User Guide

AWS AppConfig

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

AWS AppConfig: User Guide

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist AWS AppConfig? ... 1
AWS AppConfig Anwendungsfälle ... 2
Vorteile der Verwendung AWS AppConfig ... 3
Wie AWS AppConfig funktioniert ... 4
Erste Schritte mit AWS AppConfig ... 6
SDKs ... 6
Preisgestaltung für AWS AppConfig ... 7
AWS AppConfig Kontingente ... 7
Einrichten AWS AppConfig ... 8
Melden Sie sich an für ein AWS-Konto ... 8
Erstellen eines Benutzers mit Administratorzugriff ... 8
Erteilen programmgesteuerten Zugriffs ... 10
Unterstützung verstehen IPv6 ... 11
(Empfohlen) Konfigurieren Sie die Berechtigungen für das automatische Rollback ... 12
Schritt 1: Erstellen Sie die Berechtigungsrichtlinie für ein Rollback auf der Grundlage von Alarmen CloudWatch ... 13
Schritt 2: Erstellen Sie die IAM-Rolle für Rollback auf der Grundlage von Alarmen CloudWatch ... 14
Schritt 3: Hinzufügen einer Vertrauensstellung ... 15
Erstellen ... 17
Grundlegendes zur IAM-Rolle des Konfigurationsprofils ... 19
Einen Namespace erstellen ... 20
Eine AWS AppConfig Anwendung (Konsole) erstellen ... 21
Eine AWS AppConfig Anwendung erstellen (Befehlszeile) ... 22
Erstellen von Umgebungen ... 23
Eine AWS AppConfig Umgebung (Konsole) erstellen ... 24
Eine AWS AppConfig Umgebung erstellen (Befehlszeile) ... 25
Erstellen eines Konfigurationsprofils in AWS AppConfig ... 27
Erstellen eines Feature-Flag-Konfigurationsprofils ... 31
Erstellen eines Freiform-Konfigurationsprofils ... 64
Erstellen eines Konfigurationsprofils für nicht-native Datenquellen ... 80
Bereitstellen ... 83
Arbeiten mit Bereitstellungsstrategien ... 84
Verwenden vordefinierter Bereitstellungsstrategien ... 87

Erstellen einer Bereitstellungsstrategie ... 89
Bereitstellen einer Konfiguration ... 93
Stellen Sie eine Konfiguration bereit (Konsole) ... 94
Stellen Sie eine Konfiguration bereit (Befehlszeile) ... 95
Bereitstellen mit CodePipeline ... 98
Wie funktioniert die Integration ... 99
Konfiguration rückgängig machen ... 100
Wird abgerufen ... 102
Was ist AWS AppConfig Agent? ... 103
Wie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten ... 105
Verwenden Sie den AWS AppConfig Agenten mit AWS Lambda ... 105
AWS AppConfig Agent mit Amazon EC2 - und lokalen Maschinen verwenden ... 199
AWS AppConfig Agent mit Amazon ECS und Amazon EKS verwenden ... 220
Feature-Flags werden abgerufen ... 241
Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren ... 244
Generierung eines Clients mithilfe der OpenAPI-Spezifikation ... 256
Mit dem lokalen Entwicklungsmodus des Agenten AWS AppConfig arbeiten ... 258
Überlegungen zur Nutzung von Browsern und Mobilgeräten ... 263
Konfigurationsdaten und Abrufen von Flaggen ... 264
Authentifizierung und Amazon Cognito ... 264
Caching ... 265
Segmentierung ... 266
Bandbreite (mobile Anwendungsfälle) ... 266
Zusätzliche Anwendungsfälle kennzeichnen ... 267
Konfigurationsdaten werden ohne AWS AppConfig Agent abgerufen ... 267
(Beispiel) Abrufen einer Konfiguration durch Aufrufen AWS AppConfig APIs ... 269
Erweiterung von AWS AppConfig Arbeitsabläufen ... 272
AWS AppConfig Erweiterungen verstehen ... 272
Schritt 1: Ermitteln Sie, was Sie mit Erweiterungen machen möchten ... 273
Schritt 2: Ermitteln Sie, wann die Erweiterung ausgeführt werden soll ... 274
Schritt 3: Erstellen Sie eine Erweiterungszuordnung ... 275
Schritt 4: Stellen Sie eine Konfiguration bereit und überprüfen Sie, ob die Erweiterungsaktionen ausgeführt wurden ... 276
Mit erstellten Erweiterungen AWS arbeiten ... 276
Verwenden der Amazon CloudWatch Evidenly-Erweiterung ... 277
EventBridge Erweiterung „ AWS AppConfig Deployment Events to Amazon“ verwenden ... 278

Verwenden der AWS AppConfig Bereitstellungsereignisse für die Amazon SNS SNS-Erweiterung ... 280
Verwenden der AWS AppConfig Bereitstellungsereignisse für die Amazon SQS SQS-Erweiterung ... 283
Verwendung der Jira-Erweiterung ... 286
Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig . 291
Schritt 1: Erstellen Sie eine Lambda-Funktion für eine benutzerdefinierte Erweiterung AWS AppConfig ... 293
Schritt 2: Konfigurieren Sie die Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung ... 299
Schritt 3: Erstellen Sie eine benutzerdefinierte AWS AppConfig Erweiterung ... 301
Schritt 4: Erstellen Sie eine Erweiterungszuordnung für eine benutzerdefinierte Erweiterung AWS AppConfig ... 305
Codebeispiele ... 308
Erstellen oder Aktualisieren einer Freiformkonfiguration, die im gehosteten Konfigurationsspeicher gespeichert ist ... 308
Erstellen eines Konfigurationsprofils für ein in Secrets Manager gespeichertes Geheimnis ... 311
Bereitstellen eines Konfigurationsprofils ... 312
Verwenden von AWS AppConfig Agent zum Lesen eines Freiform-Konfigurationsprofils ... 317
Verwenden AWS AppConfig des Agenten zum Lesen eines bestimmten Feature-Flags ... 319
Verwenden von AWS AppConfig Agent zum Abrufen eines Feature-Flags mit Varianten ... 320
Verwenden der GetLatestConfiguration API-Aktion zum Lesen eines Freiform-Konfigurationsprofils ... 322
Säubere deine Umgebung ... 329
Löschschutz ... 335
Umgehen oder Erzwingen einer Löschschutzprüfung ... 336
Sicherheit ... 339
Implementieren des Zugriffs mit geringsten Berechtigungen ... 339
Verschlüsselung der Daten im Ruhezustand für AWS AppConfig ... 340
AWS PrivateLink ... 345
Überlegungen ... 345
Erstellen eines Schnittstellenendpunkts ... 345
Erstellen einer Endpunktrichtlinie ... 346
Secrets Manager Schlüsselrotation ... 347
Einrichtung der automatischen Rotation von Secrets Manager Manager-Geheimnissen, bereitgestellt von AWS AppConfig ... 347

Überwachen 350
CloudTrail protokolliert 351
AWS AppConfig Datenereignisse in CloudTrail 352
AWS AppConfig Managementereignisse in CloudTrail 354
AWS AppConfig Beispiele für Ereignisse 354
Protokollierung von Metriken für AWS AppConfig Datenebenenanrufe 356
Einen Alarm für eine CloudWatch Metrik erstellen 359
Bereitstellungen im Hinblick auf automatisches Rollback überwachen 359
Empfohlene Metriken zur Überwachung des automatischen Rollbacks 360
Dokumentverlauf 367
..... cccxcvi

Was ist AWS AppConfig?

AWS AppConfig Feature-Flags und dynamische Konfigurationen helfen Softwareentwicklern dabei, das Anwendungsverhalten in Produktionsumgebungen schnell und sicher anzupassen, ohne dass vollständige Codebereitstellungen erforderlich sind. AWS AppConfig beschleunigt die Häufigkeit von Softwareveröffentlichungen, verbessert die Ausfallsicherheit von Anwendungen und hilft Ihnen, neu auftretende Probleme schneller zu lösen. Mithilfe von Feature-Flags können Sie schrittweise neue Funktionen für Benutzer bereitstellen und die Auswirkungen dieser Änderungen messen, bevor Sie die neuen Funktionen vollständig für alle Benutzer bereitstellen. Mithilfe von Betriebsflags und dynamischen Konfigurationen können Sie Sperrlisten und Zulassungslisten aktualisieren, Grenzwerte einschränken, den Umfang der Protokollierung einschränken und andere betriebliche Optimierungen vornehmen, um schnell auf Probleme in Produktionsumgebungen zu reagieren.

> **Note**
>
> AWS AppConfig ist ein Tool in. AWS Systems Manager

Verbessern Sie die Effizienz und veröffentlichen Sie Änderungen schneller

Die Verwendung von Feature-Flags mit neuen Funktionen beschleunigt den Prozess der Veröffentlichung von Änderungen in Produktionsumgebungen. Anstatt sich auf langlebige Entwicklungszweige zu verlassen, die vor einer Veröffentlichung komplizierte Zusammenführungen erfordern, ermöglichen Ihnen Feature-Flags, Software mithilfe von Trunk-basierter Entwicklung zu schreiben. Mit Feature-Flags können Sie Vorabversions-Code sicher in einer CI/CD-Pipeline bereitstellen, die für Benutzer unsichtbar ist. Wenn Sie bereit sind, die Änderungen zu veröffentlichen, können Sie das Feature-Flag aktualisieren, ohne neuen Code bereitzustellen. Nach Abschluss des Starts kann das Flag weiterhin als Blockschalter dienen, um eine neue Funktion oder Funktion zu deaktivieren, ohne dass die Codebereitstellung rückgängig gemacht werden muss.

Vermeiden Sie unbeabsichtigte Änderungen oder Ausfälle mit integrierten Sicherheitsfunktionen

AWS AppConfig bietet die folgenden Sicherheitsfunktionen, mit denen Sie verhindern können, dass Sie Feature-Flags aktivieren oder Konfigurationsdaten aktualisieren, die zu Anwendungsausfällen führen könnten.

- Validatoren: Ein Validator stellt sicher, dass Ihre Konfigurationsdaten syntaktisch und semantisch korrekt sind, bevor die Änderungen in Produktionsumgebungen implementiert werden.

- Bereitstellungsstrategien: Eine Bereitstellungsstrategie ermöglicht es Ihnen, Änderungen an Produktionsumgebungen langsam innerhalb von Minuten oder Stunden zu veröffentlichen.
- Überwachung und automatisches Rollback: AWS AppConfig lässt sich in Amazon integrieren CloudWatch , um Änderungen an Ihren Anwendungen zu überwachen. Wenn Ihre Anwendung aufgrund einer fehlerhaften Konfigurationsänderung fehlerhaft wird und diese Änderung einen Alarm auslöst, wird die Änderung AWS AppConfig automatisch rückgängig gemacht CloudWatch, um die Auswirkungen auf Ihre Anwendungsbenutzer zu minimieren.

Sichere und skalierbare Feature-Flag-Bereitstellungen

AWS AppConfig lässt sich in AWS Identity and Access Management (IAM) integrieren, um einen detaillierten, rollenbasierten Zugriff auf den Service zu ermöglichen. AWS AppConfig lässt sich auch mit AWS Key Management Service (AWS KMS) für Verschlüsselung und Auditing integrieren. AWS CloudTrail Bevor sie für externe Kunden freigegeben wurden, wurden alle AWS AppConfig Sicherheitskontrollen zunächst mit internen Kunden entwickelt und von diesen validiert, die den Service in großem Umfang nutzen.

AWS AppConfig Anwendungsfälle

Trotz der Tatsache, dass der Inhalt der Anwendungskonfiguration von Anwendung zu Anwendung stark variieren kann, AWS AppConfig unterstützt es die folgenden Anwendungsfälle, die ein breites Spektrum an Kundenanforderungen abdecken:

- Feature-Flags und Toggles — Stellen Sie Ihren Kunden neue Funktionen sicher in einer kontrollierten Umgebung zur Verfügung. Machen Sie Änderungen sofort rückgängig, wenn Sie auf ein Problem stoßen.
- Anwendungsoptimierung — Führen Sie Anwendungsänderungen sorgfältig ein und testen Sie gleichzeitig, wie sich diese Änderungen auf Benutzer in Produktionsumgebungen auswirken.
- Zulassungsliste oder Sperrliste — Steuern Sie den Zugriff auf Premium-Funktionen oder blockieren Sie sofort bestimmte Benutzer, ohne neuen Code bereitstellen zu müssen.
- Zentralisierter Konfigurationsspeicher — Sorgen Sie dafür, dass Ihre Konfigurationsdaten über alle Ihre Workloads hinweg organisiert und konsistent sind. Sie können AWS AppConfig die Bereitstellung von Konfigurationsdaten verwenden, die im AWS AppConfig gehosteten Konfigurationsspeicher AWS Secrets Manager, im Systems Manager Parameter Store oder in Amazon S3 gespeichert sind.

