Region	ARN
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:136`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:130`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:134`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:165`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:121`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:49`
Asia Pasifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:26`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:146`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:75`

Region	ARN
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:179`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:123`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:52`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:91`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:125`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:80`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:80`

Tanggal diganti dengan ekstensi yang lebih baru: 12/12/2024

Versi 2.0.719

Region	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:173
AS Timur (Ohio)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:132
AS Barat (California Utara)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:221
AS Barat (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:229
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:121
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:158
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:75
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:159

Region	ARN
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:120
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:132
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:224
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:110
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:72
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:104
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:102
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:112
Asia Pacific (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:125

Region	ARN
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:135
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:129
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:132
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:164
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:120
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:48
Asia Pasifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:25
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:145
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:74

Region	ARN
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:178`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:122`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:50`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:90`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:124`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:79`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:79`

Tanggal diganti dengan ekstensi yang lebih baru: 08/08/2024

Versi 2.0.678

Region	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:167
AS Timur (Ohio)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:126
AS Barat (California Utara)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:213
AS Barat (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:223
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:116
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:21
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:152
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:70
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:153

Region	ARN
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:114
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:126
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:218
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:104
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:67
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:99
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:97
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:106
Asia Pacific (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:119

Region	ARN
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:129
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:123
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:127
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:158
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:114
Asia Pasifik (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:42
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:139
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:68
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:172

Region	ARN
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:116`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:45`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:84`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:118`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:73`

Tanggal diganti dengan ekstensi yang lebih baru: 07/23/2024

Versi 2.0.501

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:153`

Region	ARN
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:112`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:195`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:210`
Canada (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:101`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:136`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:53`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:144`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:99`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:111`

Region	ARN
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:201`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:89`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:50`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:85`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:83`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:91`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:104`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:114`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:107`

Region	ARN
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:112`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:142`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:98`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:26`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:125`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:53`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:155`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:102`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:28`

Region	ARN
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:68`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:103`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:59`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:59`

Tanggal diganti dengan ekstensi yang lebih baru: 07/01/2024

Versi 2.0.358

Region	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`

Region	ARN
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
Europe (Zurich)	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
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83

Region	ARN
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`
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`

Region	ARN
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`
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`
Africa (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 (UAE)	`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`

Region	ARN
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`

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

Versi 2.0.181

Region	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`
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`

Region	ARN
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93`
Europe (Zurich)	`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`
Europe (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`

Region	ARN
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
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

Region	ARN
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`
Africa (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 (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34`
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

Region	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`
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`
Europe (Zurich)	`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`

Region	ARN
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
Europe (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
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

Region	ARN
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`
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`

Region	ARN
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71`
Timur Tengah (UAE)	`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

Region	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`

Region	ARN
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
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
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54

Region	ARN
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`
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`

Region	ARN
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`
Africa (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

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`

Region	ARN
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`
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`

Region	ARN
Europe (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`
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`

Region	ARN
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`
Africa (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

Region	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
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50

Region	ARN
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97
Europe (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
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50

Region	ARN
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`
Africa (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

Region	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
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48

Region	ARN
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86
Europe (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
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45

Region	ARN
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`
Africa (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 (ARM64 platform)

Daftar tabel berikut ARNs dan Wilayah AWS untuk versi lama dari ekstensi AWS AppConfig Agen Lambda yang dikembangkan untuk platform. ARM64

Tanggal diganti dengan ekstensi yang lebih baru: 02/17/2026

Versi 2.0.8693

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:212`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:187`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:225`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:237`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:148`
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:120`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:204`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:136`

Region	ARN
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:199`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:159`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:154`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:192`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:143`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:137`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:145`
Asia Pasifik (Taipei)	`arn:aws:lambda:ap-east-2:730335625313:layer:AWS-AppConfig-Extension-Arm64:74`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:181`

Region	ARN
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:147
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:149
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:176
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:215
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:159
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:137
Asia Pasifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:102
Asia Pasifik (Selandia Baru)	arn:aws:lambda:ap-southeast-6:381491832265:layer:AWS-AppConfig-Extension-Arm64:31
Asia Pasifik (Thailand)	arn:aws:lambda:ap-southeast-7:851725616657:layer:AWS-AppConfig-Extension-Arm64:97

Region	ARN
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:190
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:137
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:176
Meksiko (Tengah)	arn:aws:lambda:mx-central-1:891376990304:layer:AWS-AppConfig-Extension-Arm64:97
Africa (Cape Town)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:153
Timur Tengah (UAE)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:151
Timur Tengah (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:155
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:138
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:127

Region	ARN
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:125`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:130`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:128`

Tanggal diganti dengan ekstensi yang lebih baru: 11/20/2025

Versi 2.0.2037

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:140`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:114`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:135`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:164`

Region	ARN
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:72
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:47
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:132
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:64
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:127
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:85
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:81
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:118
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:68

Region	ARN
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:63`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:70`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:108`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:73`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:74`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:108`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:142`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:87`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:63`

Region	ARN
Asia Pasifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:30
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:117
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:62
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:103
Africa (Cape Town)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:80
Timur Tengah (UAE)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:76
Timur Tengah (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:82
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:64
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:55

Region	ARN
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:53`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:56`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:55`

Tanggal diganti dengan ekstensi yang lebih baru: 05/20/2025

Versi 2.0.1079

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:107`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:85`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:100`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:132`

Region	ARN
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:43
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:18
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:102
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:35
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:98
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:73
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:52
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:84
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:39

Region	ARN
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:35`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:41`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:79`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:44`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:45`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:86`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:108`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:58`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:34`

Region	ARN
Asia Pasifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:88`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:33`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:67`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:51`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:47`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:53`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:35`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:28`

Region	ARN
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:26`

Tanggal diganti dengan ekstensi yang lebih baru: 12/12/2024

Versi 2.0.678

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:106`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:84`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:98`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:131`

Region	ARN
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:41
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:17
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:101
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:33
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:97
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:72
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:51
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:83
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:38

Region	ARN
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:33`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:40`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:78`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:43`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:44`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:84`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:107`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:57`
Asia Pacific (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:33`

Region	ARN
Asia Pasifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:87`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:32`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:66`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:50`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:46`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:52`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:33`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:26`

Region	ARN
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:24`
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:25`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:25`

Tanggal diganti dengan ekstensi yang lebih baru: 08/08/2024

Versi 2.0.678

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:100`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:78`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:90`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:125`

Region	ARN
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:11`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:36`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:95`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:28`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:91`
Eropa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:66`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:45`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:77`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:32`

Region	ARN
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:28`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:34`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:72`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:37`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:38`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:79`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:101`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:51`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:27`

Region	ARN
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:81`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:26`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:60`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:44`
Timur Tengah (UAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:40`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:46`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:28`
China (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:21`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:19`

Region	ARN
AWS GovCloud (AS-Timur)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:19`
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:19`

Tanggal diganti dengan ekstensi yang lebih baru: 07/23/2024

Versi 2.0.501

Region	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:86`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:64`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:72`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:112`
Kanada Barat (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:1`

Region	ARN
Kanada (Pusat)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:21
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:79
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:11
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:82
Eropa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:51
Eropa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:30
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:60
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:17
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:11

Region	ARN
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:19`
Asia Pacific (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:57`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:22`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:22`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:64`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:85`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:35`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:11`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:67`

Region	ARN
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:11
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:43
Africa (Cape Town)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:30
Timur Tengah (UAE)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:24
Timur Tengah (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:31
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:11
China (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:7
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:5
AWS GovCloud (AS-Timur)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:5

Region	ARN
AWS GovCloud (AS-Barat)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:5`

Tanggal diganti dengan ekstensi yang lebih baru: 07/01/2024

Versi 2.0.358

Region	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`

Region	ARN
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5`
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`
Europe (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`

Region	ARN
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16
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

Region	ARN
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11`
Timur Tengah (UAE)	`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`

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

Versi 2.0.181

Region	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`

Region	ARN
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`
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`
Europe (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`

Region	ARN
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`
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`
Africa (Cape Town)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1`

Region	ARN
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

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

Region	ARN
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

Region	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`

Region	ARN
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`
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

Region	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
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

Region	ARN
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

Region	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`
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`

Region	ARN
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 AWS AppConfig Agen dengan Amazon EC2 dan mesin lokal

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 1: (Diperlukan) Membuat sumber daya dan mengonfigurasi izin
- Langkah 2: (Diperlukan) Menginstal dan memulai AWS AppConfig Agen di instans Amazon EC2
- Langkah 3: (Opsional, tetapi disarankan) Mengirim file log ke CloudWatch Log
- Langkah 4: (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon EC2
- 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:

JSON

```
{
    "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. 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 x86_64 dan arsitektur. 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 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 Penskalaan Otomatis 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 beberapa instance dengan menggunakan Run Command, yang merupakan alat di 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 salah satu perintah berikut dengan izin Administrator:

   x86_64

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

   ARM64

   ```
   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, alat di 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.

> **Note**
>
> Tabel berikut mencakup kolom Nilai sampel. Tergantung pada resolusi monitor Anda, Anda mungkin perlu menggulir ke bagian bawah tabel dan kemudian gulir ke kanan untuk melihat kolom.

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`ACCESS_TOKEN`	Variabel lingkungan ini mendefinisikan token yang	Tidak ada	MyAccessToken

Variabel lingkungan	Detail	Nilai default	Nilai sampel

	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>`		

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`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 memprakti kkan prinsip hak istimewa paling sedikit dengan izin sistem file Anda. Untuk informasi selengkap nya, lihat Keamanan	Tidak ada	/path/to/backups

Variabel lingkungan	Detail	Nilai default	Nilai sampel
	di AWS AppConfig.		
`HTTP_PORT`	Variabel lingkungan ini menentukan port di mana server HTTP untuk agen berjalan.	2772	2772
`HTTP_HOST`	Variabel HTTP_HOST mengontrol bagaimana AWS AppConfig Agen mengikat ke antarmuka jaringan. Perilaku mengikat berbeda berdasarkan lingkungan runtime untuk memastikan keamanan dan aksesibilitas yang optimal.	ECS, EKS • Pengikatan default: Semua antarmuka jaringan (0.0.0.0) EC2 dan on-prem • Pengikatan default: hanya localhost • IPv4 alamat: 127.0.0. 1:2772 • IPv6 alamat: [::1] :2772	Opsi Konfigurasi Kustom. Anda dapat mengganti perilaku default menggunakan nilai-nilai ini: • `all`(mengikat ke semua antarmuka) • `localhost` (secara eksplisit mengikat ke antarmuka localhost) • Alamat IP tertentu (mis.`192.168.1.1`) • Nama host kustom

Variabel lingkungan	Detail	Nilai default	Nilai sampel
LOG_LEVEL	Variabel lingkungan ini menentukan tingkat detail yang dicatat agen. Setiap level mencakup level saat ini dan semua level yang lebih tinggi. Nilainya tidak peka huruf besar/kecil. Dari sebagian besar hingga yang paling tidak detail, level log adalah: `trace`,`debug`,`info`,,`wa`: dan`none`. `traceLog` mencakup informasi terperinci, termasuk informasi waktu, tentang agen.	info	jejak debug info menegur kesalahan mematikan none
LOG_PATH	Lokasi disk tempat log ditulis. Jika tidak ditentukan, log ditulis ke stderr.	Tidak ada	/path/to/logs/agent.log

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk memanfaatkan fitur per-konfigurasi tambahan seperti pengambilan multi-akun dan menyimpan konfigurasi ke disk. Untuk informasi lebih lanjut tentang fitur ini, lihat Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan.	Tidak ada	Saat menggunakan AWS AppConfig konfigurasi sebagai manifes:`MyApp:MyEnv:MyManifestConfig` . Saat memuat manifes dari disk: `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Variabel lingkungan ini mengonfigurasi jumlah maksimum koneksi yang digunakan agen untuk mengambil konfigurasi dari. AWS AppConfig	3	3

Variabel lingkungan	Detail	Nilai default	Nilai sampel
POLL_INTERVAL	Variabel lingkungan ini mengontrol seberapa sering agen melakukan polling AWS AppConfig untuk data konfigurasi yang diperbarui. 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	45 45-an 5m 1 jam

Variabel lingkungan	Detail	Nilai default	Nilai sampel
PREFETCH_LIST	Variabel lingkungan ini menentukan data konfigurasi permintaan agen AWS AppConfig segera setelah dimulai. Beberapa pengidentifikasi konfigurasi dapat disediakan dalam daftar yang dipisahkan koma.	Tidak ada	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Config1,MyEnv ::Config2 MyApp MyEnv

Variabel lingkungan	Detail	Nilai default	Nilai sampel
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	true SALAH
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.	Tidak ada	header: nilai h1: v1, h2: v2

Variabel lingkungan	Detail	Nilai default	Nilai sampel
PROXY_URL	Variabel lingkungan ini menentukan URL proxy yang akan digunakan untuk koneksi dari agen ke Layanan AWS, termasuk AWS AppConfig. HTTPSdan HTTP URLs didukung.	Tidak ada	http://localhost:7474 https://my-proxy.example.com

Variabel lingkungan	Detail	Nilai default	Nilai sampel
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	3000ms	3000 3000ms 5s

Variabel lingkungan	Detail	Nilai default	Nilai sampel
	unit tidak ditentukan, agen default ke milidetik. Sebagai contoh, 5000, 5000 ms dan 5s menghasilkan nilai batas waktu permintaan yang sama.		
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	arn:aws:iam: :123456789012:peran/MyRole
ROLE_EXTERNAL_ID	Variabel lingkungan ini menentukan ID eksternal untuk digunakan dengan peran ARN yang diasumsikan.	Tidak ada	MyExternalId
ROLE_SESSION_NAME	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada	AWSAppConfigAgentSession

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`SERVICE_REGION`	Variabel lingkungan ini menentukan alternatif AWS Region 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	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesaikan startup.	true	true SALAH

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 instance 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
```

Menggunakan AWS AppConfig Agen 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 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.

Sebelum Anda mulai

Untuk berintegrasi AWS AppConfig dengan aplikasi kontainer, 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 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.

Topik

- Memulai AWS AppConfig agen untuk integrasi Amazon ECS
- Memulai AWS AppConfig agen untuk integrasi Amazon EKS
- (Opsional) Berjalan AWS AppConfig sebagai DaemonSet di Amazon EKS
- (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS
- Mengambil data konfigurasi untuk aplikasi yang berjalan di Amazon ECS dan Amazon EKS

Memulai AWS AppConfig agen untuk integrasi Amazon ECS

Kontainer sespan AWS AppConfig Agen tersedia secara otomatis di lingkungan Amazon ECS Anda. Untuk menggunakannya, Anda harus memulainya, seperti yang dijelaskan dalam prosedur berikut.

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**
>
> Perhatikan informasi berikut.
>
> - AWS AppConfig Agen adalah proses yang berjalan lama. Sebagai praktik terbaik untuk kontainer Amazon ECS, konfigurasikan pemeriksaan kesehatan untuk kontainer Anda, secara khusus mengatur ketergantungan kontainer ke kondisi SEHAT. Untuk informasi selengkapnya, lihat ContainerDependencydi Referensi API Amazon Elastic Container Service.
> - Anda dapat menyesuaikan perilaku default AWS AppConfig Agen dengan memasukkan atau mengubah variabel lingkungan. Untuk informasi tentang variabel lingkungan yang tersedia, lihat(Opsional) Menggunakan 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 menggunakannya, Anda harus memulainya. Prosedur berikut menjelaskan cara menggunakan alat baris `kubectl` perintah Amazon EKS untuk memulai agen.

> **Note**
>
> Sebelum melanjutkan, pastikan `kubeconfig` file Anda mutakhir. 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 manifes untuk aplikasi 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 manifes penerapan 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: AWS 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 (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS.
> - Untuk*AWS Region*, tentukan AWS Region kode (misalnya,`us-west-1`) tempat AWS AppConfig Agen mengambil data konfigurasi.

3. Jalankan `kubectl` perintah berikut untuk menerapkan perubahan ke cluster Anda. Ganti *my-deployment* dengan nama manifes penerapan 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, lihat(Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS.

(Opsional) Berjalan AWS AppConfig sebagai DaemonSet di Amazon EKS

Dengan Amazon EKS, Anda dapat menjalankan AWS AppConfig Agen sebagai sespan, yang menghasilkan satu wadah agen per pod aplikasi. Atau, jika Anda mau, Anda dapat menjalankan AWS AppConfig Agen sebagai DaemonSet, yang menghasilkan satu wadah agen per node di cluster Anda.

> **Note**
>
> Jika Anda menjalankan AWS AppConfig Agen sebagai a DaemonSet, agen berjalan di pod terpisah, yang berarti Anda tidak dapat mengaksesnya dengan panggilan ke`localhost`. Anda harus menyuntikkan atau menemukan alamat IP agen pod untuk memanggilnya.

Untuk menjalankan AWS AppConfig Agen sebagai DaemonSet, buat file manifes dengan konten berikut. Ganti *`highlighted`* teks dengan detail untuk aplikasi dan lingkungan Anda. Untuk*`AWS Region`*, tentukan AWS Region kode (misalnya,`us-west-1`).

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: aws-appconfig-agent
  namespace: my_namespace
  labels:
    app: my_application_label
spec:
   selector:
    matchLabels:
      app: my_application_label
  template:
    metadata:
      labels:
```

```
        app: my_application_label
    spec:
      containers:
      - name: aws-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: AWS Region
        imagePullPolicy: IfNotPresent
      # set a high priority class to ensure the agent is running on every node
      priorityClassName: system-node-critical
```

Jalankan perintah berikut untuk menerapkan AWS AppConfig Agen DaemonSet ke cluster Anda. Ganti *aws_appconfig_agent_daemonset* dengan nama DaemonSet manifes Anda.

```
kubectl apply -f aws_appconfig_agent_daemonset.yml
```

(Opsional) 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.

> **Note**
>
> Tabel berikut mencakup kolom Nilai sampel. Tergantung pada resolusi monitor Anda, Anda mungkin perlu menggulir ke bagian bawah tabel dan kemudian gulir ke kanan untuk melihat kolom.

Variabel lingkungan	Detail	Nilai default	Nilai sampel
ACCESS_TOKEN	Variabel lingkungan ini mendefinisikan token yang	Tidak ada	MyAccessToken

Variabel lingkungan	Detail	Nilai default	Nilai sampel

	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>`		

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`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 memprakti kkan prinsip hak istimewa paling sedikit dengan izin sistem file Anda. Untuk informasi selengkap nya, lihat Keamanan	Tidak ada	/path/to/backups

Variabel lingkungan	Detail	Nilai default	Nilai sampel
	di AWS AppConfig.		
`HTTP_PORT`	Variabel lingkungan ini menentukan port di mana server HTTP untuk agen berjalan.	2772	2772
`HTTP_HOST`	Variabel HTTP_HOST mengontrol bagaimana AWS AppConfig Agen mengikat ke antarmuka jaringan. Perilaku mengikat berbeda berdasarkan lingkungan runtime untuk memastikan keamanan dan aksesibilitas yang optimal.	ECS, EKS • Pengikatan default: Semua antarmuka jaringan (0.0.0.0) EC2 dan on-prem • Pengikatan default: hanya localhost • IPv4 alamat: 127.0.0. 1:2772 • IPv6 alamat: [::1] :2772	Opsi Konfigurasi Kustom. Anda dapat mengganti perilaku default menggunakan nilai-nilai ini: • `all`(mengikat ke semua antarmuka) • `localhost` (secara eksplisit mengikat ke antarmuka localhost) • Alamat IP tertentu (mis.`192.168.1.1`) • Nama host kustom

Variabel lingkungan	Detail	Nilai default	Nilai sampel
LOG_LEVEL	Variabel lingkungan ini menentukan tingkat detail yang dicatat agen. Setiap level mencakup level saat ini dan semua level yang lebih tinggi. Nilainya tidak peka huruf besar/kecil. Dari sebagian besar hingga yang paling tidak detail, level log adalah: `trace`debug,`info`,,`wa`: dannone. `trace`Log mencakup informasi terperinci, termasuk informasi waktu, tentang agen.	info	jejak debug info menegur kesalahan mematikan none
LOG_PATH	Lokasi disk tempat log ditulis. Jika tidak ditentukan, log ditulis ke stderr.	Tidak ada	/path/to/logs/agent.log

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk memanfaatkan fitur per-konfigurasi tambahan seperti pengambilan multi-akun dan menyimpan konfigurasi ke disk. Untuk informasi lebih lanjut tentang fitur ini, lihat Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan.	Tidak ada	Saat menggunakan AWS AppConfig konfigurasi sebagai manifes:`MyApp:MyEnv:MyManifestConfig` . Saat memuat manifes dari disk: `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Variabel lingkungan ini mengonfigurasi jumlah maksimum koneksi yang digunakan agen untuk mengambil konfigurasi dari. AWS AppConfig	3	3

Variabel lingkungan	Detail	Nilai default	Nilai sampel
POLL_INTERVAL	Variabel lingkungan ini mengontrol seberapa sering agen melakukan polling AWS AppConfig untuk data konfigurasi yang diperbarui. 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	45 45-an 5m 1 jam

Variabel lingkungan	Detail	Nilai default	Nilai sampel
PREFETCH_LIST	Variabel lingkungan ini menentukan data konfigurasi permintaan agen AWS AppConfig segera setelah dimulai. Beberapa pengidentifikasi konfigurasi dapat disediakan dalam daftar yang dipisahkan koma.	Tidak ada	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Config1,MyEnv ::Config2 MyApp MyEnv

Variabel lingkungan	Detail	Nilai default	Nilai sampel
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	true SALAH
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.	Tidak ada	header: nilai h1: v1, h2: v2

Variabel lingkungan	Detail	Nilai default	Nilai sampel
PROXY_URL	Variabel lingkungan ini menentukan URL proxy yang akan digunakan untuk koneksi dari agen ke Layanan AWS, termasuk AWS AppConfig. HTTPSdan HTTP URLs didukung.	Tidak ada	http://localhost:7474 https://my-proxy.example.com

Variabel lingkungan	Detail	Nilai default	Nilai sampel
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	3000ms	3000 3000ms 5s

Variabel lingkungan	Detail	Nilai default	Nilai sampel
	unit tidak ditentukan, agen default ke milidetik. Sebagai contoh, 5000, 5000 ms dan 5s menghasilkan nilai batas waktu permintaan yang sama.		
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	arn:aws:iam: :123456789012:peran/MyRole
ROLE_EXTERNAL_ID	Variabel lingkungan ini menentukan ID eksternal untuk digunakan dengan peran ARN yang diasumsikan.	Tidak ada	MyExternalId
ROLE_SESSION_NAME	Variabel lingkungan ini menentukan nama sesi yang akan dikaitkan dengan kredensyal untuk peran IAM yang diasumsikan.	Tidak ada	AWSAppConfigAgentSession

Variabel lingkungan	Detail	Nilai default	Nilai sampel
`SERVICE_REGION`	Variabel lingkungan ini menentukan alternatif AWS Region 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	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Variabel lingkungan ini mengonfigurasi AWS AppConfig Agen untuk menunggu hingga manifes diproses sebelum menyelesaikan startup.	true	true SALAH

Mengambil data konfigurasi untuk aplikasi yang berjalan di Amazon ECS dan Amazon EKS

Anda dapat mengambil data konfigurasi dari AWS AppConfig Agen untuk aplikasi yang berjalan di Amazon ECS dan Amazon EKS 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"
```

Panggilan mengembalikan metadata konfigurasi di header HTTP, termasuk versi konfigurasi, jenis konten, dan label versi konfigurasi (jika ada). Tubuh respon agen berisi konten konfigurasi. Inilah contohnya:

```
HTTP/1.1 200 OK
Configuration-Version: 1
Content-Type: application/json
Date: Tue, 18 Feb 2025 20:20:16 GMT
Content-Length: 31

My test config
```

Mengambil flag fitur dasar dan multi-varian

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

> **Note**
>
> Kemampuan untuk memanggil flag fitur tunggal atau subset flag dalam konfigurasi hanya tersedia di AWS AppConfig Agent 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_KEY`* kueri untuk profil AWS AppConfig konfigurasi.

Untuk mengambil bendera tunggal dan atributnya

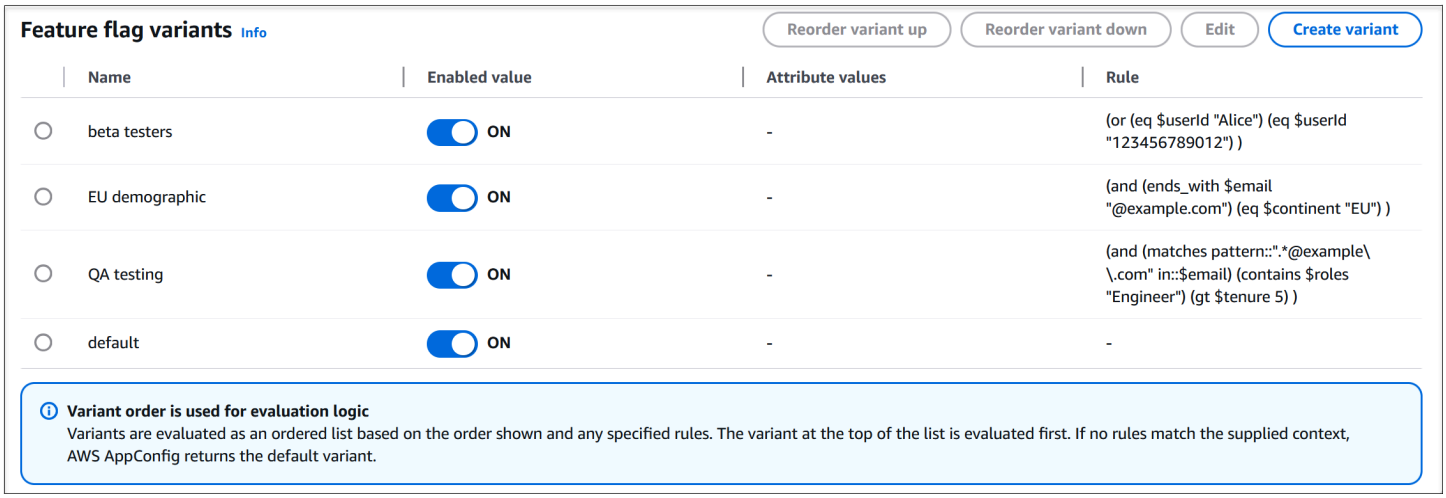
```
curl "http://localhost:2772/applications/APPLICATION_NAME/
environments/ENVIRONMENT_NAME/configurations/CONFIGURATION_NAME?flag=FLAG_KEY"
```

Untuk mengambil beberapa bendera dan atributnya