Vorteile der Verwendung AWS AppConfig

AWS AppConfig bietet die folgenden Vorteile für Ihr Unternehmen:

- Reduzieren Sie unerwartete Ausfallzeiten für Ihre Kunden

  AWS AppConfig reduziert die Ausfallzeiten von Anwendungen, indem Sie Regeln zur Validierung Ihrer Konfiguration erstellen können. Konfigurationen, die nicht gültig sind, können nicht bereitgestellt werden. AWS AppConfig bietet die folgenden zwei Optionen für die Validierung von Konfigurationen:

  - Für die syntaktische Validierung können Sie ein JSON-Schema verwenden. AWS AppConfig validiert Ihre Konfiguration mithilfe des JSON-Schemas, um sicherzustellen, dass die Konfigurationsänderungen den Anwendungsanforderungen entsprechen.
  - Zur semantischen Validierung AWS AppConfig können Sie eine AWS Lambda Funktion aufrufen, deren Eigentümer Sie sind, um die Daten in Ihrer Konfiguration zu validieren.
- Implementieren Sie Änderungen schnell für eine Reihe von Zielen

  AWS AppConfig vereinfacht die Verwaltung von Anwendungen in großem Umfang, indem Konfigurationsänderungen von einem zentralen Ort aus implementiert werden. AWS AppConfig unterstützt Konfigurationen, die im AWS AppConfig gehosteten Konfigurationsspeicher, im Systems Manager Parameter Store, in Systems Manager (SSM) -Dokumenten und in Amazon S3 gespeichert sind. Sie können es AWS AppConfig mit Anwendungen verwenden, die auf EC2 Instances AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten gehostet werden.

  Ziele müssen nicht mit dem Systems Manager SSM Agent oder dem für andere Systems Manager Manager-Tools erforderlichen IAM-Instanzprofil konfiguriert werden. Das bedeutet, dass dies mit nicht AWS AppConfig verwalteten Instanzen funktioniert.
- Unterbrechungsfreie Aktualisierung von Anwendungen

  AWS AppConfig implementiert zur Laufzeit Konfigurationsänderungen an Ihren Zielen, ohne dass ein aufwändiger Build-Prozess erforderlich ist oder Ihre Ziele außer Betrieb genommen werden müssen.
- Steuern der Bereitstellung von Änderungen in Ihrer Anwendung

  AWS AppConfig Ermöglicht es Ihnen, bei der Implementierung von Konfigurationsänderungen an Ihren Zielen das Risiko zu minimieren, indem Sie eine Bereitstellungsstrategie verwenden. Mithilfe von Bereitstellungsstrategien können Sie Konfigurationsänderungen langsam in Ihrer Flotte

einführen. Wenn bei der Bereitstellung ein Problem auftritt, können Sie die Konfigurationsänderung rückgängig machen, bevor sie die meisten Ihrer Hosts erreicht.

Wie AWS AppConfig funktioniert

Dieser Abschnitt enthält eine allgemeine Beschreibung der AWS AppConfig Funktionsweise und der ersten Schritte.

1. Identifizieren Sie die Konfigurationswerte im Code, den Sie in der Cloud verwalten möchten

   Bevor Sie mit der Erstellung von AWS AppConfig Artefakten beginnen, empfehlen wir Ihnen, die Konfigurationsdaten in Ihrem Code zu identifizieren, die Sie dynamisch verwalten möchten AWS AppConfig. Gute Beispiele hierfür sind Feature-Flags oder Toggles, Zulassungs- und Sperrlisten, ausführliche Protokollierung, Dienstbeschränkungen und Drosselungsregeln, um nur einige zu nennen.

   Wenn Ihre Konfigurationsdaten bereits in der Cloud vorhanden sind, können Sie die AWS AppConfig Validierungs-, Bereitstellungs- und Erweiterungsfunktionen nutzen, um die Verwaltung der Konfigurationsdaten weiter zu optimieren.
2. Erstellen Sie einen Anwendungs-Namespace

   Um einen Namespace zu erstellen, erstellen Sie ein AWS AppConfig Artefakt, das als Anwendung bezeichnet wird. Eine Anwendung ist einfach ein organisatorisches Konstrukt wie ein Ordner.
3. Erstellen von Umgebungen.

   Für jede AWS AppConfig Anwendung definieren Sie eine oder mehrere Umgebungen. Eine Umgebung ist eine logische Gruppierung von Zielen, z. B. Anwendungen in einer `Beta` `Production` Oder-Umgebung, AWS Lambda Funktionen oder Containern. Sie können auch Umgebungen für Anwendungsunterkomponenten wie, `Web` und `Mobile` definieren. `Back-end`

   Sie können CloudWatch Amazon-Alarme für jede Umgebung konfigurieren. Das System überwacht Alarme während einer Konfigurationsbereitstellung. Wenn ein Alarm ausgelöst wird, setzt das System die Konfiguration zurück.
4. Konfigurationsprofil erstellen

   Ein Konfigurationsprofil umfasst unter anderem eine URI, die es ermöglicht, Ihre Konfigurationsdaten AWS AppConfig an ihrem gespeicherten Speicherort zu finden, und

einen Profiltyp. AWS AppConfig unterstützt zwei Typen von Konfigurationsprofilen: Feature-Flags und Freiform-Konfigurationen. Feature-Flag-Konfigurationsprofile speichern ihre Daten im AWS AppConfig gehosteten Konfigurationsspeicher, und der URI ist einfach`hosted`. Bei Freiform-Konfigurationsprofilen können Sie Ihre Daten im AWS AppConfig gehosteten Konfigurationsspeicher oder in einem beliebigen AWS Dienst speichern, der integriert werden kann AWS AppConfig, wie unter beschrieben. Erstellen Sie ein frei formatiertes Konfigurationsprofil in AWS AppConfig

Ein Konfigurationsprofil kann auch optionale Validatoren enthalten, um sicherzustellen, dass Ihre Konfigurationsdaten syntaktisch und semantisch korrekt sind. AWS AppConfig führt eine Überprüfung mithilfe der Validatoren durch, wenn Sie eine Bereitstellung starten. Wenn Fehler festgestellt werden, kehrt die Bereitstellung zu den vorherigen Konfigurationsdaten zurück.

5. Stellen Sie die Konfigurationsdaten bereit

   Wenn Sie eine neue Bereitstellung erstellen, geben Sie Folgendes an:

   - Eine Anwendungs-ID
   - Eine Konfigurationsprofil-ID
   - Eine Konfigurationsversion
   - Eine Umgebungs-ID, in der Sie die Konfigurationsdaten bereitstellen möchten
   - Eine Bereitstellungsstrategie-ID, die definiert, wie schnell die Änderungen wirksam werden sollen

   AWS AppConfig Führt beim Aufrufen der StartDeploymentAPI-Aktion die folgenden Aufgaben aus:

   1. Ruft die Konfigurationsdaten mithilfe des Standort-URI im Konfigurationsprofil aus dem zugrunde liegenden Datenspeicher ab.
   2. Überprüft mithilfe der Validatoren, die Sie bei der Erstellung Ihres Konfigurationsprofils angegeben haben, dass die Konfigurationsdaten syntaktisch und semantisch korrekt sind.
   3. Speichert eine Kopie der Daten im Cache, sodass sie von Ihrer Anwendung abgerufen werden können. Diese zwischengespeicherte Kopie wird als bereitgestellte Daten bezeichnet.

6. Rufen Sie die Konfiguration ab

   Sie können den AWS AppConfig Agenten als lokalen Host konfigurieren und den Agenten AWS AppConfig nach Konfigurationsupdates fragen lassen. Der Agent ruft die Aktionen StartConfigurationSessionund die GetLatestConfigurationAPI-Aktionen auf und speichert Ihre Konfigurationsdaten lokal im Cache. Um die Daten abzurufen, sendet Ihre Anwendung

einen HTTP-Aufruf an den Localhost-Server. AWS AppConfig Der Agent unterstützt mehrere Anwendungsfälle, wie unter beschriebenWie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten.

Wenn AWS AppConfig Agent für Ihren Anwendungsfall nicht unterstützt wird, können Sie Ihre Anwendung so konfigurieren, dass sie AWS AppConfig nach Konfigurationsupdates fragt, indem Sie die GetLatestConfigurationAPI-Aktionen StartConfigurationSessionund direkt aufrufen.

Erste Schritte mit AWS AppConfig

Die folgenden Ressourcen können Ihnen helfen, direkt damit zu arbeiten AWS AppConfig.

Weitere AWS Videos finden Sie auf dem Amazon Web Services YouTube Services-Kanal.

Die folgenden Blogs können Ihnen helfen, mehr über AWS AppConfig und die Funktionen zu erfahren:

- AWS AppConfig Feature-Flags verwenden
- Bewährte Methoden für die Validierung von AWS AppConfig Feature-Flags und Konfigurationsdaten

SDKs

Informationen zu AWS AppConfig sprachspezifischen SDKs Inhalten finden Sie in den folgenden Ressourcen:

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

Preisgestaltung für AWS AppConfig

Die Preisgestaltung für AWS AppConfig pay-as-you-go basiert auf den Konfigurationsdaten und dem Abrufen von Feature-Flags. Wir empfehlen, den AWS AppConfig Agenten zu verwenden, um die Kosten zu optimieren. Weitere Informationen finden Sie unter AWS Systems Manager - Preisgestaltung.

AWS AppConfig Kontingente

Informationen zu AWS AppConfig Endpunkten und Servicekontingenten sowie zu anderen Systems Manager Manager-Kontingenten finden Sie in der Allgemeine Amazon Web Services-Referenz.

> **Note**
>
> Informationen zu Kontingenten für Dienste, die AWS AppConfig Konfigurationen speichern, finden Sie unterGrundlegendes zu Kontingenten und Einschränkungen des Konfigurationsspeichers.

Einrichten AWS AppConfig

Falls Sie dies noch nicht getan haben, registrieren Sie sich für einen AWS-Konto und erstellen Sie einen Administratorbenutzer.

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Ein Teil des Anmeldevorgangs umfasst den Empfang eines Telefonanrufs oder einer Textnachricht und die Eingabe eines Bestätigungscodes auf der Telefontastatur.

   Wenn Sie sich für eine anmelden AWS-Konto, wird eine Root-Benutzer des AWS-Kontoserstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können Ihre aktuellen Kontoaktivitäten jederzeit einsehen und Ihr Konto verwalten, indem Sie zu https://aws.amazon.com/gehen und Mein Konto auswählen.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerhandbuch.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des interagieren möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Ref erenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldeinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Unterstützung verstehen IPv6

Alle unterstützen IPv4 und IPv6 rufen AWS AppConfig APIs uneingeschränkt an.

Kontrollebene APIs

Verwenden Sie den folgenden Endpunkt für IPv4 und IPv6 Dual-Stack-Aufrufe an die Steuerungsebene:

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

Zum Beispiel: appconfig.us-east-1.api.aws

IPv4 Verwenden Sie nur die folgende URL:

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

Datenebene APIs

Verwenden Sie für Dual-Stack-Aufrufe an die Datenebene den folgenden Endpunkt:

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

Zum Beispiel: appconfig.us-east-1.api.aws

IPv4 Verwenden Sie nur die folgende URL:

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

> **Note**
>
> Weitere Informationen finden Sie unter AWS AppConfig -Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

(Empfohlen) Konfigurieren Sie die Berechtigungen für das automatische Rollback

Sie können so konfigurieren AWS AppConfig , dass Sie als Reaktion auf einen oder mehrere CloudWatch Amazon-Alarme zu einer früheren Version einer Konfiguration zurückkehren. Wenn Sie eine Bereitstellung so konfigurieren, dass sie auf CloudWatch Alarme reagiert, geben Sie eine AWS Identity and Access Management (IAM-) Rolle an. AWS AppConfig benötigt diese Rolle, um

CloudWatch Alarme überwachen zu können. Dieses Verfahren ist optional, wird aber dringend empfohlen.

> **Note**
>
> Notieren Sie die folgenden Informationen:
>
> - Die IAM-Rolle muss zum aktuellen Konto gehören. Standardmäßig AWS AppConfig können nur Alarme überwacht werden, die dem aktuellen Konto gehören.
> - Informationen zu den zu überwachenden Metriken und AWS AppConfig zur Konfiguration des automatischen Rollbacks finden Sie unterBereitstellungen im Hinblick auf automatisches Rollback überwachen.