```
curl "http://localhost:2772/applications/APPLICATION_NAME/
environments/ENVIRONMENT_NAME/configurations/CONFIGURATION_NAME?
flag=FLAG_KEY_ONE&flag=FLAG_KEY_TWO"
```

Untuk mengambil varian bendera fitur berdasarkan konteks pemanggil

Contoh cURL berikut menunjukkan cara mengambil varian flag fitur berdasarkan konteks pemanggil. Untuk mengilustrasikan cara melakukan panggilan ini dengan sebaik-baiknya, bagian ini menggunakan contoh panggilan berdasarkan skenario di mana pelanggan membuat varian yang mirip dengan berikut ini:



Note

Untuk mengambil varian bendera, Anda harus menggunakan versi terbaru AWS AppConfig Agen di lingkungan komputasi Anda. Untuk informasi selengkapnya, lihat topik berikut yang menjelaskan cara memperbarui, menginstal, atau menambahkan agen untuk setiap lingkungan komputasi berikut:

- Untuk lingkungan komputasi Lambda: Menambahkan ekstensi AWS AppConfig Agen Lambda
- Untuk lingkungan komputasi Amazon EC2: Langkah 2: (Diperlukan) Menginstal dan memulai AWS AppConfig Agen di instans Amazon EC2
- Untuk lingkungan komputasi Amazon ECS: Memulai AWS AppConfig agen untuk integrasi Amazon ECS
- Untuk lingkungan komputasi Amazon EKS: Memulai AWS AppConfig agen untuk integrasi Amazon EKS

Untuk mengambil data bendera menggunakan konteks pemanggil jane_doe@example.org (yang belum memilih program beta):

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
-H "Context: email=jane_doe@example.org" \
-H "Context: opted_in_to_beta=false"
{
  "ui_refresh": {"_variant":"QA","dark_mode_support":true,"enabled":true}
```

```
}
```

Untuk mengambil data bendera menggunakan konteks pemanggil jane_doe@example.org (yang telah memilih program beta):

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
-H "Context: email=jane_doe@example.org" \
-H "Context: opted_in_to_beta=true"
{
  "ui_refresh": {"_variant":"QA","dark_mode_support":true,"enabled":true}
}
```

Untuk mengambil data bendera menggunakan konteks pemanggil jane_doe@qa-testers.example.org (yang merupakan penguji jaminan kualitas di Organisasi Contoh):

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
-H "Context: email=jane_doe@qa-testers.example.org"
{
  "ui_refresh": {"_variant":"QA","dark_mode_support":true,"enabled":true}
}
```

Untuk mengambil data bendera tanpa konteks pemanggil (yang mengembalikan varian Default)

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features
{
"ui_refresh": {"_variant":"Default Variant","enabled":false}
}
```

Untuk mengambil data bendera untuk skenario pemisahan lalu lintas untuk menentukan apakah 1 dari 10 penelepon acak menerima varian 'populasi sampel'

```
for i in {0..9} do ; \
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
-H "Context: email=$i@example.org"
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
```

```
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Sample
 Population","dark_mode_support":false,"enabled":true}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
```

Menggunakan manifes untuk mengaktifkan fitur pengambilan tambahan

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

- Mengkonfigurasi AWS AppConfig Agen untuk mengambil konfigurasi dari beberapa akun: Gunakan AWS AppConfig Agen dari primer atau pengambilan Akun AWS untuk mengambil data konfigurasi dari beberapa akun vendor.

- Mengkonfigurasi AWS AppConfig Agen untuk menulis 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.

Memahami 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:aws: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:/path/to/agent-manifest.json	Gunakan metode ini jika manifes Anda tidak akan sering berubah.
AWS AppConfig konfigurasi	*application-name*:*environment-name*:*configuration-name*	Gunakan metode ini untuk pembaruan dinamis. Anda dapat memperbarui dan menerapkan manifes yang disimpan AWS AppConfig sebagai konfigurasi dengan 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 lingkungan 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

Mengkonfigurasi AWS AppConfig Agen untuk mengambil konfigurasi dari beberapa 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 “kredensial” menggunakan format berikut:

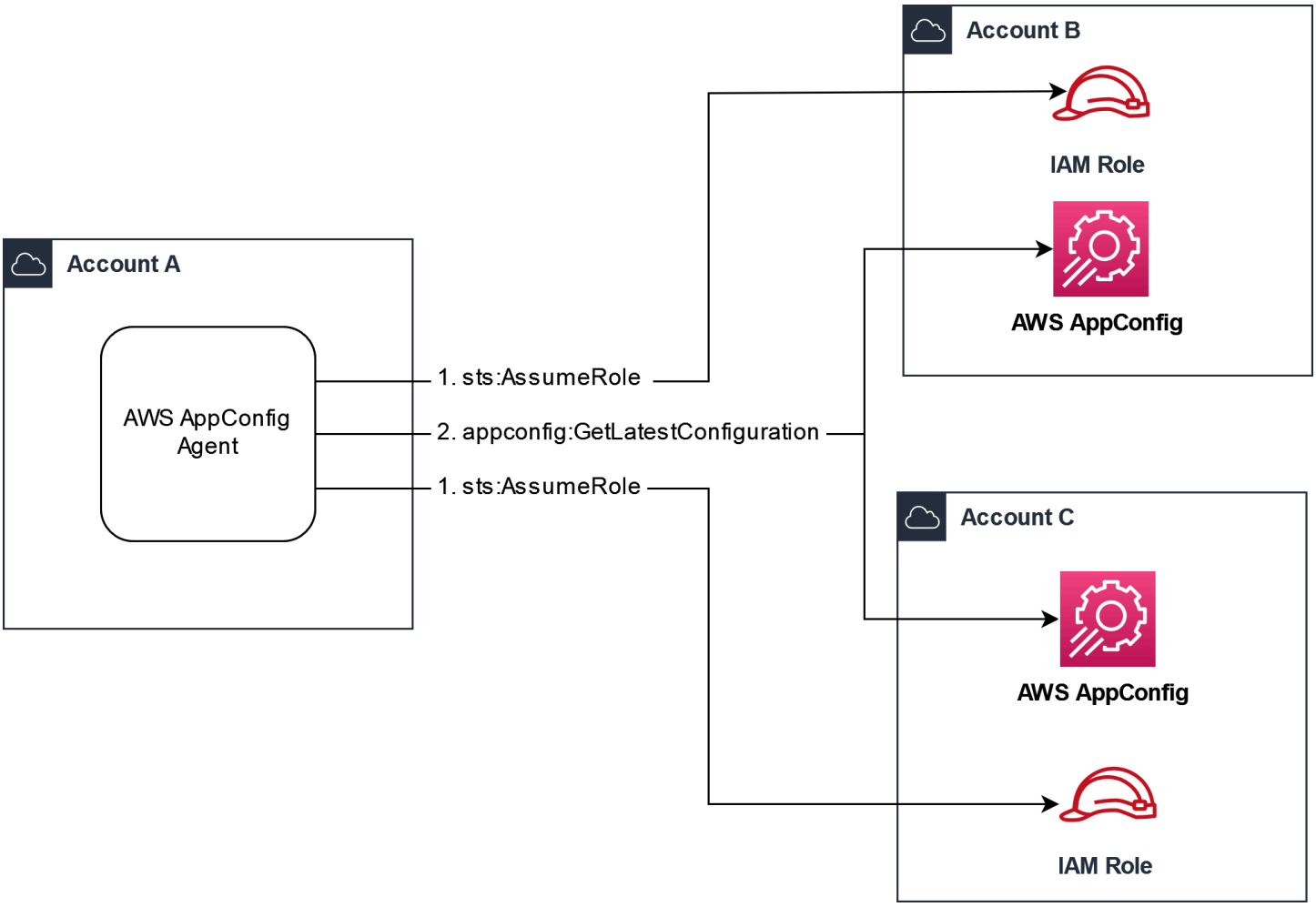
```
{
    "application_name:environment_name:configuration_name": {
        "credentials": {
            "roleArn": "arn:partition:iam::account_ID:role/roleName",
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
        }
    }
}
```

Inilah contohnya:

```
{
    "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
        "credentials": {
            "roleArn": "arn:aws: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 GetLatestConfigurationAPI 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.

- IDs 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 Konsol Manajemen AWS 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 masing-masing *`example resource placeholder`* dengan detail akun vendor.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:StartConfigurationSession",
                "appconfig:GetLatestConfiguration"
            ],
            "Resource": "arn:aws:appconfig:us-
east-1:111122223333:application/vendor_application_ID/
environment/vendor_environment_ID/configuration/vendor_configuration_ID"
        }
    ]
}
```

Inilah contohnya:

JSON

```
{
    "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 Berikutnya.
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 Konsol Manajemen AWS 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 Berikutnya.
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 Berikutnya.
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 masing-masing *example resource placeholder* dengan informasi dari akun pengambilan Anda.

    JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::111122223333: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 bucket tersebut. 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:aws: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
```

Mengkonfigurasi AWS AppConfig Agen untuk menulis 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 '*application*:*environment*:*configuration*' to*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
```

Menghasilkan klien menggunakan spesifikasi 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: AWS AppConfig Agent API
  description: An API model for AWS AppConfig Agent.
servers:
  - url: http://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
        - in: query
          name: flag
          description: The key(s) of the feature flag(s) to retrieve. If not provided,
all flags are returned.
          required: false
          schema:
            type: array
            items:
              type: string
        - in: header
          name: context
          description: Request context used to evaluate multi-variant feature flags.
          required: false
          schema:
            type: array
            items:
              type: string
              pattern: '^\w+=\w+$'
      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
```

Bekerja dengan mode pengembangan lokal AWS AppConfig Agen

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**
>
> - Untuk melihat contoh bendera fitur yang dapat Anda tambahkan ke file di `LOCAL_DEVELOPMENT_DIRECTORY` direktori Anda, lihatContoh bendera fitur untuk mode pengembangan lokal AWS AppConfig Agen.
> - (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.

Contoh bendera fitur untuk mode pengembangan lokal AWS AppConfig Agen

Bagian ini mencakup contoh bendera fitur yang dapat Anda gunakan dengan AWS AppConfig Agen dalam mode pengembangan lokal. Mode pengembangan lokal mengharapkan data tanda fitur dalam format waktu pengambilan data. Format waktu pengambilan adalah format yang dikembalikan saat bendera diambil dari GetLatestConfigurationAPI, yang hanya berisi nilai bendera. Format waktu pengambilan tidak menyertakan definisi lengkap flag (seperti yang diteruskan ke API). CreateHostedConfigurationVersion Definisi lengkap untuk bendera juga berisi informasi seperti nama dan nilai atribut, batasan, dan status diaktifkan bendera.

Topik

- Sampel bendera fitur dasar
- Sampel bendera fitur multi-varian

Sampel bendera fitur dasar

Gunakan contoh bendera fitur dasar berikut dengan AWS AppConfig Agen dalam mode pengembangan lokal.

> **Note**
>
> Jika Anda ingin agen melaporkan jenis konten data bendera fitur lokal Anda sebagai `application/json` (seperti saat mengambil data bendera dari AWS AppConfig lingkungan yang bukan mode pengembangan lokal), file bendera fitur lokal Anda harus menggunakan ekstensi.json. Misalnya, `Local:MyFeatureFlags:SampleB1.json`.

Contoh 1: Bendera tunggal yang mewakili penyegaran UI.

```
{
    "ui_refresh": {
        "enabled": true,
        "new_styleguide_colors": true
    }
}
```

Contoh 2: Beberapa bendera yang mewakili bendera fitur operasional.

```
{
```

```
   "background_worker": {
        "enabled": true,
        "num_threads": 4,
        "queue_name": "MyWorkQueue"
   },
   "emergency_shutoff_switch": {
        "enabled": false
   },
   "logger_settings": {
        "enabled": true,
        "level": "INFO"
   }
}
```

Sampel bendera fitur multi-varian

Format waktu pengambilan konfigurasi flag fitur yang berisi setidaknya satu flag fitur multi-varian direpresentasikan sebagai data Amazon Ion, bukan data JSON. Dalam format ini, flag multi-varian direpresentasikan sebagai daftar beranotasi, dan flag dasar direpresentasikan sebagai string beranotasi. Elemen daftar dari bendera multi-varian adalah tuple (daftar dengan panjang dua), yang mewakili varian tunggal, atau string, yang mewakili varian default. Dalam tupel varian, elemen pertama adalah ekspresi s yang mewakili aturan varian, dan elemen kedua adalah string yang mewakili konten varian.

Agar agen dapat menafsirkan file-file ini dengan benar, file bendera fitur lokal Anda harus menggunakan ekstensi berikut:. `application%ion%type=AWS.AppConfig.FeatureFlags`. Misalnya, `Local:MyFeatureFlags:SampleMV1.application%ion %type=AWS.AppConfig.FeatureFlags`.

Contoh 1: Bendera multi-varian yang mewakili rilis berjenjang dari fitur baru.

```
'tiered_release'::[
  [
    (or (and (eq $group "Tier1") (split by::$userId pct::1 seed::"2025.01.01")) (and
 (eq $group "Tier2") (split by::$userId pct::7 seed::"2025.01.01"))),
    '''{"_variant": "ShowFeature", "enabled": true}'''
  ],
  '''{"_variant": "HideFeature", "enabled": false}'''
]
```

Contoh 2: Beberapa bendera yang mewakili tampilan UX yang berbeda berdasarkan ID pengguna. Dua bendera pertama adalah multi-varian dan bendera terakhir adalah dasar.