Verwenden Sie die folgenden Verfahren, um eine IAM-Rolle zu erstellen, die ein Rollback auf der Grundlage von Alarmen ermöglicht AWS AppConfig . CloudWatch In diesem Abschnitt werden folgende Verfahren beschrieben.

1. Schritt 1: Erstellen Sie die Berechtigungsrichtlinie für ein Rollback auf der Grundlage von Alarmen CloudWatch
2. Schritt 2: Erstellen Sie die IAM-Rolle für Rollback auf der Grundlage von Alarmen CloudWatch
3. Schritt 3: Hinzufügen einer Vertrauensstellung

Schritt 1: Erstellen Sie die Berechtigungsrichtlinie für ein Rollback auf der Grundlage von Alarmen CloudWatch

Gehen Sie wie folgt vor, um eine IAM-Richtlinie zu erstellen, die zum Aufrufen der `DescribeAlarms` API-Aktion AWS AppConfig berechtigt.

So erstellen Sie eine IAM-Berechtigungsrichtlinie für Rollback auf der Grundlage von Alarmen CloudWatch

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Richtlinien und dann Richtlinie erstellen.
3. Wählen Sie auf der Seite Richtlinie erstellen die Registerkarte JSON aus.
4. Ersetzen Sie den Standardinhalt auf der Registerkarte JSON durch die folgende Berechtigungsrichtlinie und wählen Sie dann Weiter: Tags aus.

> **Note**
>
> Um Informationen über CloudWatch zusammengesetzte Alarme zurückzugeben, müssen dem DescribeAlarmsAPI-Vorgang * Berechtigungen zugewiesen werden, wie hier gezeigt. Sie können keine Informationen über zusammengesetzte Alarme zurückgeben, wenn der Anwendungsbereich enger `DescribeAlarms` ist.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:DescribeAlarms"
                ],
                "Resource": "*"
            }
        ]
    }
```

5. Geben Sie Tags für diese Rolle ein und wählen Sie dann Next: Review (Weiter: Prüfen) aus.
6. Geben Sie auf der Seite „Überprüfung“ **`SSMCloudWatchAlarmDiscoveryPolicy`** in das Feld „Name“ ein.
7. Wählen Sie Richtlinie erstellen aus. Das System führt Sie zur Seite Policies (Richtlinien) zurück.

Schritt 2: Erstellen Sie die IAM-Rolle für Rollback auf der Grundlage von Alarmen CloudWatch

Gehen Sie wie folgt vor, um eine IAM-Rolle zu erstellen und ihr die Richtlinie zuzuweisen, die Sie im vorherigen Verfahren erstellt haben.

So erstellen Sie eine IAM-Rolle für Rollback auf der Grundlage von Alarmen CloudWatch

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und dann Create role (Rolle erstellen).

3. Wählen Sie unter Select type of trusted entity (Typ der vertrauenswürdigen Entität auswählen) die Option AWS -Service aus.
4. Wählen Sie unmittelbar unter Wählen Sie den Dienst aus, der diese Rolle verwenden soll die Option EC2: Erlaubt EC2 Instances, AWS Dienste in Ihrem Namen aufzurufen, und klicken Sie dann auf Weiter: Berechtigungen.
5. Suchen Sie auf der Seite mit den Richtlinien für angehängte Berechtigungen nach SSMCloudWatchAlarmDiscoveryPolicy.
6. Wählen Sie diese Richtlinie aus, und klicken Sie dann aufNext: Tags (Weiter: Tags).
7. Geben Sie Tags für diese Rolle ein und wählen Sie dann Next: Review (Weiter: Prüfen) aus.
8. Geben Sie auf der Seite Rolle erstellen **`SSMCloudWatchAlarmDiscoveryRole`** in das Feld Rollenname ein und wählen Sie dann Rolle erstellen aus.
9. Wählen Sie auf der Seite Roles (Rollen) die von Ihnen soeben erstellte Rolle aus. Die Seite Summary (Übersicht) wird geöffnet.

Schritt 3: Hinzufügen einer Vertrauensstellung

Gehen Sie wie folgt vor, um die Rolle, die Sie gerade erstellt haben, für AWS AppConfig als Vertrauensstellung zu konfigurieren.

Um eine Vertrauensbeziehung hinzuzufügen für AWS AppConfig

1. Wählen Sie auf der Seite Summary für die eben erstellte Rolle die Registerkarte Trust Relationships und wählen Sie dann Edit Trust Relationship.
2. Bearbeiten Sie die Richtlinie so, dass sie nur "`appconfig.amazonaws.com`" enthält, wie im folgenden Beispiel gezeigt:

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

```
}
```

3. Wählen Sie Update Trust Policy (Trust Policy aktualisieren).

Erstellen von Feature-Flags und Freiform-Konfigurationsdaten in AWS AppConfig

Die Themen in diesem Abschnitt helfen Ihnen bei der Ausführung der folgenden Aufgaben in AWS AppConfig. Diese Aufgaben erzeugen wichtige Artefakte für die Bereitstellung von Konfigurationsdaten.

1. Erstellen Sie einen Anwendungs-Namespace

   Um einen Anwendungsnamespace zu erstellen, erstellen Sie ein AWS AppConfig Artefakt, das als Anwendung bezeichnet wird. Eine Anwendung ist einfach ein organisatorisches Konstrukt wie ein Ordner.

2. Umgebungen erstellen

   Für jede AWS AppConfig Anwendung definieren Sie eine oder mehrere Umgebungen. Eine Umgebung ist eine logische Bereitstellungsgruppe von AWS AppConfig Zielen, z. B. Anwendungen in einer `Beta` `Production` Oder-Umgebung. Sie können auch Umgebungen für Anwendungsunterkomponenten wie,`AWS Lambda functions`, `Containers` `WebMobile`, und `Back-end` definieren.

   Sie können CloudWatch Amazon-Alarme für jede Umgebung so konfigurieren, dass problematische Konfigurationsänderungen automatisch rückgängig gemacht werden. Das System überwacht Alarme während einer Konfigurationsbereitstellung. Wenn ein Alarm ausgelöst wird, setzt das System die Konfiguration zurück.

3. Erstellen Sie ein Konfigurationsprofil

   Bei Konfigurationsdaten handelt es sich um eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Ein Konfigurationsprofil umfasst unter anderem eine URI, die es ermöglicht, Ihre Konfigurationsdaten AWS AppConfig an ihrem gespeicherten Speicherort zu finden, und einen Konfigurationstyp. AWS AppConfig unterstützt die folgenden Arten von Konfigurationsprofilen:

   - Feature-Flags: Sie können Feature-Flags verwenden, um Funktionen in Ihren Anwendungen zu aktivieren oder zu deaktivieren oder um verschiedene Eigenschaften Ihrer Anwendungsfunktionen mithilfe von Flag-Attributen zu konfigurieren. AWS AppConfig speichert Feature-Flag-Konfigurationen im AWS AppConfig gehosteten Konfigurationsspeicher in einem Feature-Flag-Format, das Daten und Metadaten zu Ihren Flags und den Flag-Attributen enthält. Der URI für Feature-Flag-Konfigurationen ist einfach`hosted`.

- Freiformkonfigurationen: Eine Freiformkonfiguration kann Daten in einem der folgenden Tools AWS-Services und Systems Manager Manager-Tools speichern:
  - AWS AppConfig gehosteter Konfigurationsspeicher
  - Amazon Simple Storage Service
  - AWS CodePipeline
  - AWS Secrets Manager
  - AWS Systems Manager (SSM) Parameterspeicher
  - SSM-Dokumentenspeicher

> **Note**
>
> Wenn möglich, empfehlen wir, Ihre Konfigurationsdaten im AWS AppConfig gehosteten Konfigurationsspeicher zu hosten, da dieser die meisten Funktionen und Verbesserungen bietet.

4. (Optional, aber empfohlen) Erstellen Sie Feature-Flags mit mehreren Varianten

   AWS AppConfig bietet grundlegende Feature-Flags, die (falls aktiviert) pro Anfrage einen bestimmten Satz von Konfigurationsdaten zurückgeben. Um Anwendungsfälle wie Benutzersegmentierung und Traffic-Aufteilung besser zu unterstützen, bietet es AWS AppConfig auch mehrere Varianten von Feature-Flags, mit denen Sie einen Satz möglicher Flag-Werte definieren können, die bei einer Anfrage zurückgegeben werden sollen. Sie können auch verschiedene Status (aktiviert oder deaktiviert) für Flags mit mehreren Varianten konfigurieren. Wenn Sie ein mit Varianten konfiguriertes Kennzeichen anfordern, stellt Ihre Anwendung einen Kontext bereit, der anhand einer Reihe von benutzerdefinierten Regeln AWS AppConfig ausgewertet wird. Abhängig vom in der Anfrage angegebenen Kontext und den für die Variante definierten Regeln werden unterschiedliche Flagwerte an die Anwendung AWS AppConfig zurückgegeben.

Themen

- Grundlegendes zur IAM-Rolle des Konfigurationsprofils
- Erstellen Sie einen Namespace für Ihre Anwendung in AWS AppConfig
- Umgebungen für Ihre Anwendung erstellen in AWS AppConfig
- Erstellen eines Konfigurationsprofils in AWS AppConfig

Grundlegendes zur IAM-Rolle des Konfigurationsprofils

Sie können die IAM-Rolle, die den Zugriff auf die Konfigurationsdaten ermöglicht, mithilfe von erstellen. AWS AppConfig Sie können die IAM-Rolle auch selbst erstellen. Wenn Sie die Rolle mithilfe von erstellen AWS AppConfig, erstellt das System die Rolle und gibt eine der folgenden Berechtigungsrichtlinien an, je nachdem, welche Art von Konfigurationsquelle Sie wählen.

Die Konfigurationsquelle ist ein Secrets Manager Manager-Geheimnis

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

Die Konfigurationsquelle ist ein Parameter Store-Parameter

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

Wenn die Konfigurationsquelle ein SSM-Dokument ist:

```
{

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

Wenn Sie die Rolle mithilfe von erstellen AWS AppConfig, erstellt das System auch die folgende Vertrauensstellung für die Rolle.

```
{

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

Erstellen Sie einen Namespace für Ihre Anwendung in AWS AppConfig

Die Verfahren in diesem Abschnitt helfen Ihnen beim Erstellen eines AWS AppConfig Artefakts, das als Anwendung bezeichnet wird. Eine Anwendung ist einfach ein Organisationskonstrukt wie ein Ordner, der den Namespace Ihrer Anwendung identifiziert. Dieses Organisations-Konstrukt

hat eine Beziehung zu einer Einheit von ausführbarem Code. Sie könnten beispielsweise eine Anwendung erstellen, die aufgerufen wird, MyMobileApp um Konfigurationsdaten für eine von Ihren Benutzern installierte mobile Anwendung zu organisieren und zu verwalten. Sie müssen diese Artefakte erstellen, bevor Sie sie AWS AppConfig zum Bereitstellen und Abrufen von Feature-Flags oder Freiform-Konfigurationsdaten verwenden können.

Das folgende Verfahren bietet Ihnen die Möglichkeit, eine Erweiterung einem Feature-Flag-Konfigurationsprofil zuzuordnen. Eine Erweiterung erweitert Ihre Fähigkeit, Logik oder Verhalten an verschiedenen Stellen während des AWS AppConfig Workflows zum Erstellen oder Bereitstellen einer Konfiguration einzufügen. Weitere Informationen finden Sie unter AWS AppConfig Erweiterungen verstehen.

> **Note**
>
> Sie können AWS CloudFormation sie verwenden, um AWS AppConfig Artefakte zu erstellen, darunter Anwendungen, Umgebungen, Konfigurationsprofile, Bereitstellungen, Bereitstellungsstrategien und gehostete Konfigurationsversionen. Weitere Informationen finden Sie unter AWS AppConfig Ressourcentyp-Referenz im AWS CloudFormation - Benutzerhandbuch.

Themen

- Eine AWS AppConfig Anwendung (Konsole) erstellen
- Eine AWS AppConfig Anwendung erstellen (Befehlszeile)

Eine AWS AppConfig Anwendung (Konsole) erstellen

Gehen Sie wie folgt vor, um eine AWS AppConfig Anwendung mithilfe der AWS Systems Manager Konsole zu erstellen.

So erstellen Sie eine Anwendung

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Applications (Anwendungen) und anschließend Create a new application (Neue Anwendung erstellen).
3. Geben Sie unter Name einen Namen für die Anwendung ein.

4. Geben Sie unter Description (Beschreibung) Informationen zur Anwendung ein.
5. (Optional) Wählen Sie im Abschnitt Erweiterungen eine Erweiterung aus der Liste aus. Weitere Informationen finden Sie unter AWS AppConfig Erweiterungen verstehen.
6. (Optional) Geben Sie im Abschnitt „Tags“ einen Schlüssel und einen optionalen Wert ein. Sie können maximal 50 Tags für eine Ressource angeben.
7. Wählen Sie Create application aus.

AWS AppConfig erstellt die Anwendung und zeigt dann die Registerkarte Umgebungen an. Fahren Sie mit Umgebungen für Ihre Anwendung erstellen in AWS AppConfig fort.

Eine AWS AppConfig Anwendung erstellen (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie die AWS CLI (unter Linux oder Windows) verwenden oder AWS -Tools für PowerShell eine AWS AppConfig Anwendung erstellen.

So erstellen Sie Schritt für Schritt eine Anwendung

1. Öffne das AWS CLI.
2. Führen Sie den folgenden Befehl aus, um eine Anwendung zu erstellen.

   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
```