```
'colorway'::[
  [
    (contains $userId "beta"),
    '''{"_variant": "BetaTesters", "enabled": true, "background": "blue", "foreground":
 "red"}''',
  ],
  [
    (split by::$userId pct::10),
    '''{"_variant": "SplitRollOutRedAndBlue", "enabled": true, "background": "blue",
 "foreground": "red"}''',
  ],
  '''{"_variant": "default", "enabled": true, "background": "green", "foreground":
 "green"}''',
]

'simple_feature'::[
  [
    (contains $userId "beta"),
    '''{"_variant": "BetaTesters", "enabled": true}'''
  ],
  '''{"_variant": "default", "enabled": false}'''
]

'button_color'::'''{"enabled": true, "color": "orange"}'''
```

AWS AppConfig pertimbangan penggunaan browser dan seluler

Bendera fitur memungkinkan Anda memperbarui pengalaman halaman web dan aplikasi seluler Anda dengan cepat, tanpa biaya overhead, risiko, atau kekakuan rilis toko aplikasi. Dengan menggunakan flag fitur, Anda dapat secara bertahap merilis perubahan ke basis pengguna pada saat yang Anda pilih. Jika Anda mengalami kesalahan, Anda dapat langsung memutar kembali perubahan tanpa mengharuskan pengguna untuk meningkatkan ke versi perangkat lunak baru. Singkatnya, flag fitur memberikan kontrol dan fleksibilitas yang lebih besar saat menerapkan perubahan pada aplikasi Anda.

Bagian berikut menjelaskan pertimbangan penting untuk menggunakan bendera AWS AppConfig fitur dengan halaman web dan perangkat seluler.

Topik

- Data konfigurasi dan pengambilan bendera
- Otentikasi dan Amazon Cognito
- Pembuatan cache
- Segmentasi
- Bandwidth (kasus penggunaan seluler)
- Kasus penggunaan bendera tambahan

Data konfigurasi dan pengambilan bendera

Untuk kasus penggunaan browser dan seluler, banyak pelanggan memilih untuk menggunakan lapisan proxy antara web atau aplikasi seluler dan AWS AppConfig. Melakukannya memisahkan volume AWS AppConfig panggilan Anda dari ukuran basis pengguna Anda, yang mengurangi biaya. Ini juga memungkinkan Anda memanfaatkan AWS AppConfig Agen, yang mengoptimalkan kinerja pengambilan bendera dan mendukung fitur seperti flag multi-varian. AWS AppConfig merekomendasikan penggunaan AWS Lambda untuk membuat proxy. Alih-alih mengambil flag langsung dari AWS AppConfig, konfigurasikan ekstensi AWS AppConfig Lambda untuk mengambil flag fitur Anda dalam fungsi Lambda. Tulis fungsi untuk menerima parameter AWS AppConfig pengambilan dari permintaan acara dan mengembalikan data konfigurasi yang sesuai dalam respons Lambda. Paparkan proxy Anda ke internet menggunakan fungsi URLsLambda.

Setelah Anda mengkonfigurasi proxy Anda, pertimbangkan frekuensi di mana Anda mengambil data. Kasus penggunaan seluler biasanya tidak memerlukan interval polling frekuensi tinggi. Konfigurasikan AWS AppConfig Agen untuk menyegarkan data AWS AppConfig lebih sering daripada aplikasi Anda yang di-refresh dari proxy.

Otentikasi dan Amazon Cognito

Fungsi Lambda URLs mendukung dua bentuk kontrol akses, `AWS_IAM` dan. `NONE` Gunakan `NONE` jika Anda lebih suka menerapkan otentikasi dan otorisasi Anda sendiri dalam fungsi Lambda Anda. `NONE`juga merupakan opsi yang disarankan jika kasus penggunaan Anda memungkinkan mengekspos titik akhir Anda ke publik dan data konfigurasi Anda tidak berisi data sensitif. Untuk semua kasus penggunaan lainnya, gunakan`AWS_IAM`.

> **Important**
>
> Jika Anda mengekspos titik akhir Anda ke internet tanpa autentikasi, pastikan bahwa data konfigurasi Anda tidak membocorkan data sensitif, termasuk informasi identitas pribadi (PII), pengguna, atau nama fitur yang belum dirilis. IDs

Jika Anda memilih untuk menggunakan`AWS_IAM`, Anda harus mengelola kredensyal dengan Amazon Cognito. Untuk memulai Amazon Cognito, Anda membuat kumpulan identitas. Kumpulan identitas memungkinkan Anda untuk menjual kredensi jangka pendek ke aplikasi Anda untuk pengguna yang diautentikasi atau tamu. Anda perlu menambahkan peran dalam kumpulan identitas yang memungkinkan pengguna untuk menggunakan fungsi `InvokeFunctionUrl` untuk Lambda Anda. Melakukannya memungkinkan instance aplikasi Anda untuk mengakses kredensyal yang diperlukan untuk mengambil data konfigurasi Anda.

Saat bekerja dengan Amazon Cognito di aplikasi Anda, pertimbangkan untuk menggunakan. AWS Amplify Amplify menyederhanakan interaksi mobile/web aplikasi dengan AWS dan menyediakan dukungan bawaan untuk Amazon Cognito.

Pembuatan cache

Saat menggunakan AWS AppConfig, Anda harus selalu menyimpan data konfigurasi Anda secara lokal di perangkat atau di browser. Caching menawarkan keuntungan sebagai berikut:

- Meningkatkan kinerja dengan mengurangi latensi dan pengurasan baterai
- Menawarkan stabilitas dengan menghilangkan dependensi ke akses jaringan
- Menurunkan biaya dengan mengurangi frekuensi pengambilan data

Untuk kasus penggunaan seluler, kami sarankan Anda menerapkan cache dalam memori dan persisten di perangkat. Konfigurasikan aplikasi Anda untuk mencoba mengambil konfigurasi yang diinginkan dari cache dalam memori dan kembali mengambil dari proxy Anda, jika perlu. Setelah berhasil diambil dari proxy Anda, perbarui cache dalam memori dan kemudian pertahankan konfigurasi ke perangkat. Gunakan proses latar belakang untuk mengulangi cache dan menyegarkan setiap konfigurasi. Saat mengambil konfigurasi untuk pertama kalinya setelah startup aplikasi, jika pengambilan tidak berhasil, tunda ke konfigurasi persisten (dan gunakan untuk menyemai cache dalam memori).

Segmentasi

Saat menggunakan flag fitur, Anda mungkin ingin mengelompokkan pengalaman penandaan fitur di seluruh basis pelanggan Anda. Untuk melakukannya, berikan konteks ke panggilan pengambilan flag Anda, dan konfigurasikan aturan untuk menampilkan varian flag fitur yang berbeda berdasarkan konteks yang disediakan. Misalnya, Anda mungkin memiliki varian flag fitur untuk pengguna iOS 18.X, varian untuk pengguna iOS 17.X, dan tanda default untuk semua versi iOS lainnya. Dengan varian, Anda dapat mengonfigurasi setiap versi iOS aplikasi Anda untuk menargetkan konfigurasi yang sama di lingkungan yang sama, tetapi berdasarkan konteks yang disediakan dalam panggilan pengambilan (misalnya, “versi”: “i OS18 .1"), perangkat akan menerima varian konfigurasi yang sesuai.

> **Note**
>
> Jika Anda menggunakan varian tanda AWS AppConfig fitur untuk kasus penggunaan seluler, Anda harus menggunakan AWS AppConfig Agen dan proxy untuk mengambil flag fitur.

Jika Anda memilih untuk tidak menggunakan AWS AppConfig Agen untuk mengambil flag fitur, Anda dapat memanfaatkan AWS AppConfig lingkungan untuk segmentasi kardinalitas rendah yang sederhana. Lingkungan adalah grup penyebaran logis untuk target Anda. Selain mempartisi konfigurasi Anda ke lingkungan pengembangan, pengujian, dan produksi, Anda dapat membagi basis pelanggan Anda dengan membuat lingkungan khusus seluler, seperti jenis perangkat (tablet vs ponsel) atau versi utama OS. Dengan lingkungan terpisah, Anda dapat menyebarkan kumpulan data konfigurasi yang sama atau berbeda untuk memenuhi persyaratan khusus basis pelanggan Anda.

Bandwidth (kasus penggunaan seluler)

Secara umum, bertujuan untuk menjaga ukuran setiap bendera yang ditetapkan kecil. Kasus penggunaan seluler cenderung melibatkan kendala bandwidth rendah. Meminimalkan ukuran data Anda akan membantu Anda mempertahankan pengalaman yang konsisten di seluruh basis pengguna Anda. Juga, pertimbangkan bahwa karena perangkat seluler sering beroperasi antara lingkungan rendah dan tanpa bandwidth, caching di perangkat sangat penting. Kode aplikasi yang gagal dengan baik jika tidak ada data konfigurasi yang dapat diambil juga penting.

Kasus penggunaan bendera tambahan

Kekuatan bendera fitur melampaui kenyamanan rilis fitur. Bendera operasional yang berdiri lama dapat digunakan untuk meningkatkan postur operasional aplikasi Anda. Misalnya, Anda dapat membuat sakelar pemantauan kinerja yang memancarkan metrik tambahan dan data debug selama acara berlangsung. Atau, Anda mungkin ingin mempertahankan dan menyesuaikan kecepatan refresh aplikasi Anda untuk segmen basis pelanggan Anda.

Mengambil data konfigurasi tanpa Agen AWS AppConfig

Cara yang disarankan untuk mengambil data konfigurasi AWS AppConfig adalah dengan menggunakan Agen yang dikembangkan dan dikelola Amazon. AWS AppConfig Dengan agen, Anda dapat menyimpan data konfigurasi secara lokal dan secara asinkron melakukan polling layanan pesawat AWS AppConfig data untuk pembaruan. caching/polling Proses ini memastikan bahwa data konfigurasi Anda selalu tersedia untuk aplikasi Anda sambil meminimalkan latensi dan biaya. Jika Anda memilih untuk tidak menggunakan agen, Anda dapat menghubungi publik APIs langsung dari layanan pesawat AWS AppConfig data.

Layanan data plane menggunakan dua tindakan API, StartConfigurationSessiondan GetLatestConfiguration. Layanan pesawat data juga menggunakan titik akhir terpisah dari bidang AWS AppConfig kontrol.

> **Note**
>
> Layanan bidang data menggantikan proses pengambilan data konfigurasi sebelumnya dengan menggunakan tindakan `GetConfiguration` API. `GetConfiguration`API tidak digunakan lagi.

Cara kerjanya

Begini cara kerja proses panggilan langsung AWS AppConfig APIs menggunakan layanan pesawat data.

Aplikasi Anda mengambil data konfigurasi dengan terlebih dahulu membuat sesi konfigurasi menggunakan operasi StartConfigurationSessionAPI. Klien sesi Anda kemudian melakukan panggilan berkala GetLatestConfigurationuntuk 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 keGetLatestConfiguration.

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 AndaGetLatestConfiguration. Anda harus menggunakan token baru di GetLatestConfiguration response (NextPollConfigurationToken) di setiap panggilan berikutnyaGetLatestConfiguration. Untuk mendukung kasus penggunaan polling yang panjang, token berlaku hingga 24 jam. Jika GetLatestConfiguration panggilan menggunakan token kedaluwarsa, sistem akan kembaliBadRequestException.

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.
- NextPollIntervalInSeconds: durasi yang harus ditunggu klien sebelum melakukan panggilan berikutnyaGetLatestConfiguration.
- Konfigurasi: data terbaru yang ditujukan untuk sesi tersebut. Ini mungkin kosong jika klien sudah memiliki versi konfigurasi terbaru.

> **Important**
>
> Perhatikan informasi penting berikut.
>
> - StartConfigurationSessionAPI 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.

(Contoh) Mengambil konfigurasi dengan memanggil AWS AppConfig APIs

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 IDs (atau nama) AWS AppConfig aplikasi, lingkungan, dan profil konfigurasi. API mengembalikan sebuah 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 InitialConfigurationTokenuntuk memanggil GetLatestConfigurationuntuk 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 AWS AppConfig 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

- Memahami AWS AppConfig ekstensi
- Bekerja dengan ekstensi yang AWS ditulis
- Walkthrough: Membuat ekstensi khusus AWS AppConfig

Memahami 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 1: Tentukan apa yang ingin Anda lakukan dengan ekstensi
- Langkah 2: Tentukan kapan Anda ingin ekstensi berjalan

- 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
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.

Ekstensi	Deskripsi
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:

Titik tindakan PRE_*: 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.

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`

ON_* action points: Ekstensi juga dapat berjalan secara paralel dengan AWS AppConfig alur kerja dengan menggunakan titik tindakan`ON_*`. `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.

- `ON_DEPLOYMENT_START`
- `ON_DEPLOYMENT_STEP`

- `ON_DEPLOYMENT_BAKING`
- `ON_DEPLOYMENT_COMPLETE`
- `ON_DEPLOYMENT_ROLLED_BACK`

Titik aksi `AT_*`: Tindakan ekstensi yang dikonfigurasi pada titik `AT_*` tindakan dipanggil secara sinkron dan paralel dengan alur kerja. AWS AppConfig Jika ekstensi mengalami kesalahan selama titik `AT_*` tindakan, layanan menghentikan alur kerja dan memutar kembali penerapan.

- `AT_DEPLOYMENT_TICK`

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 Konsol Manajemen AWS atau dengan memanggil tindakan API ekstensi langsung dari AWS CLI, AWS Tools for PowerShell, atau SDK.

Ekstensi	Deskripsi
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 setiap kali Anda membuat perubahan pada bendera fitur.

Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon EventBridge

`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 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"
   }
}
```

Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon SNS

`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"
        }
    }
}
```

Menggunakan peristiwa AWS AppConfig penerapan ke ekstensi Amazon SQS

`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"
}
```

Menggunakan 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 AWS Region 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 kredensil 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 menghapus 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_IDapplication_ID*, dan *configuration_profile_ID* dengan informasi dari lingkungan bendera AWS AppConfig fitur Anda.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:CreateExtensionAssociation",
                "appconfig:ListExtensionAssociations",
                "appconfig:GetConfigurationProfile"
            ],
            "Resource": [
                "arn:aws:appconfig:us-
east-1:111122223333:application/application_ID",
                "arn:aws:appconfig:us-
east-1:111122223333:application/application_ID/
configurationprofile/configuration_profile_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:ListApplications"
            ],
            "Resource": [
                "arn:aws:appconfig:us-east-1:111122223333:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "appconfig:ListConfigurationProfiles"
            ],
            "Resource": [
                "arn:aws:appconfig:us-
east-1:111122223333: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 Berikutnya: 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: Tinjauan.
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 AWS Region. 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 instance 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 AWS Region, 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 AWS Region.

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, lihatLangkah 3: Buat 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

- Langkah 1: Buat fungsi Lambda untuk ekstensi khusus AWS AppConfig
- Langkah 2: Konfigurasikan izin untuk ekstensi khusus AWS AppConfig
- Langkah 3: Buat AWS AppConfig ekstensi khusus
- Langkah 4: Buat asosiasi ekstensi untuk AWS AppConfig ekstensi khusus

Langkah 1: Buat 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 penelusuran 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

AtDeploymentTick

```
{
    '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',
    'DeploymentState': 'DEPLOYING',
    'PercentageComplete': '0.0'
}
```

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 ditampilkan oleh fungsi Lambda Anda sebagai respons terhadap permintaan dari ekstensi kustom AWS AppConfig .

PRE_* Peristiwa sinkron - respons yang berhasil

Jika Anda ingin mengubah konten, gunakan yang berikut ini:

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

AT_* Acara sinkron - respons yang berhasil

Jika Anda ingin mengontrol langkah penerapan berikutnya (lanjutkan penerapan atau putar kembali) set `Directive` dan `Description` atribut dalam respons.

```
"Directive": "ROLL_BACK"
"Description" "Deployment event log description"
```

`Directive`mendukung dua nilai: `CONTINUE` atau`ROLL_BACK`. Gunakan enum ini dalam respons payload Anda untuk mengontrol langkah penerapan selanjutnya.

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"
        }]
    }
```

Langkah 2: Konfigurasikan izin untuk ekstensi khusus 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 mengasumsikannya

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 khusus.

   JSON

```
{
  "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 *`Your Lambda ARN`* dengan Nama Sumber Daya Amazon (ARN) Lambda Anda.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:function-
name"
    }
  ]
}
```

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 kustom Anda, pilih ikon Refresh dan kemudian 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.

Langkah 3: Buat 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:

Titik tindakan PRE_*: 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.

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`

ON_* action points: Ekstensi juga dapat berjalan secara paralel dengan AWS AppConfig alur kerja dengan menggunakan titik tindakan`ON_*`. `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.

- `ON_DEPLOYMENT_START`
- `ON_DEPLOYMENT_STEP`
- `ON_DEPLOYMENT_BAKING`
- `ON_DEPLOYMENT_COMPLETE`
- `ON_DEPLOYMENT_ROLLED_BACK`

Titik aksi AT_*: Tindakan ekstensi yang dikonfigurasi pada titik `AT_*` tindakan dipanggil secara sinkron dan paralel dengan alur kerja. AWS AppConfig Jika ekstensi mengalami kesalahan selama titik `AT_*` tindakan, layanan menghentikan alur kerja dan memutar kembali penerapan.

- `AT_DEPLOYMENT_TICK`

Titik `AT_DEPLOYMENT_TICK` tindakan mendukung integrasi pemantauan pihak ketiga. `AT_DEPLOYMENT_TICK`dipanggil selama orkestrasi pemrosesan penerapan konfigurasi. Jika Anda menggunakan solusi pemantauan pihak ketiga (misalnya, Datadog atau New Relic), Anda dapat membuat AWS AppConfig ekstensi yang memeriksa alarm di titik `AT_DEPLOYMENT_TICK` tindakan dan, sebagai pagar pembatas keselamatan, memutar kembali penerapan jika memicu alarm.

Jika Anda menggunakan solusi pemantauan pihak ketiga seperti Datadog atau New Relic, Anda dapat membuat AWS AppConfig ekstensi yang memeriksa alarm di titik `AT_DEPLOYMENT_TICK` tindakan dan, sebagai pagar pembatas keselamatan, memutar kembali penerapan jika memicu alarm. Untuk informasi selengkapnya, lihat contoh integrasi Datadog dan New Relic berikut tentang: GitHub

- Datadog
- New Relic

Untuk informasi selengkapnya tentang AWS AppConfig ekstensi, lihat topik berikut:

- Memperluas AWS AppConfig alur kerja menggunakan ekstensi
- Walkthrough: Membuat ekstensi khusus AWS AppConfig

Ekstensi sampel

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 AWS Region 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`

Langkah 4: Buat asosiasi ekstensi untuk AWS AppConfig ekstensi khusus

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. Sebagai contoh: `arn:aws:lambda:`*`aws-region`*`:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Pilih Buat asosiasi ke sumber daya.

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.

Menggunakan contoh kode untuk melakukan AWS AppConfig tugas-tugas umum

Bagian ini mencakup contoh kode untuk melakukan tindakan umum AWS AppConfig secara terprogram. Kami menyarankan Anda menggunakan sampel ini dengan Java, Python, dan JavaScript SDKs 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
- Membuat profil konfigurasi untuk rahasia yang disimpan di Secrets Manager
- Menerapkan profil konfigurasi
- Menggunakan AWS AppConfig Agen untuk membaca profil konfigurasi bentuk bebas
- Menggunakan AWS AppConfig Agen untuk membaca bendera fitur tertentu
- Menggunakan AWS AppConfig Agen untuk mengambil bendera fitur dengan varian
- Menggunakan tindakan GetLatestConfiguration API untuk membaca profil konfigurasi bentuk bebas
- Membersihkan lingkungan Anda

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 menyebut sebagai berikut APIs:

- 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 menyebut sebagai berikut APIs:

- 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 menyebut sebagai berikut APIs:

- 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 How to use AWS AppConfig Agent
        */

        // 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
# How to use AWS AppConfig Agent
#

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
// How to use AWS AppConfig Agent

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 AWS AppConfig Agen untuk mengambil bendera fitur dengan varian

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode.

Java

```
public static void retrieveConfigFromAgentWithVariants() throws Exception {
    /*
    This sample retrieves feature flag configuration data
    containing variants from AWS AppConfig Agent.

    For more information about the agent, see How to use AWS AppConfig Agent
    */

    // 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();

    // Provide context in the 'Context' header
```

```
    // In the header value, use '=' to separate context key from context value
    // Note: Multiple context values may be passed either across
    // multiple headers or as comma-separated values in a single header
    con.setRequestProperty("Context", "country=US");

    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

```
# This sample retrieve features flag configuration data
# containing variants from AWS AppConfig Agent.

# For more information about the agent, see How to use AWS AppConfig Agent

import requests

application_name = 'MyDemoApp'
environment_name = 'Beta'
configuration_profile_name = 'MyConfigProfile'

# 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/{configuration_profile_name}",
                        headers = {
                            "Context": "country=US" # Provide context in the
 'Context' header
                                                    # In the header value, use '='
 to separate context key from context value
                                                    # Note: Multiple context values
 may be passed either across
                                                    # multiple headers or as comma-
separated values in a single header
```

```
                        }
)
print("Configuration from agent via HTTP: ", response.json())
```

JavaScript

```
// This sample retrieves feature flag configuration data
// containing variants from AWS AppConfig Agent.

// For more information about the agent, see How to use AWS AppConfig Agent

const application_name = "MyDemoApp";
const environment_name = "Beta";
const configuration_profile_name = "MyConfigProfile";

const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${configuration_profile_name}`;

// make a GET request to the agent's local server to retrieve the configuration data
const response = await fetch(url, {
    method: 'GET',
    headers: {
        'Context': 'country=US' // Provide context in the 'Context' header
                                // In the header value, use '=' to separate context
 key from context value
                                // Note: Multiple context values may be passed
 either across
                                // multiple headers or as comma-separated values in
 a single header
    }
});

const config = await response.json();
console.log("Configuration from agent via HTTP: ", config);
```

Menggunakan tindakan GetLatestConfiguration API untuk membaca profil konfigurasi bentuk bebas

Masing-masing sampel berikut mencakup komentar tentang tindakan yang dilakukan oleh kode. Sampel di bagian ini menyebut sebagai berikut APIs:

- GetLatestConfiguration
- StartConfigurationSession

Java

```
/*
The example below uses two AWS AppConfig Data APIs: StartConfigurationSession and
 GetLatestConfiguration.
For more information about these APIs, see AWS AppConfig Data.

This class is meant to be used as a singleton to retrieve the latest configuration
 data from AWS AppConfig.
This class maintains a cache of the latest configuration data in addition to the
 configuration token to be
passed to the next GetLatestConfiguration API call.
*/
public class AppConfigApiRetriever {

    /*
     * Set of AppConfig invalid parameter problems that require restarting the
 configuration session.
     * If the GetLatestConfiguration API call fails with any of these problems (e.g.
 token is EXPIRED or CORRUPTED),
     * we need to call StartConfigurationSession again to obtain a new configuration
 token before retrying.
     */
    private final Set<InvalidParameterProblem> SESSION_RESTART_REQUIRED =
            Stream.of(InvalidParameterProblem.EXPIRED,
 InvalidParameterProblem.CORRUPTED)
                    .collect(Collectors.toSet());

    /* AWS AppConfig Data SDK client used to interact with the AWS AppConfig Data
 service.
     */
    private final AppConfigDataClient appConfigData;

    /*
    The configuration token to be passed to the next GetLatestConfiguration API
 call.
     */
    private String configurationToken;
```

```
   /*
   The cached configuration data to be returned when there is no new configuration
data available.
    */
   private SdkBytes configuration;

   public AppConfigApiRetriever() {
       this.appConfigData = AppConfigDataClient.create();
   }

   /*
   Returns the latest configuration data stored in AWS AppConfig.
    */
   public SdkBytes getConfig() {
       /*
       If there is no configuration token yet, get one by starting a new session
with the StartConfigurationSession API.
       Note that this API does not return configuration data. Rather, it returns an
initial configuration token that is
       subsequently passed to the GetLatestConfiguration API.
        */
       if (this.configurationToken == null) {
           startNewSession();
       }

       GetLatestConfigurationResponse response = null;

       try {
           /*
           Retrieve the configuration from the GetLatestConfiguration API,
providing the current configuration token.
           If this caller does not yet have the latest configuration (e.g. this is
the first call to GetLatestConfiguration
           or new configuration data has been deployed since the first call), the
latest configuration data will be returned.
           Otherwise, the GetLatestConfiguration API will not return any data since
the caller already has the latest.
            */
           response = appConfigData.getLatestConfiguration(
                   GetLatestConfigurationRequest.builder()

.configurationToken(this.configurationToken)
                                                   .build());
```

```
        } catch (ResourceNotFoundException e) {
            // Handle resource not found by refreshing the session
            System.err.println("Resource not found — refreshing session and
 retrying...");
            startNewSession();
            response = appConfigData.getLatestConfiguration(
                    GetLatestConfigurationRequest.builder()

 .configurationToken(this.configurationToken)
                                                    .build());
        } catch (BadRequestException e) {
            // Handle expired or corrupted token by refreshing the session
            boolean needsNewSession = Optional.ofNullable(e.details())
                                                .map(details ->
 details.invalidParameters()
                                                                        .values()
                                                                        .stream()
                                                                        .anyMatch(val -
> SESSION_RESTART_REQUIRED.contains(val.problem())))
                                                .orElse(false);
            if (needsNewSession) {
                System.err.println("Configuration token expired or corrupted —
 refreshing session and retrying...");
                startNewSession();
                response = appConfigData.getLatestConfiguration(
                        GetLatestConfigurationRequest.builder()

 .configurationToken(this.configurationToken)
                                                        .build());
            } else {
                throw e; // rethrow if it's another kind of bad request
            }
        }

        if (response == null) {
            // Should not happen, but return cached config if no response
            return this.configuration;
        }

        /*
        Save the returned configuration token so that it can be passed to the next
 GetLatestConfiguration API call.
        Warning: Not persisting this token for use in the next
 GetLatestConfiguration API call may result in higher
```

```
        than expected usage costs.
         */
        this.configurationToken = response.nextPollConfigurationToken();

        /*
        If the GetLatestConfiguration API returned configuration data, update the
 cached configuration with the returned data.
        Otherwise, assume the configuration has not changed, and return the cached
 configuration.
         */
        SdkBytes configFromApi = response.configuration();
        if (configFromApi != null && configFromApi.asByteArray().length != 0) {
            this.configuration = configFromApi;
            System.out.println("Configuration contents have changed since the last
 GetLatestConfiguration call, new contents = "
                    + this.configuration.asUtf8String());
        } else {
            System.out.println("GetLatestConfiguration returned an empty response
 because we already have the latest configuration");
        }

        return this.configuration;
    }

    /*
    Starts a new session with AppConfig and retrieves an initial configuration
 token.
     */
    private void startNewSession() {
        StartConfigurationSessionResponse session =
 appConfigData.startConfigurationSession(req -> req
                .applicationIdentifier("MyDemoApp")
                .configurationProfileIdentifier("MyConfig")
                .environmentIdentifier("Beta"));
        this.configurationToken = session.initialConfigurationToken();
    }
}
```