Das System gibt unter anderem folgende Informationen zurück

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
```

Umgebungen für Ihre Anwendung erstellen in AWS AppConfig

Für jede AWS AppConfig Anwendung definieren Sie eine oder mehrere Umgebungen. Eine Umgebung ist eine logische Bereitstellungsgruppe von AppConfig Zielen, wie z. B. Anwendungen in einer `Beta` `Production` Oder-Umgebung, AWS Lambda Funktionen oder Container. Sie können auch Umgebungen für Anwendungsunterkomponenten wie `Web`Mobile, und `Back-end` definieren. Sie können CloudWatch Amazon-Alarme für jede Umgebung konfigurieren. Das System überwacht Alarme während einer Konfigurationsbereitstellung. Wenn ein Alarm ausgelöst wird, setzt das System die Konfiguration zurück.

Bevor Sie beginnen

Wenn Sie das Rollback einer Konfiguration als Reaktion auf einen CloudWatch Alarm aktivieren AWS AppConfig möchten, müssen Sie eine AWS Identity and Access Management (IAM-) Rolle mit Berechtigungen konfigurieren, um auf CloudWatch Alarme reagieren AWS AppConfig zu können. Diese Rolle wählen Sie im folgenden Verfahren aus. Weitere Informationen finden Sie unter (Empfohlen) Konfigurieren Sie die Berechtigungen für das automatische Rollback.

Themen

- Eine AWS AppConfig Umgebung (Konsole) erstellen
- Eine AWS AppConfig Umgebung erstellen (Befehlszeile)

Eine AWS AppConfig Umgebung (Konsole) erstellen

Gehen Sie wie folgt vor, um mithilfe der AWS Systems Manager Konsole eine AWS AppConfig Umgebung zu erstellen.

So erstellen Sie eine Umgebung

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Applications und anschließend den Namen einer Anwendung aus, um die Detailseite zu öffnen.
3. Wählen Sie die Registerkarte Umgebungen und dann Umgebung erstellen aus.
4. Geben Sie unter Name einen Namen für die Umgebung ein.
5. Geben Sie unter Description (Beschreibung) Informationen zur Umgebung ein.
6. (Optional) Wählen Sie im Abschnitt Monitore das Feld IAM-Rolle und dann eine IAM-Rolle mit der Berechtigung aus, die Metriken abzurufen, die Sie `cloudwatch:DescribeAlarms` auf Alarme überwachen möchten.
7. Geben Sie in der CloudWatch Alarmliste die Amazon-Ressourcennamen (ARNs) ein oder mehrere zu überwachende Metriken ein. AWS AppConfig macht Ihre Konfigurationsbereitstellung rückgängig, wenn eine dieser Metriken einen `ALARM` Status annimmt. Informationen zu empfohlenen Metriken finden Sie unter Bereitstellungen im Hinblick auf automatisches Rollback überwachen
8. (Optional) Wählen Sie im Abschnitt Associate extensions eine Erweiterung aus der Liste aus. Weitere Informationen finden Sie unter AWS AppConfig Erweiterungen verstehen.

9. (Optional) Geben Sie im Abschnitt Tags einen Schlüssel und einen optionalen Wert ein. Sie können maximal 50 Tags für eine Ressource angeben.
10. Wählen Sie Create environment (Umgebung erstellen) aus.

AWS AppConfig erstellt die Umgebung und zeigt dann die Seite mit den Umgebungsdetails an. Fahren Sie mit Erstellen eines Konfigurationsprofils in AWS AppConfig fort.

Eine AWS AppConfig Umgebung erstellen (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie die AWS CLI (unter Linux oder Windows) verwenden oder AWS -Tools für PowerShell eine AWS AppConfig Umgebung erstellen.

Um Schritt für Schritt eine Umgebung zu erstellen

1. Öffne das AWS CLI.
2. Führen Sie den folgenden Befehl aus, um eine Umgebung zu erstellen.

   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
```

Das System gibt unter anderem folgende Informationen zurück

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
```

Fahren Sie mit Erstellen eines Konfigurationsprofils in AWS AppConfig fort.

Erstellen eines Konfigurationsprofils in AWS AppConfig

Bei Konfigurationsdaten handelt es sich um eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Ein Konfigurationsprofil umfasst unter anderem eine URI, die es ermöglicht, Ihre Konfigurationsdaten AWS AppConfig an ihrem gespeicherten Speicherort zu finden, und einen Konfigurationstyp. AWS AppConfig unterstützt die folgenden Arten von Konfigurationsprofilen:

- Feature-Flags: Sie können Feature-Flags verwenden, um Funktionen in Ihren Anwendungen zu aktivieren oder zu deaktivieren oder um verschiedene Eigenschaften Ihrer Anwendungsfunktionen mithilfe von Flag-Attributen zu konfigurieren. AWS AppConfig speichert Feature-Flag-Konfigurationen im AWS AppConfig gehosteten Konfigurationsspeicher in einem Feature-Flag-Format, das Daten und Metadaten zu Ihren Flags und den Flag-Attributen enthält. Der URI für Feature-Flag-Konfigurationen ist einfach`hosted`.
- Freiformkonfigurationen: Eine Freiformkonfiguration kann Daten in einem der folgenden Tools AWS-Services und Systems Manager Manager-Tools speichern:
  - AWS AppConfig gehosteter Konfigurationsspeicher

- Amazon Simple Storage Service
- AWS CodePipeline
- AWS Secrets Manager
- AWS Systems Manager (SSM) Parameterspeicher
- SSM-Dokumentenspeicher

> **Note**
>
> Wenn möglich, empfehlen wir, Ihre Konfigurationsdaten im AWS AppConfig gehosteten Konfigurationsspeicher zu hosten, da dieser die meisten Funktionen und Verbesserungen bietet.

Im Folgenden finden Sie einige Beispiele für Konfigurationsdaten, die Ihnen helfen sollen, die verschiedenen Typen von Konfigurationsdaten besser zu verstehen und zu erfahren, wie sie entweder in einem Feature-Flag oder in einem Profil ohne Konfiguration verwendet werden können.

Konfigurationsdaten für Feature-Flags

Die folgenden Konfigurationsdaten für Feature-Flags aktivieren oder deaktivieren mobile Zahlungen und Standardzahlungen pro Region.

JSON

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

YAML

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

```
  enabled: true
```

Daten zur Betriebskonfiguration

Die folgenden Freiform-Konfigurationsdaten beschränken die Art und Weise, wie eine Anwendung Anfragen verarbeitet.

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
```

Konfigurationsdaten der Zugriffskontrollliste

Die folgenden Freiform-Konfigurationsdaten für die Zugriffskontrollliste geben an, welche Benutzer oder Gruppen auf eine Anwendung zugreifen können.