Python

```
# The example below uses two AWS AppConfig Data APIs: StartConfigurationSession and
 GetLatestConfiguration.
# For more information about these APIs, see AWS AppConfig Data.
```

```
#
# This class is meant to be used as a singleton to retrieve the latest configuration
 data from AWS AppConfig.
# This class maintains a cache of the latest configuration data in addition to the
 configuration token to be
# passed to the next GetLatestConfiguration API call.
class AppConfigApiRetriever:
    # Set of AppConfig invalid parameter problems that require restarting the
 configuration session.
    # If the GetLatestConfiguration API call fails with any of these problems (e.g.
 token is EXPIRED or CORRUPTED),
    # we need to call StartConfigurationSession again to obtain a new configuration
 token before retrying.
    SESSION_RESTART_REQUIRED = {"EXPIRED", "CORRUPTED"}

    def __init__(self):
        # AWS AppConfig Data SDK client used to interact with the AWS AppConfig Data
 service.
        self.appconfigdata = boto3.client('appconfigdata')

        # The configuration token to be passed to the next GetLatestConfiguration
 API call.
        self.configuration_token = None

        # The cached configuration data to be returned when there is no new
 configuration data available.
        self.configuration = None

    # Returns the latest configuration data stored in AWS AppConfig.
    def get_config(self):
        # If there is no configuration token yet, get one by starting a new session
 with the StartConfigurationSession API.
        # Note that this API does not return configuration data. Rather, it returns
 an initial configuration token that is
        # subsequently passed to the GetLatestConfiguration API.
        if not self.configuration_token:
            self._start_new_session()

        response = None
        try:
            # Retrieve the configuration from the GetLatestConfiguration API,
 providing the current configuration token.
            # If this caller does not yet have the latest configuration (e.g. this
 is the first call to GetLatestConfiguration
```

```
            # or new configuration data has been deployed since the first call), the
latest configuration data will be returned.
            # Otherwise, the GetLatestConfiguration API will not return any data
since the caller already has the latest.
            response = self.appconfigdata.get_latest_configuration(
                ConfigurationToken=self.configuration_token
            )
        except ClientError as e:
            error_code = e.response.get("Error", {}).get("Code")
            # ResourceNotFoundException — usually means the token/session is invalid
or expired
            if error_code == "ResourceNotFoundException":
                print("Resource not found — refreshing session and retrying...")
                self._start_new_session()
                response = self.appconfigdata.get_latest_configuration(
                    ConfigurationToken=self.configuration_token
                )
            # BadRequestException — check if it's expired or corrupted token
            elif error_code == "BadRequestException":
                details = e.response.get("Error", {}).get("Details", {}) or {}
                invalid_params = details.get("InvalidParameters", {}) or {}
                needs_new_session = any(
                    param.get("Problem") in self.SESSION_RESTART_REQUIRED
                    for param in invalid_params.values()
                )
                if needs_new_session:
                    print("Configuration token expired or corrupted — refreshing
session and retrying...")
                    self._start_new_session()
                    response = self.appconfigdata.get_latest_configuration(
                        ConfigurationToken=self.configuration_token
                    )
                else:
                    raise
            else:
                raise

        if response is None:
            # Should not happen, but return cached config if no response
            return self.configuration

        # Save the returned configuration token so that it can be passed to the next
GetLatestConfiguration API call.
```

```
        # Warning: Not persisting this token for use in the next
 GetLatestConfiguration API call may result in higher
        # than expected usage costs.
        self.configuration_token = response['NextPollConfigurationToken']

        # If the GetLatestConfiguration API returned configuration data, update the
 cached configuration with the returned data.
        # Otherwise, assume the configuration has not changed, and return the cached
 configuration.
        config_stream = response.get('Configuration')
        if config_stream:
            config_from_api = config_stream.read()
            if config_from_api:
                self.configuration = config_from_api
                print(
                    'Configuration contents have changed since the last
 GetLatestConfiguration call, new contents = '
                    + self.configuration.decode('utf-8', errors='ignore')
                )
            else:
                print('GetLatestConfiguration returned an empty response because we
 already have the latest configuration')

        return self.configuration

    # Starts a new session with AppConfig and retrieves an initial configuration
 token.
    def _start_new_session(self):
        session = self.appconfigdata.start_configuration_session(
            ApplicationIdentifier='MyDemoApp',
            ConfigurationProfileIdentifier='MyConfig',
            EnvironmentIdentifier='Beta'
        )
        self.configuration_token = session['InitialConfigurationToken']
```

JavaScript

```
/*
The example below uses two AWS AppConfig Data APIs: StartConfigurationSession and
 GetLatestConfiguration.
For more information about these APIs, see AWS AppConfig Data
.
```

```
This class is meant to be used as a singleton to retrieve the latest configuration
 data from AWS AppConfig.
This class maintains a cache of the latest configuration data in addition to the
 configuration token to be
passed to the next GetLatestConfiguration API call.
*/
class AppConfigApiRetriever {
  constructor() {
    /* AWS AppConfig Data SDK client used to interact with the AWS AppConfig Data
 service.
     */
    this.appconfigdata = new AppConfigDataClient();

    /*
    The configuration token to be passed to the next GetLatestConfiguration API
 call.
     */
    this.configurationToken = null;

    /*
    The cached configuration data to be returned when there is no new configuration
 data available.
     */
    this.configuration = null;
  }

  async startSession() {
    /*
    Starts a new session with the StartConfigurationSession API to get an initial
 configuration token.
    */
    const session = await this.appconfigdata.send(
      new StartConfigurationSessionCommand({
        ApplicationIdentifier: "MyDemoApp",
        ConfigurationProfileIdentifier: "MyConfig",
        EnvironmentIdentifier: "Beta"
      })
    );
    this.configurationToken = session.InitialConfigurationToken;
  }

  /*
  Returns the latest configuration data stored in AWS AppConfig.
   */
```

```
 async getConfig() {
   /*
   If there is no configuration token yet, get one by starting a new session with
the StartConfigurationSession API.
   Note that this API does not return configuration data. Rather, it returns an
initial configuration token that is
   subsequently passed to the GetLatestConfiguration API.
   */
   if (!this.configurationToken) {
     await this.startSession();
   }

   let response;
   try {
     /*
     Retrieve the configuration from the GetLatestConfiguration API, providing the
current configuration token.
     If this caller does not yet have the latest configuration (e.g. this is the
first call to GetLatestConfiguration
     or new configuration data has been deployed since the first call), the latest
configuration data will be returned.
     Otherwise, the GetLatestConfiguration API will not return any data since the
caller already has the latest.
     */
     response = await this.appconfigdata.send(
       new GetLatestConfigurationCommand({
         ConfigurationToken: this.configurationToken
       })
     );
   } catch (err) {
     /*
     Add session restart logic – if the token is invalid or expired, restart the
session and try once more.
     */
     if (err.name === "ResourceNotFoundException" || err.name ===
"BadRequestException") {
       console.warn(
         "Configuration token invalid or expired. Restarting session..."
       );
       await this.startSession();
       response = await this.appconfigdata.send(
         new GetLatestConfigurationCommand({
           ConfigurationToken: this.configurationToken
         })
```

```
        );
      } else {
        throw err;
      }
    }

    /*
    Save the returned configuration token so that it can be passed to the next
 GetLatestConfiguration API call.
    Warning: Not persisting this token for use in the next GetLatestConfiguration
 API call may result in higher
    than expected usage costs.
    */
    this.configurationToken = response.NextPollConfigurationToken;

    /*
    If the GetLatestConfiguration API returned configuration data, update the cached
 configuration with the returned data.
    Otherwise, assume the configuration has not changed, and return the cached
 configuration.
    */
    const configFromApi = response.Configuration
      ? await response.Configuration.transformToString()
      : null;

    if (configFromApi) {
      this.configuration = configFromApi;
      console.log(
        "Configuration contents have changed since the last GetLatestConfiguration
 call, new contents = " +
          this.configuration
      );
    } else {
      console.log(
        "GetLatestConfiguration returned an empty response because we already have
 the latest configuration"
      );
    }

    return this.configuration;
  }
}
```

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 menyebut sebagai berikut APIs:

- 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})`)
  }
}
```

Mengkonfigurasi perlindungan AWS AppConfig penghapusan

AWS AppConfig menyediakan pengaturan akun untuk membantu mencegah pengguna menghapus lingkungan dan profil konfigurasi yang digunakan secara tidak sengaja. AWS AppConfig memantau panggilan ke GetLatestConfigurationdan GetConfigurationdan melacak profil konfigurasi dan lingkungan mana yang telah disertakan dalam panggilan ini dalam interval 60 menit (pengaturan default). Setiap profil konfigurasi atau lingkungan yang diakses dalam interval tersebut akan dianggap aktif. Jika Anda mencoba untuk menghapus profil konfigurasi aktif atau lingkungan, AWS AppConfig mengembalikan kesalahan. Jika perlu, Anda dapat melewati kesalahan ini dengan menggunakan `DeletionProtectionCheck` parameter. Untuk informasi selengkapnya, lihat Melewati atau memaksa pemeriksaan perlindungan penghapusan.

Konfigurasikan perlindungan penghapusan menggunakan konsol

Gunakan prosedur berikut untuk mengonfigurasi perlindungan penghapusan dengan menggunakan konsol. AWS Systems Manager

Untuk mengkonfigurasi perlindungan penghapusan (konsol)

1. Buka AWS Systems Manager konsol di https://console.aws.amazon.com/systems-manager/appconfig/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Gunakan sakelar untuk mengaktifkan atau menonaktifkan perlindungan penghapusan.
4. Untuk periode Perlindungan, tetapkan definisi sumber daya aktif menjadi antara 15 dan 1440 menit.
5. Klik Terapkan.

Konfigurasikan perlindungan penghapusan menggunakan AWS CLI

Gunakan prosedur berikut untuk mengonfigurasi perlindungan penghapusan dengan menggunakan. AWS CLI Ganti *`value`* dalam perintah berikut dengan nilai yang ingin Anda gunakan di lingkungan Anda.

> **Note**
>
> Sebelum Anda mulai, kami sarankan Anda memperbarui ke versi terbaru dari AWS CLI. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui ke versi terbaru dari Panduan AWS Command Line Interface Pengguna. AWS CLI

Untuk mengkonfigurasi perlindungan penghapusan (CLI)

1. Jalankan perintah berikut untuk melihat pengaturan perlindungan penghapusan saat ini.

```
aws appconfig get-account-settings
```

2. Jalankan perintah berikut untuk mengaktifkan atau menonaktifkan perlindungan penghapusan. Tentukan `false` untuk menonaktifkan perlindungan penghapusan atau `true` untuk mengaktifkannya.

```
aws appconfig update-account-settings --deletion-protection Enabled=value
```

3. Anda dapat meningkatkan interval default hingga maksimal 24 jam. Jalankan perintah berikut untuk menentukan interval baru.

```
aws appconfig update-account-settings --deletion-protection
 Enabled=true,ProtectionPeriodInMinutes=a number between 15 and 1440
```

Melewati atau memaksa pemeriksaan perlindungan penghapusan

Untuk membantu Anda mengelola perlindungan penghapusan, DeleteEnvironmentdan DeleteConfigurationProfile APIs sertakan parameter yang disebut. `DeletionProtectionCheck` Parameter ini mendukung nilai-nilai berikut:

- `BYPASS`: Instruksikan AWS AppConfig untuk melewati pemeriksaan perlindungan penghapusan dan menghapus profil konfigurasi meskipun perlindungan penghapusan akan mencegahnya.
- `APPLY`: Menginstruksikan pemeriksaan perlindungan penghapusan untuk dijalankan, bahkan jika perlindungan penghapusan dinonaktifkan di tingkat akun. `APPLY`juga memaksa pemeriksaan perlindungan penghapusan untuk dijalankan terhadap sumber daya yang dibuat dalam satu jam terakhir, yang biasanya dikecualikan dari pemeriksaan perlindungan penghapusan.

- `ACCOUNT_DEFAULT`: Pengaturan default, yang menginstruksikan AWS AppConfig untuk menerapkan nilai perlindungan penghapusan yang ditentukan dalam API. `UpdateAccountSettings`

> **Note**
>
> Secara default, `DeletionProtectionCheck` lewati profil konfigurasi dan lingkungan yang dibuat dalam satu jam terakhir. Konfigurasi default dimaksudkan untuk mencegah perlindungan penghapusan mengganggu pengujian dan demo yang membuat sumber daya berumur pendek. Anda dapat mengganti perilaku ini dengan meneruskan `DeletionProtectionCheck=APPLY` saat menelepon `DeleteEnvironment` atau`DeleteConfigurationProfile`.

Panduan CLI berikut menggunakan perintah sampel untuk mengilustrasikan cara menggunakan parameter. `DeletionProtectionCheck` Ganti *`ID`* perintah berikut dengan ID untuk AWS AppConfig artefak Anda.

1. Panggil GetLatestConfigurationkonfigurasi yang diterapkan.