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
```

Themen

- Erstellen eines Feature-Flag-Konfigurationsprofils in AWS AppConfig
- Erstellen Sie ein frei formatiertes Konfigurationsprofil in AWS AppConfig
- Erstellen eines Konfigurationsprofils für nicht-native Datenquellen

Erstellen eines Feature-Flag-Konfigurationsprofils in AWS AppConfig

Sie können Feature-Flags verwenden, um Funktionen in Ihren Anwendungen zu aktivieren oder zu deaktivieren oder um verschiedene Eigenschaften Ihrer Anwendungsfunktionen mithilfe von Flag-Attributen zu konfigurieren. AWS AppConfig speichert Feature-Flag-Konfigurationen im AWS AppConfig gehosteten Konfigurationsspeicher in einem Feature-Flag-Format, das Daten und Metadaten zu Ihren Flags und den Flag-Attributen enthält.

> **Note**
>
> Wenn Sie ein Feature-Flag-Konfigurationsprofil erstellen, können Sie im Rahmen des Konfigurationsprofil-Workflows ein grundlegendes Feature-Flag erstellen. AWS AppConfig unterstützt auch Feature-Flags mit mehreren Varianten. Feature-Flags mit mehreren Varianten ermöglichen es Ihnen, eine Reihe möglicher Flag-Werte zu definieren, die für eine Anfrage zurückgegeben werden sollen. Wenn Sie ein mit Varianten konfiguriertes Kennzeichen anfordern, stellt Ihre Anwendung einen Kontext bereit, der anhand einer Reihe von benutzerdefinierten Regeln AWS AppConfig ausgewertet wird. Abhängig vom in der Anfrage angegebenen Kontext und den für die Variante definierten Regeln werden unterschiedliche Flagwerte an die Anwendung AWS AppConfig zurückgegeben.
> Um Feature-Flags mit mehreren Varianten zu erstellen, erstellen Sie zuerst ein Konfigurationsprofil und bearbeiten Sie dann alle Flags innerhalb des Konfigurationsprofils, um Varianten hinzuzufügen. Weitere Informationen finden Sie unter Feature-Flags mit mehreren Varianten erstellen.

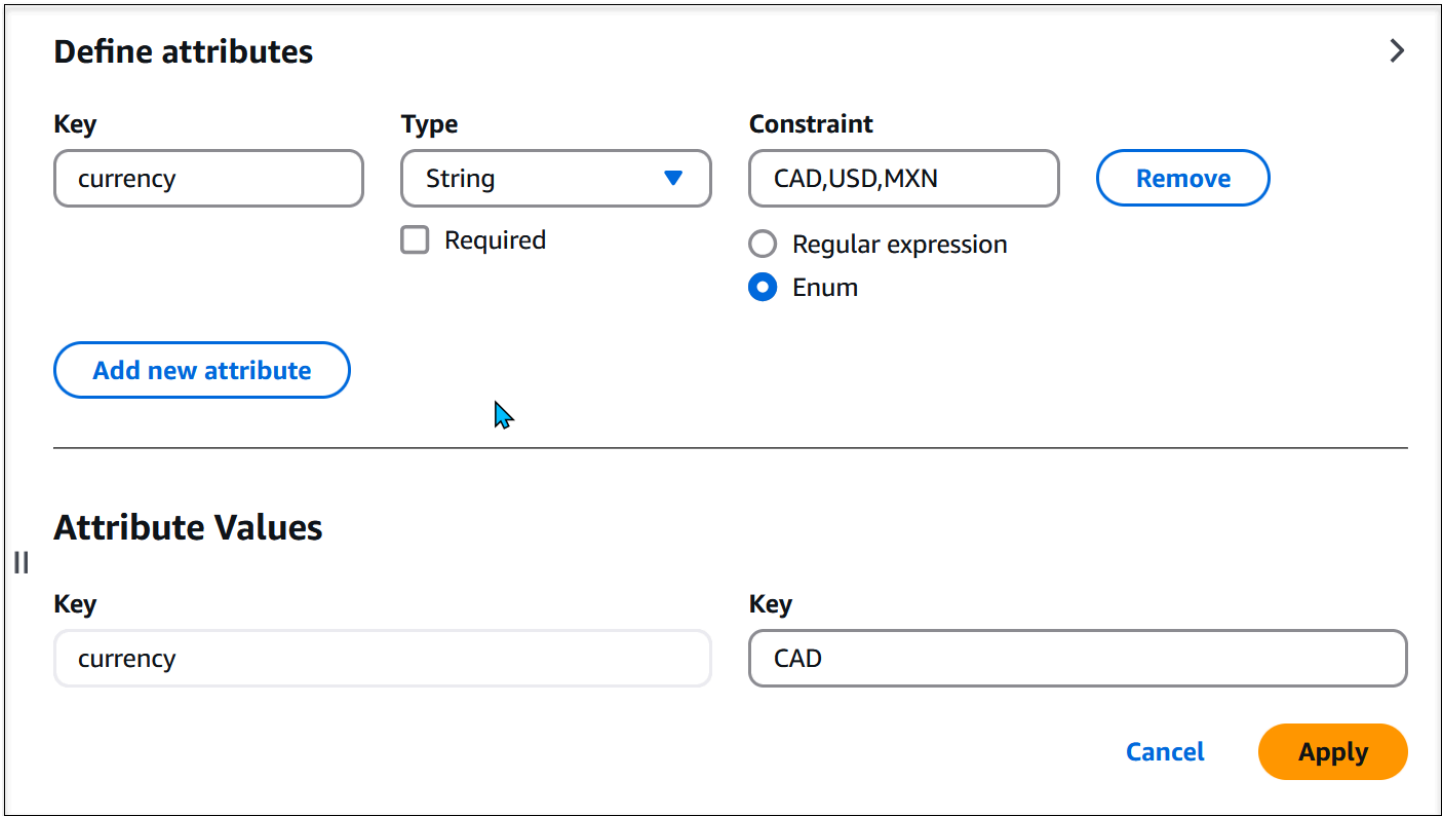
Themen

- Die Attribute von Feature-Flags verstehen
- Erstellen eines Feature-Flag-Konfigurationsprofils (Konsole)
- Erstellen eines Feature-Flag-Konfigurationsprofils (Befehlszeile)
- Feature-Flags mit mehreren Varianten erstellen
- Grundlegendes zur Typenreferenz für AWS.AppConfig.FeatureFlags
- Speichern einer früheren Feature-Flag-Version in einer neuen Version

Die Attribute von Feature-Flags verstehen

Wenn Sie ein Feature-Flag-Konfigurationsprofil oder ein neues Flag in einem vorhandenen Konfigurationsprofil erstellen, können Sie Attribute und entsprechende Einschränkungen für das Flag angeben. Ein Attribut ist ein Feld, das Sie Ihrem Feature-Flag zuordnen, um Eigenschaften auszudrücken, die sich auf Ihr Feature-Flag beziehen. Attribute werden zusammen mit Ihrem Flaggenschlüssel und dem `disable` Wert `enable` oder der Markierung an Ihre Anwendung übermittelt.

Einschränkungen stellen sicher, dass keine unerwarteten Attributwerte in Ihrer Anwendung bereitgestellt werden. In der folgenden Abbildung sehen Sie ein Beispiel.



> **Note**
>
> Beachten Sie die folgenden Informationen zu Flaggenattributen.
>
> - Für Attributnamen ist das Wort „aktiviert“ reserviert. Sie können kein Feature-Flag-Attribut mit dem Namen „aktiviert“ erstellen. Es gibt keine anderen reservierten Wörter.
> - Die Attribute eines Feature-Flags sind nur dann in der `GetLatestConfiguration` Antwort enthalten, wenn dieses Flag aktiviert ist.

- Flaggenattributschlüssel für eine bestimmte Flagge müssen eindeutig sein.

AWS AppConfig unterstützt die folgenden Typen von Flaggenattributen und die entsprechenden Einschränkungen.

Typ	Constraint	Beschreibung
Zeichenfolge	Regulärer Ausdruck	Regex-Muster für die Zeichenfolge
	Enum	Liste der akzeptablen Werte für die Zeichenfolge
Zahl	Minimum	Numerischer Mindestwert für das Attribut
	Maximum	Maximaler numerischer Wert für das Attribut
Boolesch	Keine	Keine
Zeichenketten-Array	Regulärer Ausdruck	Regex-Muster für die Elemente des Arrays
	Enum	Liste der akzeptablen Werte für die Elemente des Arrays
Zahlenarray	Minimum	Numerischer Mindestwert für die Elemente des Arrays
	Maximum	Maximaler numerischer Wert für die Elemente des Arrays

Erstellen eines Feature-Flag-Konfigurationsprofils (Konsole)

Gehen Sie wie folgt vor, um mithilfe der AWS AppConfig Konsole ein AWS AppConfig Feature-Flag-Konfigurationsprofil zu erstellen. Wenn Sie das Konfigurationsprofil erstellen, können Sie auch ein grundlegendes Feature-Flag erstellen.

So erstellen Sie ein Konfigurationsprofil

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Anwendungen und dann eine Anwendung aus, in der Sie sie erstellt haben. Erstellen Sie einen Namespace für Ihre Anwendung in AWS AppConfig
3. Wählen Sie auf der Registerkarte Konfigurationsprofile und Feature-Flags die Option Konfiguration erstellen aus.
4. Wählen Sie im Abschnitt Konfigurationsoptionen die Option Feature-Flag aus.
5. Geben Sie im Abschnitt Konfigurationsprofil für den Namen des Konfigurationsprofils einen Namen ein.
6. (Optional) Erweitern Sie Beschreibung und geben Sie eine Beschreibung ein.
7. (Optional) Erweitern Sie Zusätzliche Optionen und füllen Sie bei Bedarf die folgenden Schritte aus.

   a. Wählen Sie in der Verschlüsselungsliste einen AWS Key Management Service (AWS KMS) Schlüssel aus der Liste aus. Mit diesem vom Kunden verwalteten Schlüssel können Sie neue Versionen von Konfigurationsdaten im AWS AppConfig gehosteten Konfigurationsspeicher verschlüsseln. Weitere Informationen zu diesem Schlüssel finden Sie unter AWS AppConfig Unterstützt Schlüssel für Kundenmanager inSicherheit in AWS AppConfig.

   b. Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus und geben Sie dann einen Schlüssel und einen optionalen Wert an.
8. Wählen Sie Weiter aus.
9. Geben Sie im Abschnitt Feature-Flag-Definition für Flagname einen Namen ein.
10. Geben Sie unter Flag-Schlüssel eine Flag-ID ein, um Flags innerhalb desselben Konfigurationsprofils voneinander zu unterscheiden. Flags innerhalb desselben Konfigurationsprofils können nicht denselben Schlüssel haben. Nachdem das Flag erstellt wurde, können Sie den Flag-Namen bearbeiten, aber nicht den Flaggenschlüssel.
11. (Optional) Erweitern Sie Beschreibung und geben Sie Informationen zu dieser Flagge ein.
12. Wählen Sie „Dies ist eine kurzfristige Kennzeichnung“ und wählen Sie optional ein Datum aus, an dem die Kennzeichnung deaktiviert oder gelöscht werden soll. AWS AppConfig deaktiviert die Markierung am Verfallsdatum nicht.
13. (Optional) Wählen Sie im Abschnitt Feature-Flag-Attribute die Option Attribut definieren aus. Mithilfe von Attributen können Sie zusätzliche Werte in Ihrer Flagge angeben. Weitere

Informationen zu Attributen und Einschränkungen finden Sie unterDie Attribute von Feature-Flags verstehen.

a. Geben Sie für Schlüssel einen Flaggenschlüssel an und wählen Sie seinen Typ aus der Liste Typ aus. Informationen zu den unterstützten Optionen für die Felder Wert und Einschränkungen finden Sie im Abschnitt über Attribute, auf den zuvor verwiesen wurde.

b. Wählen Sie Erforderlicher Wert aus, um anzugeben, ob ein Attributwert erforderlich ist.

c. Wählen Sie „Attribut definieren“, um weitere Attribute hinzuzufügen.

14. Wählen Sie im Abschnitt Feature-Flag-Wert die Option Aktiviert aus, um die Markierung zu aktivieren. Verwenden Sie denselben Schalter, um eine Markierung zu deaktivieren, wenn sie ein bestimmtes Verfallsdatum erreicht, falls zutreffend.
15. Wählen Sie Weiter aus.
16. Überprüfen Sie auf der Seite Überprüfen und speichern die Details der Markierung und klicken Sie dann auf Speichern und mit der Bereitstellung fortfahren.

Fahren Sie mit Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig fort.

Erstellen eines Feature-Flag-Konfigurationsprofils (Befehlszeile)

Im folgenden Verfahren wird beschrieben, wie Sie mithilfe von AWS Command Line Interface (unter Linux oder Windows) oder Tools für Windows PowerShell ein AWS AppConfig Feature-Flag-Konfigurationsprofil erstellen. Wenn Sie das Konfigurationsprofil erstellen, können Sie auch ein grundlegendes Feature-Flag erstellen.

Um eine Feature-Flag-Konfiguration zu erstellen

1. Öffnen Sie das AWS CLI.
2. Erstellen Sie ein Konfigurationsprofil für Feature-Flags und geben Sie dessen Typ als an`AWS.AppConfig.FeatureFlags`. Das Konfigurationsprofil muss den URI `hosted` für den Standort verwenden.

   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. Erstellen Sie Ihre Feature-Flag-Konfigurationsdaten. Ihre Daten müssen in einem JSON-Format vorliegen und dem `AWS.AppConfig.FeatureFlags` JSON-Schema entsprechen. Weitere Informationen zum Schema finden Sie unterGrundlegendes zur Typenreferenz für AWS.AppConfig.FeatureFlags.
4. Verwenden Sie die `CreateHostedConfigurationVersion` API, um Ihre Feature-Flag-Konfigurationsdaten zu speichern 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
```

Der Befehl lädt den für den `Content` Parameter angegebenen Inhalt von der Festplatte. Der Inhalt muss dem folgenden Beispiel ähneln.