```
aws appconfigdata get-latest-configuration --configuration-token $(aws
 appconfigdata start-configuration-session --application-identifier ID --
environment-identifier ID --configuration-profile-identifier ID --query
 InitialConfigurationToken) outfile.txt
```

2. Tunggu 60 detik AWS AppConfig untuk mendaftarkan bahwa konfigurasi aktif.
3. Jalankan perintah berikut untuk memanggil DeleteEnvironmentdan menerapkan perlindungan penghapusan pada lingkungan.

```
aws appconfig delete-environment --environment-id ID --application-id ID --
deletion-protection-check APPLY
```

   Perintah harus mengembalikan kesalahan berikut.

```
An error occurred (BadRequestException) when calling the DeleteEnvironment
 operation: Environment Beta is actively being used in your application and cannot
 be deleted.
```

4. Jalankan perintah berikut untuk melewati perlindungan penghapusan dan menghapus lingkungan.

```
aws appconfig delete-environment --environment-id ID --application-id ID --
deletion-protection-check BYPASS
```

Keamanan di AWS AppConfig

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Systems Manager, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

AWS AppConfig adalah alat di AWS Systems Manager. Untuk memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS AppConfig, lihat Keamanan di AWS Systems Manager. Bagian tersebut menjelaskan cara mengonfigurasi Systems Manager untuk memenuhi tujuan keamanan dan kepatuhan AWS AppConfig.

Terapkan akses hak akses paling rendah

Sebagai praktik terbaik keamanan, 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 Berikan akses hak istimewa paling sedikit di AWS Well-Architected Tool Panduan Pengguna. Untuk informasi selengkapnya tentang fitur AWS AppConfig Agen yang disebutkan di bagian ini, lihatMenggunakan manifes untuk mengaktifkan fitur pengambilan tambahan.

Enkripsi data saat istirahat untuk AWS AppConfig

AWS AppConfig menyediakan enkripsi secara default untuk melindungi data pelanggan saat istirahat menggunakan Kunci 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 menggunakan Kunci 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 Pengembang AWS Key Management Service .

Meskipun Anda tidak dapat menonaktifkan lapisan enkripsi ini atau memilih jenis enkripsi alternatif, Anda dapat menentukan kunci terkelola 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 ada Kunci 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 AppConfig mendukung kunci yang dikelola pelanggan

AWS AppConfig menawarkan 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 AppConfig juga 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 AppConfig menghasilkan kunci AWS KMS data dengan ini `KmsKeyIdentifier` untuk mengenkripsi data pada operasi `StartDeployment` API.

AWS AppConfig akses enkripsi

Saat membuat kunci yang dikelola pelanggan, gunakan kebijakan kunci berikut untuk memastikan bahwa kunci tersebut dapat digunakan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333: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:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey",
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/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.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-ID"
    }
  ]
}
```

Saat memanggil operasi AWS AppConfig StartDeploymentAPI, pemanggilan identitas `StartDeployment` memerlukan kebijakan IAM berikut yang dapat ditetapkan ke pengguna, grup, atau peran:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-ID"
    }
  ]
}
```

Saat memanggil operasi AWS AppConfig GetLatestConfigurationAPI, pemanggilan identitas `GetLatestConfiguration` memerlukan kebijakan berikut yang dapat ditetapkan ke pengguna, grup, atau peran:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-ID"
        }
    ]
}
```

Konteks enkripsi

Konteks enkripsi adalah kumpulan opsional pasangan kunci-nilai yang berisi informasi kontekstual tambahan tentang data.

AWS KMS menggunakan 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 AppConfig konteks 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 dikirim AWS 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. 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 oleh AWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditakdirkan. 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 antarmuka AWS AppConfig, tinjau Pertimbangan dalam Panduan.AWS PrivateLink

AWS AppConfig mendukung 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 PrivateLink Panduan.

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 titik akhir VPC 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 itu AWS Secrets Manager? dalam AWS Secrets Manager User 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 kredensional 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 oleh AWS 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))
```

Pemantauan AWS AppConfig

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS AppConfig dan AWS solusi Anda yang lain. AWS menyediakan alat pemantauan berikut untuk menonton AWS AppConfig, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- Amazon CloudWatch memantau AWS sumber daya Anda dan dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari instans Amazon EC2 Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.
- 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 durabel. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.
- Amazon EventBridge dapat digunakan untuk mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem, seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis mana yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

Topik

- Pencatatan panggilan AWS AppConfig API menggunakan AWS CloudTrail
- Metrik pencatatan untuk panggilan pesawat AWS AppConfig data
- Memantau penyebaran untuk rollback otomatis

Pencatatan panggilan AWS AppConfig API menggunakan AWS CloudTrail

AWS AppConfig terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk AWS AppConfig sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari AWS AppConfig konsol dan panggilan kode ke operasi AWS AppConfig API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat AWS AppConfig, alamat IP dari mana permintaan dibuat, kapan dibuat, dan detail tambahan.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Anda Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. AWS Region Untuk informasi selengkapnya, lihat Bekerja dengan riwayat CloudTrail Acara di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara CloudTrailDanau.

CloudTrail jalan setapak

> Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan Konsol Manajemen AWS Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan. AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. AWS

Region Untuk informasi selengkapnya tentang jejak, lihat Membuat jejak untuk Anda Akun AWS dan Membuat jejak untuk organisasi di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga. Untuk informasi tentang harga Amazon S3, lihat Harga Amazon S3.

CloudTrail Menyimpan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail Lake mengonversi peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC. ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara tingkat lanjut. Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat Bekerja dengan AWS CloudTrail Danau di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih opsi harga yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa, dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

AWS AppConfig peristiwa 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 sering kali merupakan aktivitas bervolume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data. Riwayat CloudTrail peristiwa tidak merekam peristiwa data.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Anda dapat mencatat peristiwa data untuk jenis AWS AppConfig sumber daya menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Tabel di bagian ini menunjukkan jenis sumber daya yang tersedia untuk AWS 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 kustom 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 menggunakan AWS 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 or. AWS CLI CloudTrail APIs CloudTrailKolom Data yang APIs dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk CloudTrail 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 AppConfig acara manajemen di CloudTrail

AWS AppConfig mencatat 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.

AWS AppConfig contoh acara

Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang operasi API yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga peristiwa tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail peristiwa yang menunjukkan StartConfigurationSessionoperasi.