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

Das System gibt unter anderem folgende Informationen zurück

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
```

Das `service_returned_content_file` enthält Ihre Konfigurationsdaten, die einige AWS AppConfig generierte Metadaten enthalten.

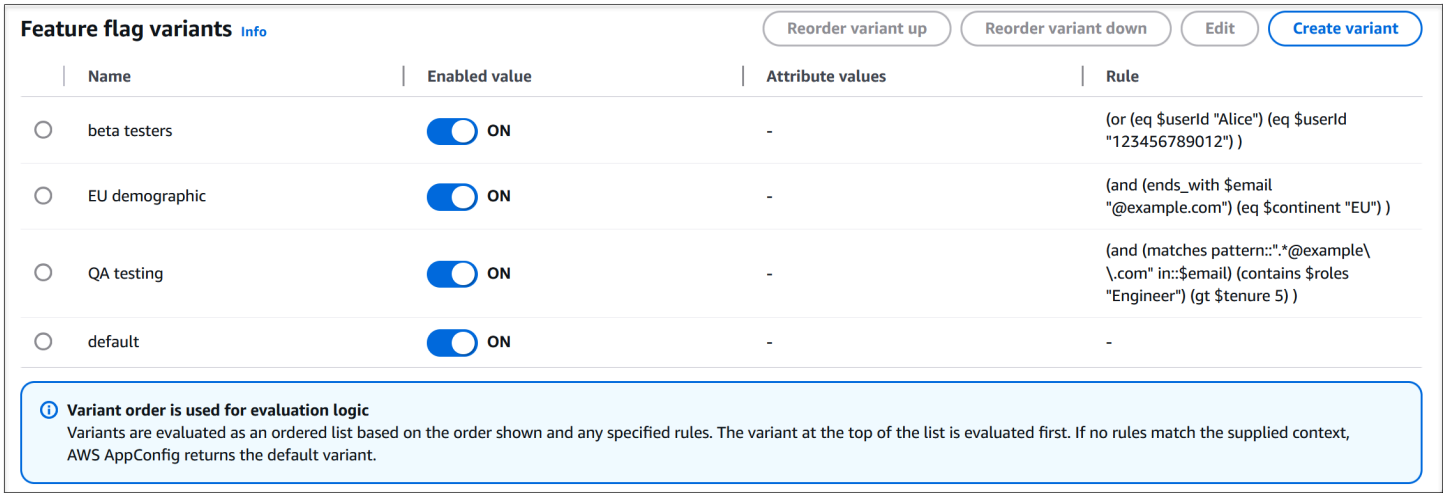
> **Note**
>
> Wenn Sie die gehostete Konfigurationsversion erstellen, wird AWS AppConfig überprüft, ob Ihre Daten dem `AWS.AppConfig.FeatureFlags` JSON-Schema entsprechen. AWS AppConfig überprüft außerdem, ob jedes Feature-Flag-Attribut in Ihren Daten die Einschränkungen erfüllt, die Sie für diese Attribute definiert haben.

Feature-Flags mit mehreren Varianten erstellen

Feature-Flag-Varianten ermöglichen es Ihnen, eine Reihe möglicher Flag-Werte zu definieren, die für eine Anfrage zurückgegeben werden sollen. Sie können auch verschiedene Status (aktiviert oder deaktiviert) für Flags mit mehreren Varianten konfigurieren. Wenn Sie ein mit Varianten konfiguriertes Kennzeichen anfordern, stellt Ihre Anwendung einen Kontext bereit, der anhand einer Reihe von benutzerdefinierten Regeln AWS AppConfig ausgewertet wird. Abhängig vom in der Anfrage

angegebenen Kontext und den für die Variante definierten Regeln werden unterschiedliche Flagwerte an die Anwendung AWS AppConfig zurückgegeben.

Der folgende Screenshot zeigt ein Beispiel für ein Feature-Flag mit drei benutzerdefinierten Varianten und der Standardvariante.



Themen

- Grundlegendes zu Konzepten von Feature-Flags mit mehreren Varianten und zu häufigen Anwendungsfällen
- Grundlegendes zu Feature-Flag-Regeln mit mehreren Varianten
- Erstellen eines Feature-Flags mit mehreren Varianten

Grundlegendes zu Konzepten von Feature-Flags mit mehreren Varianten und zu häufigen Anwendungsfällen

Um Ihnen ein besseres Verständnis von Feature-Flag-Varianten zu vermitteln, werden in diesem Abschnitt die Konzepte von Flag-Varianten und allgemeine Anwendungsfälle erläutert.

Konzepte

- Feature-Flag: Ein AWS AppConfig Konfigurationstyp, der verwendet wird, um das Verhalten einer Funktion in einer Anwendung zu steuern. Ein Flag hat einen Status (aktiviert oder deaktiviert) und einen optionalen Satz von Attributen, die beliebige Zeichenketten-, numerische, boolesche oder Array-Werte enthalten.
- Feature-Flag-Variante: Eine spezifische Kombination von Status- und Attributwerten, die zu einem Feature-Flag gehören. Ein Feature-Flag kann mehrere Varianten haben.

- Variantenregel: Ein benutzerdefinierter Ausdruck, der zur Auswahl einer Feature-Flag-Variante verwendet wird. Jede Variante hat ihre eigene Regel, die AWS AppConfig auswertet, um zu bestimmen, ob sie zurückgegeben werden soll oder nicht.
- Standardvariante: Eine spezielle Variante, die zurückgegeben wird, wenn keine andere Variante ausgewählt wurde. Alle Feature-Flags mit mehreren Varianten haben eine Standardvariante.

  Beachten Sie, dass die Standardvariante in Ihrer Reihenfolge der Varianten an letzter Stelle stehen muss und dass ihr keine Regeln zugeordnet sein dürfen. Wenn sie nicht zuletzt definiert wurde, wird a AWS AppConfig zurückgegeben, `BadRequestException` wenn Sie versuchen, das Kennzeichen für mehrere Varianten zu erstellen.
- Kontext: Benutzerdefinierte Schlüssel und Werte, an die AWS AppConfig beim Abrufen der Konfiguration übergeben wurde. Kontextwerte werden bei der Regelauswertung verwendet, um die Feature-Flag-Variante auszuwählen, die zurückgegeben werden soll.

> **Note**
>
> AWS AppConfig Der Agent bewertet Variantenregeln und bestimmt anhand des bereitgestellten Kontextes, welche Regel für die Anfrage gilt. Weitere Informationen zum Abrufen von Feature-Flags mit mehreren Varianten finden Sie unter. Feature-Flags für grundlegende und variantenreiche Funktionen werden abgerufen

Allgemeine Anwendungsfälle

In diesem Abschnitt werden zwei häufige Anwendungsfälle für Feature-Flag-Varianten beschrieben.

Benutzersegmentierung

Bei der Benutzersegmentierung werden Benutzer anhand bestimmter Attribute aufgeteilt. Sie könnten beispielsweise Flaggenvarianten verwenden, um eine Funktion einigen Benutzern zugänglich zu machen, anderen jedoch nicht, basierend auf ihrer Benutzer-ID, ihrem geografischen Standort, ihrem Gerätetyp oder ihrer Kaufhäufigkeit.

Nehmen wir am Beispiel der Kaufhäufigkeit an, dass Ihre E-Commerce-Anwendung eine Funktion zur Steigerung der Kundenbindung unterstützt. Sie können Flaggenvarianten verwenden, um verschiedene Arten von Anreizen zu konfigurieren, die einem Nutzer angezeigt werden, je nachdem, wann er zuletzt etwas gekauft hat. Einem neuen Benutzer kann ein kleiner discount angeboten

werden, um ihn zu ermutigen, Kunde zu werden, wohingegen einem Stammkunden möglicherweise ein größerer discount gewährt wird, wenn er etwas aus einer neuen Kategorie kauft.

Datenverkehrsaufteilung

Beim Traffic Splitting wird eine zufällige, aber konsistente Flaggenvariante ausgewählt, die auf einem von Ihnen definierten Kontextwert basiert. Möglicherweise möchten Sie ein Experiment durchführen, bei dem ein kleiner Prozentsatz Ihrer Benutzer (identifiziert anhand ihrer Benutzer-ID) eine bestimmte Variante sieht. Oder Sie möchten eine schrittweise Einführung einer Funktion durchführen, bei der eine Funktion zunächst für 5% Ihrer Benutzer verfügbar ist, dann für 15%, dann für 40% und dann für 100%, wobei während der gesamten Einführung ein einheitliches Benutzererlebnis gewährleistet ist.

Anhand des Versuchsbeispiels könnten Sie Flaggenvarianten verwenden, um einen neuen Schaltflächenstil für die primäre Aktion auf der Startseite Ihrer Anwendung zu testen, um festzustellen, ob er zu mehr Klicks führt. Für Ihr Experiment könnten Sie eine Flaggenvariante mit einer Regel zur Aufteilung des Datenverkehrs erstellen, bei der 5% der Benutzer ausgewählt werden, um den neuen Stil zu sehen, während die Standardvariante die Benutzer angibt, denen der bestehende Stil weiterhin angezeigt werden soll. Wenn das Experiment erfolgreich ist, können Sie den Prozentwert erhöhen oder diese Variante sogar zur Standardvariante machen.

Grundlegendes zu Feature-Flag-Regeln mit mehreren Varianten

Wenn Sie eine Feature-Flag-Variante erstellen, geben Sie eine Regel für diese an. Regeln sind Ausdrücke, die Kontextwerte als Eingabe verwenden und als Ausgabe ein boolesches Ergebnis erzeugen. Sie könnten beispielsweise eine Regel definieren, um eine Flag-Variante für Beta-Benutzer auszuwählen, die anhand ihrer Konto-ID identifiziert werden, um eine Aktualisierung der Benutzeroberfläche zu testen. Für dieses Szenario gehen Sie wie folgt vor:

1. Erstellen Sie ein neues Feature-Flag-Konfigurationsprofil mit dem Namen UI Refresh.
2. Erstellen Sie ein neues Feature-Flag namens ui_refresh.
3. Bearbeiten Sie das Feature-Flag, nachdem Sie es erstellt haben, um Varianten hinzuzufügen.
4. Erstellen und aktivieren Sie eine neue Variante namens BetaUsers.
5. Definieren Sie eine Regel, die die Variante auswählt BetaUsers, wenn die Konto-ID aus dem Anforderungskontext in einer Liste von Konten enthalten ist, die für die Nutzung der neuen Beta-Version IDs zugelassen sind.
6. Vergewissern Sie sich, dass der Status der Standardvariante auf Deaktiviert gesetzt ist.

> **Note**
>
> Varianten werden anhand der Reihenfolge, in der sie in der Konsole definiert sind, als geordnete Liste bewertet. Die Variante ganz oben in der Liste wird zuerst ausgewertet. Wenn keine Regeln mit dem angegebenen Kontext übereinstimmen, wird die Standardvariante AWS AppConfig zurückgegeben.

Bei AWS AppConfig der Verarbeitung der Feature-Flag-Anforderung wird der angegebene Kontext, der die AccountID (für dieses Beispiel) enthält, zuerst mit der BetaUsers Variante verglichen. Wenn der Kontext der Regel für entspricht BetaUsers, werden die Konfigurationsdaten für das Beta-Erlebnis AWS AppConfig zurückgegeben. Wenn der Kontext keine Konto-ID enthält oder wenn die Konto-ID auf etwas anderes als 123 endet, werden Konfigurationsdaten für die Standardregel AWS AppConfig zurückgegeben, was bedeutet, dass der Benutzer das aktuelle Erlebnis in der Produktionsumgebung aufruft.

> **Note**
>
> Informationen zum Abrufen von Feature-Flags mit mehreren Varianten finden Sie unter. Feature-Flags für grundlegende und variantenreiche Funktionen werden abgerufen

Definieren von Regeln für Feature-Flags mit mehreren Varianten

Eine Variantenregel ist ein Ausdruck, der aus einem oder mehreren Operanden und einem Operator besteht. Ein Operand ist ein bestimmter Wert, der bei der Auswertung einer Regel verwendet wird. Operandenwerte können entweder statisch sein, z. B. eine Literalzahl oder Zeichenfolge, oder variabel, z. B. der in einem Kontext gefundene Wert oder das Ergebnis eines anderen Ausdrucks. Ein Operator, z. B. „größer als“, ist ein Test oder eine Aktion, die auf seine Operanden angewendet wird und einen Wert erzeugt. Ein Variantenregelausdruck muss entweder „wahr“ oder „falsch“ ergeben, um gültig zu sein.

Operanden

Typ	Beschreibung	Beispiel
String	Eine Folge von UTF-8-Zeichen, eingeschlossen in doppelte Anführungszeichen.	`"apple", "####ë# ##š##"`
Ganzzahl	Ein 64-Bit-Ganzzahlwert.	`-7, 42`
Gleitkommazahl	Ein 64-Bit-IEEE-754-Gleitkommawert.	`3.14, 1.234e-5`
Zeitstempel	Ein bestimmter Zeitpunkt, wie in der W3C-Anmerkung zu Datums- und Uhrzeitformaten beschrieben.	`2012-03-04T05:06:07-08:00, 2024-01`
Boolesch	Ein wahrer oder falscher Wert.	`true, false`
Wert im Kontext	Ein parametrisierter Wert in Form von $*key*, der während der Regelauswertung aus dem Kontext abgerufen wird.	`$country, $userId`

Vergleichsoperatoren

Operator	Beschreibung	Beispiel
eq	Ermittelt, ob ein Kontextwert einem bestimmten Wert entspricht.	`(eq $state "Virginia")`
gt	Ermittelt, ob ein Kontextwert größer als ein bestimmter Wert ist.	`(gt $age 65)`

Operator	Beschreibung	Beispiel
gte	Ermittelt, ob ein Kontextwert größer oder gleich einem bestimmten Wert ist.	`(gte $age 65)`
lt	Ermittelt, ob ein Kontextwert kleiner als ein bestimmter Wert ist.	`(lt $age 65)`
lte	Ermittelt, ob ein Kontextwert kleiner oder gleich einem bestimmten Wert ist.	`(lte $age 65)`

Logische Operatoren

Operator	Beschreibung	Beispiel

and	Ermittelt, ob beide Operanden wahr sind.	`(and`
		`  (eq $state "Virginia")`
		`  (gt $age 65)`
		`)`

or	Ermittelt, ob mindestens einer der Operanden wahr ist.	`(or`
		`  (eq $state "Virginia")`
		`  (gt $age 65)`
		`)`

not	Kehrt den Wert eines Ausdrucks um.	`(not (eq $state "Virginia"))`

Benutzerdefinierte Operatoren

Operator	Beschreibung	Beispiel
Beginnt mit	Bestimmt, ob ein Kontextwert mit einem bestimmten Präfix beginnt.	`(begins_with $state "A")`
ends_with	Bestimmt, ob ein Kontextwert mit einem bestimmten Präfix endet.	`(ends_with $email "amazon.com")`
enthält	Ermittelt, ob ein Kontextwert eine bestimmte Teilzeichenfolge enthält.	`(contains $promoCode "WIN")`
in	Ermittelt, ob ein Kontextwert in einer Liste von Konstanten enthalten ist.	`(in $userId ["123", "456"])`
Streichhölzer	Ermittelt, ob ein Kontextwert einem bestimmten Regex-Muster entspricht.	`(matches in::$greeting pattern::"h.*y")`
exists	Ermittelt, ob ein Wert für einen Kontextschlüssel angegeben wurde.	`(exists key::"country")`
split	Berechnet bis `true` für einen bestimmten Prozentsatz des Datenverkehrs auf der Grundlage eines konsistenten Hashwerts der angegebenen Kontextwerte. Eine ausführliche Erläuterung der `split` Funktionsweise finden Sie im nächsten Abschnitt dieses Themas, Den Split-Operator verstehen.	`(split pct::10 by::$userId seed::"abc")`

Operator	Beschreibung	Beispiel
	Beachten Sie, dass `seed` es sich um eine optionale Eigenschaft handelt. Wenn Sie nichts angeben`seed`, ist der Hash lokal konsisten t, was bedeutet, dass der Verkehr für dieses Flag konsistent aufgeteilt wird, aber andere Flags, die denselben Kontextwe rt erhalten, den Verkehr möglicherweise anders aufteilen. Wenn angegeben , `seed` wird garantiert, dass jeder eindeutige Wert den Datenverkehr konsistent auf Feature-Flags, Konfigura tionsprofile und aufteilt AWS-Konten.	

Den Split-Operator verstehen

Im folgenden Abschnitt wird beschrieben, wie sich der `split` Operator verhält, wenn er in verschiedenen Szenarien verwendet wird. Zur Erinnerung: Es wird `true` für einen bestimmten Prozentsatz des Datenverkehrs auf der Grundlage eines konsistenten Hashwerts des angegebenen Kontextwerts `split` ausgewertet. Um dies besser zu verstehen, stellen Sie sich das folgende Basisszenario vor, das Split mit zwei Varianten verwendet:

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

Erwartungsgemäß ergibt die Bereitstellung einer zufälligen Menge von `uniqueId` Werten eine Verteilung, die ungefähr so aussieht:

```
A: 20%
```

```
C: 80%
```

Wenn Sie eine dritte Variante hinzufügen, aber denselben aufgeteilten Prozentsatz wie folgt verwenden:

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

Am Ende erhalten Sie die folgende Verteilung:

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

Diese potenziell unerwartete Verteilung tritt auf, weil jede Variantenregel der Reihe nach ausgewertet wird und die erste Übereinstimmung die zurückgegebene Variante bestimmt. Wenn Regel A ausgewertet wird, stimmen 20% der `uniqueId` Werte mit ihr überein, sodass die erste Variante zurückgegeben wird. Als Nächstes wird Regel B ausgewertet. Alle `uniqueId` Werte, die mit der zweiten Split-Anweisung übereinstimmen würden, wurden jedoch bereits durch Variantenregel A abgeglichen, sodass keine Werte mit B übereinstimmen. Stattdessen wird die Standardvariante zurückgegeben.

Betrachten Sie nun ein drittes Beispiel.

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

Wie im vorherigen Beispiel entsprechen die ersten 20% der `uniqueId` Werte Regel A. Bei Variante B würden 25% aller `uniqueId` Werte übereinstimmen, aber die meisten der Werte, die zuvor mit Regel A übereinstimmten, so dass 5% der Gesamtsumme für Variante B übrig bleiben und der Rest Variante C erhält. Die Verteilung würde wie folgt aussehen:

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

Verwendung der **seed** Eigenschaft

Sie können die `seed` Eigenschaft verwenden, um sicherzustellen, dass der Datenverkehr für einen bestimmten Kontextwert konsistent aufgeteilt wird, unabhängig davon, wo der Split-Operator verwendet wird. Wenn Sie nichts angeben`seed`, ist der Hash lokal konsistent, was bedeutet, dass der Verkehr für dieses Flag konsistent aufgeteilt wird, aber andere Flags, die denselben Kontextwert erhalten, den Verkehr möglicherweise anders aufteilen. Wenn angegeben, `seed` wird garantiert, dass jeder eindeutige Wert den Datenverkehr konsistent auf Feature-Flags, Konfigurationsprofile und aufteilt AWS-Konten.

In der Regel verwenden Kunden denselben `seed` Wert für alle Varianten innerhalb eines Flags, wenn sie den Traffic auf dieselbe Kontexteigenschaft aufteilen. Gelegentlich kann es jedoch sinnvoll sein, einen anderen Startwert zu verwenden. Hier ist ein Beispiel, das unterschiedliche Ausgangswerte für die Regeln A und B verwendet:

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

Wie zuvor entsprechen 20% der übereinstimmenden `uniqueId` Werte Regel A. Das bedeutet, dass 80% der Werte durchfallen und anhand der Variantenregel B getestet werden. Da es sich um einen anderen Ausgangswert handelt, besteht keine Korrelation zwischen den Werten, die mit A übereinstimmen, und den Werten, die B entsprechen. Es müssen jedoch nur 80% so viele `uniqueId` Werte aufgeteilt werden, wobei 25% dieser Zahl Regel B entsprechen und 75% nicht. Das ergibt die folgende Verteilung:

```
A: 20%
B: 20% (25% of what falls through from A, or 25% of 80%)
C: 60%
```

Erstellen eines Feature-Flags mit mehreren Varianten

Verwenden Sie die Verfahren in diesem Abschnitt, um Varianten eines Feature-Flags zu erstellen.

Bevor Sie beginnen

Notieren Sie die folgenden wichtigen Informationen.

- Sie können Varianten vorhandener Feature-Flags erstellen, indem Sie sie bearbeiten. Sie können keine Varianten eines neuen Feature-Flags erstellen, wenn Sie ein neues Konfigurationsprofil

erstellen. Sie müssen zuerst den Arbeitsablauf zur Erstellung des neuen Konfigurationsprofils abschließen. Nachdem Sie das Konfigurationsprofil erstellt haben, können Sie jeder Markierung innerhalb des Konfigurationsprofils Varianten hinzufügen. Informationen zum Erstellen eines neuen Konfigurationsprofils finden Sie unterErstellen eines Feature-Flag-Konfigurationsprofils in AWS AppConfig.

- Um Feature-Flag-Variantendaten für die Rechenplattformen Amazon EC2, Amazon ECS und Amazon EKS abzurufen, müssen Sie AWS AppConfig Agent Version 2.0.4416 oder höher verwenden.
- Aus Leistungsgründen AWS CLI und aus SDK-Aufrufen werden AWS AppConfig keine Variantendaten abgerufen. Weitere Informationen zu AWS AppConfig Agent finden Sie unterWie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten.
- Wenn Sie eine Feature-Flag-Variante erstellen, geben Sie eine Regel dafür an. Regeln sind Ausdrücke, die den Anforderungskontext als Eingabe verwenden und als Ausgabe ein boolesches Ergebnis erzeugen. Bevor Sie Varianten erstellen, überprüfen Sie die unterstützten Operanden und Operatoren für Flag-Variantenregeln. Sie können Regeln erstellen, bevor Sie Varianten erstellen. Weitere Informationen finden Sie unter Grundlegendes zu Feature-Flag-Regeln mit mehreren Varianten.

Themen

- Feature-Flag mit mehreren Varianten erstellen (Konsole)
- Ein Feature-Flag mit mehreren Varianten erstellen (Befehlszeile)

Feature-Flag mit mehreren Varianten erstellen (Konsole)

Im folgenden Verfahren wird beschrieben, wie Sie mithilfe der Konsole ein Feature-Flag mit mehreren Varianten für ein vorhandenes Konfigurationsprofil erstellen. AWS AppConfig Sie können auch vorhandene Feature-Flags bearbeiten, um Varianten zu erstellen.

Um ein Feature-Flag mit mehreren Varianten zu erstellen

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Anwendungen und dann eine Anwendung aus.
3. Wählen Sie auf der Registerkarte Konfigurationsprofile und Feature-Flags ein vorhandenes Feature-Flag-Konfigurationsprofil aus.

4. Wählen Sie im Abschnitt Flags die Option Neue Flagge hinzufügen aus.
5. Geben Sie im Abschnitt Feature-Flag-Definition für Flagname einen Namen ein.
6. Geben Sie unter Flag-Schlüssel eine Flag-ID ein, um Flags innerhalb desselben Konfigurationsprofils voneinander zu unterscheiden. Flags innerhalb desselben Konfigurationsprofils können nicht denselben Schlüssel haben. Nachdem das Flag erstellt wurde, können Sie den Flag-Namen bearbeiten, aber nicht den Flaggenschlüssel.
7. (Optional) Geben Sie im Feld Beschreibung Informationen zu dieser Flagge ein.
8. Wählen Sie im Abschnitt Varianten die Option Flagge mit mehreren Varianten aus.
9. (Optional) Wählen Sie im Abschnitt Feature-Flag-Attribute die Option Attribut definieren aus. Mithilfe von Attributen können Sie zusätzliche Werte in Ihrer Flagge angeben. Weitere Informationen zu Attributen und Einschränkungen finden Sie unterDie Attribute von Feature-Flags verstehen.

    a. Geben Sie für Schlüssel einen Flaggenschlüssel an und wählen Sie seinen Typ aus der Liste Typ aus. Informationen zu den unterstützten Optionen für die Felder Wert und Einschränkungen finden Sie im Abschnitt über Attribute, auf den zuvor verwiesen wurde.

    b. Wählen Sie Erforderlicher Wert aus, um anzugeben, ob ein Attributwert erforderlich ist.

    c. Wählen Sie „Attribut definieren“, um weitere Attribute hinzuzufügen.

    d. Wählen Sie Anwenden, um die Attributänderungen zu speichern.

10. Wählen Sie im Abschnitt Feature-Flag-Varianten die Option Variante erstellen aus.

    a. Geben Sie unter Variantenname einen Namen ein.

    b. Verwenden Sie den Schalter Aktivierter Wert, um die Variante zu aktivieren.

    c. Geben Sie im Textfeld Regel eine Regel ein.

    d. Verwenden Sie die Optionen Variante erstellen > Variante oben erstellen oder Variante unten erstellen, um zusätzliche Varianten für dieses Kennzeichen zu erstellen.

    e. Verwenden Sie im Abschnitt Standardvariante den Schalter Aktivierter Wert, um die Standardvariante zu aktivieren. Geben Sie optional Werte für die in Schritt 10 definierten Attribute an.

    f. Wählen Sie Anwenden aus.

11. Überprüfen Sie die Details der Markierung und ihrer Varianten und wählen Sie Flagge erstellen aus.

Informationen zur Bereitstellung Ihres neuen Feature-Flags mit Varianten finden Sie unterBereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig.

Ein Feature-Flag mit mehreren Varianten erstellen (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie das AWS Command Line Interface (unter Linux oder Windows) oder Tools für Windows verwenden, PowerShell um ein multivariantes Feature-Flag für ein vorhandenes Konfigurationsprofil zu erstellen. Sie können auch vorhandene Feature-Flags bearbeiten, um Varianten zu erstellen.

Bevor Sie beginnen

Führen Sie die folgenden Aufgaben aus, bevor Sie ein Feature-Flag mit mehreren Varianten mithilfe von erstellen. AWS CLI

- Erstellen Sie ein Feature-Flag-Konfigurationsprofil. Weitere Informationen finden Sie unter Erstellen eines Feature-Flag-Konfigurationsprofils in AWS AppConfig.
- Aktualisieren Sie auf die neueste Version von AWS CLI. Weitere Informationen finden Sie im AWS Command Line Interface Benutzerhandbuch unter Installation oder Aktualisierung AWS CLI auf die neueste Version von.

Um ein Feature-Flag mit mehreren Varianten zu erstellen

1. Erstellen Sie auf Ihrem lokalen Computer eine Konfigurationsdatei, die die Details des Flag mit mehreren Varianten angibt, das Sie erstellen möchten. Speichern Sie die Datei mit einer `.json` Dateierweiterung. Die Datei muss dem `AWS.AppConfig.FeatureFlags`JSON-Schema entsprechen. Der Schemainhalt Ihrer Konfigurationsdatei wird dem Folgenden ähneln.

```
{
  "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"
}
```

Hier ist ein Beispiel mit drei Varianten und der Standardvariante.