```
{
      "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"
      }
    }
```


Untuk informasi tentang konten CloudTrail rekaman, lihat konten CloudTrail rekaman di Panduan AWS CloudTrail Pengguna.

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 mulai

Aktifkan pencatatan peristiwa AWS AppConfig data di AWS CloudTrail. Prosedur berikut menjelaskan cara mengaktifkan pencatatan metrik untuk AWS AppConfig jejak yang ada di CloudTrail. Untuk informasi tentang cara mengaktifkan CloudTrail pencatatan untuk panggilan paket AWS AppConfig data, lihatAWS AppConfig peristiwa 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 AppConfig panggilan 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.

Memantau penyebaran untuk rollback otomatis

Selama penerapan, Anda dapat mengurangi situasi di mana data konfigurasi yang salah atau salah menyebabkan kesalahan dalam aplikasi Anda dengan menggunakan kombinasi strategi AWS

AppConfig penerapan dan rollback otomatis berdasarkan alarm Amazon. CloudWatch Setelah dikonfigurasi, jika satu atau lebih CloudWatch alarm masuk ke `INSUFFICIENT_DATA` status `ALARM` atau selama penerapan, AWS AppConfig secara otomatis memutar kembali data konfigurasi Anda ke versi sebelumnya, sehingga mencegah pemadaman aplikasi atau kesalahan.

> **Note**
>
> Penerapan tidak otomatis diputar kembali jika tindakan telah dinonaktifkan di CloudWatch alarm terkait.
> Anda dapat menonaktifkan dan mengaktifkan alarm dengan menggunakan tindakan DisableAlarmActionsdan EnableAlarmActionsAPI, atau enable-alarm-actionsperintah disable-alarm-actionsdan di AWS CLI.

Anda juga dapat memutar kembali konfigurasi dengan memanggil operasi StopDeploymentAPI saat penerapan masih berlangsung.

> **Important**
>
> Untuk penerapan yang berhasil diselesaikan, AWS AppConfig juga mendukung pengembalian data konfigurasi ke versi sebelumnya dengan menggunakan `AllowRevert` parameter dengan operasi API. StopDeployment Untuk beberapa pelanggan, kembali ke konfigurasi sebelumnya setelah penerapan yang berhasil menjamin data akan sama seperti sebelum penerapan. Mengembalikan juga mengabaikan monitor alarm, yang dapat mencegah kemajuan maju selama keadaan darurat aplikasi. Untuk informasi selengkapnya, lihat Mengembalikan konfigurasi.

Untuk mengonfigurasi rollback otomatis, Anda menentukan Nama Sumber Daya Amazon (ARN) dari satu atau beberapa CloudWatch metrik di bidang CloudWatch alarm saat Anda membuat (atau mengedit) lingkungan. AWS AppConfig Untuk informasi selengkapnya, lihat Membuat lingkungan untuk aplikasi Anda di AWS AppConfig.

> **Note**
>
> Jika Anda menggunakan solusi pemantauan pihak ketiga (misalnya, Datadog atau New Relic), Anda dapat membuat AWS AppConfig ekstensi yang memeriksa alarm di titik `AT_DEPLOYMENT_TICK` tindakan dan, sebagai pagar pembatas keselamatan, memutar

kembali penerapan jika memicu alarm. Untuk informasi selengkapnya, lihat contoh integrasi Datadog dan New Relic berikut tentang: GitHub

- Datadog
- New Relic

Untuk informasi selengkapnya tentang AWS AppConfig ekstensi, lihat topik berikut:

- Memperluas AWS AppConfig alur kerja menggunakan ekstensi
- Walkthrough: Membuat ekstensi khusus AWS AppConfig

Metrik yang disarankan untuk memantau rollback otomatis

Metrik yang Anda pilih untuk dipantau akan bergantung pada perangkat keras dan perangkat lunak yang digunakan oleh aplikasi Anda. AWS AppConfig pelanggan sering memantau metrik berikut. Untuk daftar lengkap metrik yang direkomendasikan yang dikelompokkan berdasarkan Layanan AWS, lihat Alarm yang disarankan di Panduan Pengguna Amazon CloudWatch .

Setelah menentukan metrik yang ingin dipantau, gunakan CloudWatch untuk mengonfigurasi alarm. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon.

Layanan	Metrik	Detail
Amazon API Gateway	4 XXError	Alarm ini mendeteksi tingkat kesalahan sisi klien yang tinggi. Hal ini dapat menunjukkan adanya masalah dalam parameter otorisasi atau permintaan klien. Hal ini juga bisa berarti bahwa sebuah sumber daya telah dihapus atau klien meminta sesuatu yang tidak ada. Pertimbangkan untuk mengaktifkan Amazon CloudWatch Logs dan memeriksa kesalahan

Layanan	Metrik	Detail
		apa pun yang mungkin menyebabkan kesalahan 4XX. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per sumber daya dan metode dan mempersempit sumber kesalahan. Kesalahan juga dapat disebabkan karena dilanggarnya batas throttling yang telah dikonfigurasi sebelumnya.
Amazon API Gateway	5 XXError	Alarm ini membantu mendeteksi tingkat kesalahan sisi server yang tinggi. Hal ini dapat menunjukkan bahwa ada sesuatu yang salah pada backend API, jaringan, atau integrasi antara gateway API dan backend API.

Layanan	Metrik	Detail
Amazon API Gateway	Latensi	Alarm ini mendeteksi latensi tinggi dalam satu panggung. Temukan nilai metrik `IntegrationLatency` untuk memeriksa latensi API backend. Jika kedua metrik sebagian besar selaras, maka API backend akan menjadi sumber latensi yang lebih tinggi dan Anda harus menyelidiki masalah yang terjadi di sana. Pertimbangkan juga mengaktifkan CloudWatch Log dan memeriksa kesalahan yang mungkin menyebabkan latensi tinggi.
Amazon EC2 Auto Scaling	GroupInServiceCapacity	Alarm ini membantu mendeteksi ketika kapasitas dalam grup berada di bawah kapasitas yang diinginkan yang diperlukan untuk beban kerja Anda. Untuk memecahkan masalah tersebut, Anda perlu memeriksa aktivitas penskalaan Anda, apakah ada kegagalan peluncuran yang terjadi, dan konfirmasikan apakah konfigurasi kapasitas yang Anda inginkan sudah dibuat dengan benar.

Layanan	Metrik	Detail
Amazon EC2	CPUUtilization	Alarm ini membantu memantau pemanfaatan CPU dari instans EC2. Tergantung pada aplikasinya, tingkat pemanfaatan yang tinggi mungkin akan normal secara terus-menerus. Tetapi jika performanya menurun, dan aplikasi tidak dibatasi oleh disk I/O, memori, atau sumber daya jaringan, maka CPU yang telah mencapai batas atasnya mungkin akan menunjukkan hambatan sumber daya atau masalah performa aplikasi.
Amazon ECS	CPUReservation	Alarm ini membantu Anda mendeteksi reservasi CPU yang tinggi dari cluster ECS. Reservasi CPU yang tinggi mungkin menunjukkan bahwa cluster kehabisan registrasi CPUs untuk tugas tersebut.
Amazon ECS	HTTPCode_Target_5XX_Count	Alarm ini membantu Anda mendeteksi jumlah kesalahan sisi server yang tinggi untuk layanan ECS. Hal ini dapat menunjukkan bahwa telah terjadi kesalahan yang menyebabkan server tidak dapat melayani permintaan.

Layanan	Metrik	Detail
Amazon EKS dengan wawasan Container	node_cpu_utilization	Alarm ini membantu mendeteksi pemanfaatan CPU yang tinggi di node pekerja kluster Amazon EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda perlu mengganti simpul pekerja Anda dengan instans yang memiliki CPU lebih besar atau Anda perlu menskalakan sistem secara horizontal.
Amazon EKS dengan wawasan Container	node_memory_utilization	Alarm ini membantu dalam mendeteksi pemanfaatan memori yang tinggi di node pekerja dari cluster Amazon EKS. Jika pemanfaatannya berada dalam level yang tinggi secara terus-menerus, hal ini mungkin menunjukkan bahwa Anda perlu menskalakan jumlah replika pod, atau Anda harus mengoptimalkan aplikasi Anda.

Layanan	Metrik	Detail
Amazon EKS dengan wawasan Container	pod_cpu_utilization_over_pod_limit	Alarm ini membantu mendeteksi pemanfaatan CPU yang tinggi di pod cluster Amazon EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda harus menaikkan batas CPU untuk pod yang terpengaruh.
Amazon EKS dengan wawasan Container	pod_memory_utilization_over_pod_limit	Alarm ini membantu mendeteksi pemanfaatan CPU yang tinggi di pod cluster Amazon EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda harus menaikkan batas CPU untuk pod yang terpengaruh.
AWS Lambda	Kesalahan	Alarm ini mendeteksi jumlah kesalahan yang tinggi. Kesalahan tersebut mencakup pengecualian yang dibuat oleh kode serta pengecualian yang dibuat oleh runtime Lambda.
AWS Lambda	Pembatasan	Alarm ini mendeteksi sejumlah besar permintaan pemanggilan yang dibatasi. Throttling terjadi ketika tidak ada konkurensi yang tersedia untuk menaikkan skala.

Layanan	Metrik	Detail
Wawasan Lambda	memory_utilization	Alarm ini digunakan untuk mendeteksi apakah pemanfaatan memori fungsi lambda mendekati batas yang dikonfigurasi.
Amazon S3	4xxErrors	Alarm ini membantu kami melaporkan jumlah total kode status kesalahan 4xx yang dibuat sebagai tanggapan atas permintaan klien. 403 kode kesalahan mungkin menunjukkan kebijakan IAM yang salah, dan 404 kode kesalahan mungkin menunjukkan aplikasi klien yang berperilaku salah, misalnya.
Amazon S3	5xxErrors	Alarm ini membantu Anda mendeteksi sejumlah besar kesalahan sisi server. Kesalahan ini menunjukkan bahwa ada klien yang sudah membuat permintaan dan tidak dapat diselesaikan oleh server. Hal ini dapat membantu Anda untuk mengkorelasikan masalah yang dihadapi aplikasi Anda karena S3.

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
Contoh ekstensi AT_DEPLOYMENT_TICK baru	Titik `AT_DEPLOYMENT_TICK` tindakan mendukung integrasi pemantauan pihak ketiga. `AT_DEPLOYMENT_TICK` dipanggil selama orkestrasi pemrosesan penerapan konfigurasi. Jika Anda menggunakan solusi pemantauan pihak ketiga seperti New Relic, Anda dapat membuat AWS AppConfig ekstensi yang memeriksa alarm di titik `AT_DEPLOYMENT_TICK` tindakan dan, sebagai pagar pembatas keselamatan, memutar kembali penerapan jika memicu alarm. Untuk informasi selengkapnya tentang AWS AppConfig ekstensi, lihat Memperluas AWS AppConfig alur kerja menggunakan ekstensi. Untuk informasi selengkapnya tentang ekstensi kustom, lihat Panduan: Membuat ekstensi kustom AWS AppConfig. Untuk melihat contoh kode	Februari 24, 2026

	AWS AppConfig ekstensi yang menggunakan titik `AT_DEPLOYMENT_TICK` tindakan untuk diintegrasikan dengan New Relic, lihat contoh New Relic pada. GitHub	
Versi baru ekstensi AWS AppConfig Agen Lambda	Agen telah diperbarui dengan penyempurnaan kecil dan perbaikan bug. Untuk melihat Nama Sumber Daya Amazon (ARNs) baru untuk ekstensi, lihat Versi ekstensi Lambda AWS AppConfig Agen yang tersedia.	Februari 20, 2026
Deprecated: AWS AppConfig ekstensi untuk Terbukti CloudWatch	AWS AppConfig Ekstensi untuk CloudWatch Evidently tidak lagi didukung.	Februari 20, 2026
IPv6 dukungan	Semua AWS AppConfig APIs sekarang sepenuhnya mendukung IPv4 dan IPv6 menelepon. Untuk informasi selengkapnya, lihat Memahami IPv6 dukungan.	April 23, 2025

Topik baru: Menyimpan versi bendera fitur sebelumnya ke versi baru	Saat Anda memperbarui bendera fitur, AWS AppConfig secara otomatis menyimpan perubahan Anda ke versi baru. Jika Anda ingin menggunakan versi bendera fitur sebelumnya, Anda harus menyalinnya ke versi draf dan kemudian menyimpannya. Anda tidak dapat mengedit dan menyimpan perubahan ke versi bendera sebelumnya tanpa menyimpannya ke versi baru. Untuk informasi selengkapnya, lihat Menyimpan versi tanda fitur sebelumnya ke versi baru.	April 15, 2025

Topik baru: Contoh bendera fitur untuk mode pengembangan lokal AWS AppConfig Agen	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. Untuk membantu Anda lebih memahami cara menggunakan mode pengembangan lokal, panduan ini sekarang berisi topik dengan contoh tanda fitur. Untuk informasi selengkapnya, lihat Contoh tanda fitur untuk mode pengembangan lokal AWS AppConfig Agen.	Februari 18, 2025
Topik baru: Membuat profil konfigurasi untuk sumber data non-native	Topik menjelaskan proses tingkat tinggi untuk menggunakan AWS AppConfig ekstensi untuk mengambil data konfigurasi dari sumber yang tidak didukung secara asli, termasuk AWS layanan lain seperti Amazon RDS dan Amazon DynamoDB, serta sumber pihak ketiga seperti,, atau repo lokal. GitHub GitLab Untuk informasi selengkapnya, lihat Membuat profil konfigurasi untuk sumber data non-native	Desember 19, 2024

Topik diperbarui: Regex tetap dalam referensi jenis bendera fitur	Skema json dalam referensi tipe flag fitur sebelumnya menunjukkan pola regex berikut di berbagai tempat:. `"^[a-z][a-zA-Z\\d-_]{0,63}$"` Pola regex yang benar adalah. `"^[a-z][a-zA-Z\\d_-]{0,63}$"` Tanda hubung terdaftar setelah garis bawah. Untuk informasi selengkapnya, lihat Memahami referensi jenis untuk AWS. AppConfig. FeatureFlags	Desember 18, 2024
Topik yang diperbarui: Sampel variabel lingkungan ditambahkan	Tabel yang menjelaskan variabel lingkungan dalam topik berikut diperbarui untuk menyertakan sampel: • (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon ECS dan Amazon EKS • (Opsional) Menggunakan variabel lingkungan untuk mengonfigurasi AWS AppConfig Agen untuk Amazon EC2 • Mengkonfigurasi ekstensi AWS AppConfig Agen Lambda	Desember 12, 2024

Bagian baru: Memahami operator split	Bagian baru menggunakan contoh untuk menjelaskan cara kerja `split` operator untuk aturan flag fitur multi-varian. Untuk informasi selengkapnya, lihat Memahami aturan tanda fitur multi-varian.	November 22, 2024

Titik tindakan ekstensi baru: AT_DEPLOYMENT_TICK	AWS AppConfig meluncurkan titik tindakan baru untuk pengguna yang membuat ekstensi khusus. Titik `AT_DEPLOYMENT_TICK` tindakan mendukung integrasi pemantauan pihak ketiga. `AT_DEPLOYMENT_TICK` dipanggil selama orkestrasi pemrosesan penerapan konfigurasi. Jika Anda menggunakan solusi pemantauan pihak ketiga (misalnya, Datadog), Anda dapat membuat AWS AppConfig ekstensi yang memeriksa alarm di titik `AT_DEPLOYMENT_TICK` tindakan dan, sebagai pagar pembatas keselamatan, memutar kembali penerapan jika memicu alarm. Untuk informasi selengkapnya tentang AWS AppConfig ekstensi, lihat Memperluas AWS AppConfig alur kerja menggunakan ekstensi. Untuk informasi selengkapnya tentang ekstensi kustom, lihat Panduan: Membuat ekstensi kustom AWS AppConfig. Untuk melihat contoh kode AWS AppConfig ekstensi yang menggunakan titik `AT_DEPLOYMENT_TICK` tindakan untuk diintegrasikan	November 22, 2024

	dengan Datadog, lihat aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub	
Topik baru: pertimbangan penggunaan AWS AppConfig seluler	Topik baru dalam panduan ini menjelaskan pertimbangan penting untuk menggunakan bendera AWS AppConfig fitur dengan perangkat seluler. Untuk informasi selengkapnya, lihat pertimbangan penggunaan AWS AppConfig seluler.	November 21, 2024
Fitur baru: Perlindungan AWS AppConfig Penghapusan	AWS AppConfig sekarang menyediakan pengaturan akun untuk membantu mencegah pengguna menghapus lingkungan dan profil konfigurasi yang digunakan secara tidak sengaja. Untuk informasi selengkapnya, lihat Mengonfigurasi AWS AppConfig perlindungan penghapusan.	Agustus 28, 2024
Versi baru ekstensi AWS AppConfig Agen Lambda	Agen telah diperbarui dengan penyempurnaan kecil dan perbaikan bug. Untuk melihat Nama Sumber Daya Amazon (ARNs) baru untuk ekstensi, lihat Versi ekstensi Lambda AWS AppConfig Agen yang tersedia.	Agustus 9, 2024

Contoh kode baru untuk mengambil varian bendera	Untuk informasi selengkapnya, lihat Menggunakan AWS AppConfig Agen untuk mengambil tanda fitur dengan varian.	Agustus 9, 2024
Versi baru ekstensi AWS AppConfig Agen Lambda	Agen telah diperbarui untuk mendukung target flag fitur, varian, dan split. Untuk melihat Nama Sumber Daya Amazon (ARNs) baru untuk ekstensi, lihat Versi ekstensi Lambda AWS AppConfig Agen yang tersedia.	Juli 23, 2024

Fitur baru: Bendera fitur multi-varian	Bendera fitur multi-varian memungkinkan Anda untuk menentukan satu set nilai bendera yang mungkin untuk dikembalikan untuk permintaan. Anda juga dapat mengonfigurasi status yang berbeda (diaktifkan atau dinonaktifkan) untuk bendera multi-varian. Saat meminta flag yang dikonfigurasi dengan varian, aplikasi Anda menyediakan konteks yang AWS AppConfig mengeval uasi terhadap serangkai an aturan yang ditetapkan pengguna. Bergantung pada konteks yang ditentukan dalam permintaan dan aturan yang ditetapkan untuk varian, AWS AppConfig mengembal ikan nilai flag yang berbeda ke aplikasi. Untuk informasi selengkapnya, lihat Membuat flag fitur multi-varian.	Juli 23, 2024
Versi baru ekstensi AWS AppConfig Agen Lambda	Agen telah diperbarui dengan penyempurnaan kecil dan perbaikan bug. Untuk melihat Nama Sumber Daya Amazon (ARNs) baru untuk ekstensi, lihat Versi ekstensi Lambda AWS AppConfig Agen yang tersedia.	Februari 28, 2024

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 • 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	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 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.	Januari 18, 2024

AWS AppConfig sekarang mendukung 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. Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses. AWS AppConfig Untuk informasi selengkapnya, lihat Akses AWS AppConfig menggunakan 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 payload 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

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 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 A/B eksperimen 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	13 September 2022

	menggunakan ekstensi, lihat Melakukan peluncuran dan A/B eksperimen dengan CloudWatch Terbukti di CloudWatch Panduan Pengguna Amazon.	
Pengakhiran tindakan API `GetConfiguration`	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, StartConfigurationSessiondan GetLatestConfiguration. Layanan pesawat data juga menggunakan 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 yang berbeda (ARNs). 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 yang berbeda (ARNs). 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 AWS Region 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 ARM64 untuk prosesor (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 `StartConfigurationSession` dan `GetLatestConfiguration` APIs sebagai gantinya. Untuk informasi selengkapnya tentang ini APIs 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) diperlukan untuk membuat Lambda di wilayah tersebut. Untuk informasi lebih lanjut tentang ARN wilayah Asia Pasifik (Osaka), lihat Menambahkan 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 menyederhanakan penggunaan AWS AppConfig sekaligus mengurangi biaya. Pengurangan biaya dihasilkan dari lebih sedikit panggilan API ke AWS AppConfig layanan dan, secara terpisah, mengurangi biaya dari waktu pemrosesan 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 , alat di AWS Systems Manager, untuk membuat, mengelola, dan menyebarkan konfigurasi aplikasi dengan cepat. AWS AppConfig mendukung penerapan terkontrol untuk aplikasi dari berbagai ukuran dan termasuk pemeriksaan validasi bawaan dan pemantauan. Anda dapat menggunakan AWS AppConfig dengan aplikasi yang dihosting di instans EC2, wadah AWS Lambda, aplikasi seluler, atau perangkat IoT.	31 Juli 2020

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.