```
{
  "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. Verwenden Sie die `CreateHostedConfigurationVersion` API, um Ihre Feature-Flag-Konfigurationsdaten zu speichern 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
   ```

Das `service_returned_content_file` enthält Ihre Konfigurationsdaten, die einige AWS AppConfig generierte Metadaten enthalten.

> **Note**
>
> Wenn Sie die gehostete Konfigurationsversion erstellen, wird AWS AppConfig überprüft, ob Ihre Daten dem `AWS.AppConfig.FeatureFlags`JSON-Schema entsprechen. AWS AppConfig überprüft außerdem, ob jedes Feature-Flag-Attribut in Ihren Daten die Einschränkungen erfüllt, die Sie für diese Attribute definiert haben.

Grundlegendes zur Typenreferenz für AWS.AppConfig.FeatureFlags

Verwenden Sie das `AWS.AppConfig.FeatureFlags` JSON-Schema als Referenz, um Ihre Feature-Flag-Konfigurationsdaten zu erstellen.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "definitions": {
    "flagSetDefinition": {
      "type": "object",
      "properties": {
        "version": {
          "$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**
>
> Um Feature-Flag-Konfigurationsdaten abzurufen, muss Ihre Anwendung die `GetLatestConfiguration` API aufrufen. Sie können die Konfigurationsdaten für Feature-Flags nicht durch einen Aufruf abrufen`GetConfiguration`, was veraltet ist. Weitere Informationen finden Sie unter GetLatestConfiguration in der AWS AppConfig -API-Referenz.

Wenn Ihre Anwendung eine neu bereitgestellte Konfiguration aufruft GetLatestConfigurationund empfängt, werden die Informationen, die Ihre Feature-Flags und -Attribute definieren, entfernt. Das vereinfachte JSON enthält eine Zuordnung von Schlüsseln, die mit jedem der von Ihnen angegebenen Flaggenschlüssel übereinstimmen. Das vereinfachte JSON enthält auch zugeordnete Werte von `true` oder `false` für das `enabled` Attribut. Wenn ein Flag `enabled` auf gesetzt ist`true`, sind alle Attribute des Flags ebenfalls vorhanden. Das folgende JSON-Schema beschreibt das Format der JSON-Ausgabe.

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

```
      },
      "maxProperties": 25,
      "additionalProperties": false
    },
    "attributeValue": {
      "oneOf": [
        { "type": "string","maxLength": 1024 },
        { "type": "number" },
        { "type": "boolean" },
        {
          "type": "array",
          "oneOf": [
            {
              "items": {
                "oneOf": [
                  {
                    "type": "string",
                    "maxLength": 1024
                  }
                ]
              }
            },
            {
              "items": {
                "oneOf": [
                  {
                    "type": "number"
                  }
                ]
              }
            }
          ]
        }
      ],
      "additionalProperties": false
    }
  }
}
```

Speichern einer früheren Feature-Flag-Version in einer neuen Version

Wenn Sie ein Feature-Flag aktualisieren, AWS AppConfig werden Ihre Änderungen automatisch in einer neuen Version gespeichert. Wenn Sie eine frühere Feature-Flag-Version verwenden möchten,

müssen Sie sie in eine Entwurfsversion kopieren und dann speichern. Sie können Änderungen an einer früheren Flag-Version nicht bearbeiten und speichern, ohne sie in einer neuen Version zu speichern.

Um eine frühere Feature-Flag-Version zu bearbeiten und sie in einer neuen Version zu speichern

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Anwendungen und dann die Anwendung mit dem Feature-Flag aus, die Sie bearbeiten und in einer neuen Version speichern möchten.
3. Wählen Sie auf der Registerkarte Konfigurationsprofile und Feature-Flags das Konfigurationsprofil mit dem Feature-Flag aus, das Sie bearbeiten und in einer neuen Version speichern möchten.
4. Wählen Sie auf der Registerkarte Feature-Flags in der Versionsliste die Version aus, die Sie bearbeiten und in einer neuen Version speichern möchten.
5. Wählen Sie In Entwurfsversion kopieren aus.
6. Geben Sie im Feld Versionsbezeichnung eine neue Bezeichnung ein (optional, aber empfohlen).
7. Geben Sie im Feld Versionsbeschreibung eine neue Beschreibung ein (optional, aber empfohlen).
8. Wählen Sie Version speichern.
9. Wählen Sie Bereitstellung starten, um die neue Version bereitzustellen.

Erstellen Sie ein frei formatiertes Konfigurationsprofil in AWS AppConfig

Bei Konfigurationsdaten handelt es sich um eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Ein Konfigurationsprofil umfasst unter anderem eine URI, die es ermöglicht, Ihre Konfigurationsdaten AWS AppConfig an ihrem gespeicherten Speicherort zu finden, und einen Konfigurationstyp. Mit Freiform-Konfigurationsprofilen können Sie Ihre Daten im AWS AppConfig gehosteten Konfigurationsspeicher oder in einem der folgenden Tools AWS-Services und Systems Manager Manager-Tools speichern:

Ort	Unterstützte Dateitypen
AWS AppConfig gehosteter Konfigurationsspeicher	YAML, JSON und Text, falls mit dem AWS Management Console hinzugefügt. Beliebiger

Ort	Unterstützte Dateitypen
	Dateityp, wenn er mithilfe der AWS AppConfig CreateHostedConfigurationVersionAPI-Aktion hinzugefügt wurde.
Amazon Simple Storage Service (Amazon-S3)	Any
AWS CodePipeline	Pipeline (wie vom Dienst definiert)
AWS Secrets Manager	Geheim (wie vom Dienst definiert)
AWS Systems Manager Parameter Store	Standard- und sichere Zeichenkettenparameter (wie von Parameter Store definiert)
AWS Systems Manager Dokumentenspeicher (SSM-Dokumente)	YAML, JSON, Text

Ein Konfigurationsprofil kann auch optionale Validatoren enthalten, um sicherzustellen, dass Ihre Konfigurationsdaten syntaktisch und semantisch korrekt sind. AWS AppConfig führt eine Überprüfung mithilfe der Validatoren durch, wenn Sie eine Bereitstellung starten. Werden Fehler erkannt, wird die Bereitstellung beendet, bevor Änderungen an den Zielen der Konfiguration vorgenommen werden.

> **Note**
>
> Wenn möglich, empfehlen wir, Ihre Konfigurationsdaten im AWS AppConfig gehosteten Konfigurationsspeicher zu hosten, da dieser die meisten Funktionen und Verbesserungen bietet.

Für Freiformkonfigurationen, die im AWS AppConfig gehosteten Konfigurationsspeicher oder in SSM-Dokumenten gespeichert sind, können Sie die Freiformkonfiguration mithilfe der Systems Manager Manager-Konsole erstellen, wenn Sie ein Konfigurationsprofil erstellen. Der Prozess wird weiter unten in diesem Thema beschrieben.

Für Freiformkonfigurationen, die in Parameter Store, Secrets Manager oder Amazon S3 gespeichert sind, müssen Sie zuerst den Parameter, das Geheimnis oder das Objekt erstellen und im entsprechenden Konfigurationsspeicher speichern. Nachdem Sie die Konfigurationsdaten

gespeichert haben, verwenden Sie das Verfahren in diesem Thema, um das Konfigurationsprofil zu erstellen.

Themen

- Validatoren verstehen
- Grundlegendes zu Kontingenten und Einschränkungen des Konfigurationsspeichers
- Grundlegendes zum AWS AppConfig gehosteten Konfigurationsspeicher
- Grundlegendes zu den in Amazon S3 gespeicherten Konfigurationen
- Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils (Konsole)
- Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils (Befehlszeile)

Validatoren verstehen

Wenn Sie ein Konfigurationsprofil erstellen, haben Sie die Möglichkeit, bis zu zwei Validatoren anzugeben. Ein Validator stellt sicher, dass Ihre Konfigurationsdaten syntaktisch und semantisch korrekt sind. Wenn Sie einen Validator verwenden möchten, müssen Sie ihn erstellen, bevor Sie das Konfigurationsprofil erstellen. AWS AppConfig unterstützt die folgenden Arten von Validatoren:

- AWS Lambda Funktionen: Wird für Feature-Flags und Freiformkonfigurationen unterstützt.
- JSON-Schema: Wird für Freiformkonfigurationen unterstützt. (validiert Feature-Flags AWS AppConfig automatisch anhand eines JSON-Schemas.)

Themen

- AWS Lambda Funktionsvalidatoren
- JSON-Schema-Validatoren

AWS Lambda Funktionsvalidatoren

Lambda-Funktionsvalidatoren müssen mit dem folgenden Ereignisschema konfiguriert werden. AWS AppConfig verwendet dieses Schema, um die Lambda-Funktion aufzurufen. Der Inhalt ist eine Base64-codierte Zeichenfolge und der URI ist eine Zeichenfolge.

```
{
    "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 überprüft, ob der `X-Amz-Function-Error` Lambda-Header in der Antwort gesetzt ist. Lambda setzt diesen Header, wenn die Funktion eine Ausnahme auslöst. Weitere Informationen zu *`X-Amz-Function-Error`* finden Sie unter Fehlerbehandlung und automatische Wiederholungen AWS Lambda im AWS Lambda Entwicklerhandbuch.

Hier ist ein einfaches Beispiel für einen Lambda-Antwortcode für eine erfolgreiche Validierung.

```
import json

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

Hier ist ein einfaches Beispiel für einen Lambda-Antwortcode für eine erfolglose Validierung.

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

Hier sehen Sie ein weiteres Beispiel, bei dem nur überprüft wird, ob der Konfigurationsparameter eine Primzahl ist.

```
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 ruft Ihre Validierung Lambda auf, wenn Sie die `StartDeployment` und `ValidateConfigurationActivity` API-Operationen aufrufen. Sie müssen `appconfig.amazonaws.com` Berechtigungen bereitstellen, um Ihr Lambda aufzurufen. Weitere Informationen finden Sie unter Funktionszugriff auf Dienste gewähren. AWS AWS AppConfig begrenzt die Validierungs-Lambda-Laufzeit auf 15 Sekunden, einschließlich Startlatenz.

JSON-Schema-Validatoren

Wenn Sie eine Konfiguration in einem SSM-Dokument erstellen, müssen Sie ein JSON-Schema für diese Konfiguration angeben oder erstellen. Ein JSON-Schema definiert die zulässigen Eigenschaften für jede Anwendungskonfigurationseinstellung. Das JSON-Schema funktioniert wie eine Reihe von Regeln. Es stellt sicher, dass neue oder aktualisierte Konfigurationseinstellungen den bewährten Methoden Ihrer Anwendung entsprechen. Ein Beispiel.

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

Wenn Sie eine Konfiguration aus einem SSM-Dokument erstellen, überprüft das System automatisch, ob die Konfiguration den Schemaanforderungen entspricht. Ist dies nicht der Fall, gibt AWS AppConfig einen Validierungsfehler zurückgegeben.

Important

Beachten Sie die folgenden wichtigen Informationen zu JSON-Schemavalidatoren:

- Konfigurationsdaten, die in SSM-Dokumenten gespeichert sind, müssen anhand eines zugehörigen JSON-Schemas validiert werden, bevor Sie die Konfiguration dem System hinzufügen können. SSM-Parameter erfordern keine Validierungsmethode, wir empfehlen jedoch, dass Sie eine Validierungsprüfung für neue oder aktualisierte SSM-Parameterkonfigurationen mithilfe von erstellen. AWS Lambda
- Eine Konfiguration in einem SSM-Dokument verwendet den `ApplicationConfiguration` Dokumenttyp. Das entsprechende JSON-Schema verwendet den `ApplicationConfigurationSchema` Dokumenttyp.
- AWS AppConfig unterstützt JSON-Schemaversion 4.X für Inline-Schemas. Wenn Ihre Anwendungskonfiguration eine andere Version des JSON-Schemas erfordert, müssen Sie einen Lambda-Validator erstellen.

Grundlegendes zu Kontingenten und Einschränkungen des Konfigurationsspeichers

Für Konfigurationsspeicher, die von AWS AppConfig unterstützt werden, gelten die folgenden Kontingente und Einschränkungen.

	AWS AppConfig gehosteter Konfigurationsspeicher	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager Dokumentenspeicher	AWS CodePipeline
Begrenzung der Konfigurationsgröße	Standard: 2 MB, maximal 4 MB	2 MB Erzwungen durch AWS AppConfig, nicht durch S3	4 KB (kostenloses Kontingent)/8 KB (erweiterte Parameter)	64 KB	64 KB	2 MB Erzwungen von AWS AppConfig, nicht CodePipeline

	AWS AppConfig gehosteter Konfigurationsspeicher	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager Dokumentenspeicher	AWS CodePipeline
Ressourcenspeicherlimit	1 GB	Unbegrenzt	10.000 Parameter (kostenloses Kontingent)/100.000 Parameter (erweiterte Parameter)	500 000	500 Dokumente	Beschränkt durch die Anzahl der Konfigurationsprofile pro Anwendung (100 Profile pro Anwendung)
Server-side encryption	Ja	SSE-S3, SSE-KMS	Ja	Ja	Nein	Ja
AWS CloudFormation Unterstützung	Ja	Nicht zum Erstellen oder Aktualisieren von Daten	Ja	Ja	Nein	Ja
Preise	Kostenfrei	Sehen Sie sich die Amazon S3 S3-Preise an	AWS Systems Manager Preise ansehen	AWS Secrets Manager Preise ansehen	Kostenfrei	AWS CodePipeline Preise ansehen

Grundlegendes zum AWS AppConfig gehosteten Konfigurationsspeicher

AWS AppConfig beinhaltet einen internen oder gehosteten Konfigurationsspeicher. Konfigurationen müssen 2 MB oder weniger groß sein. Der AWS AppConfig gehostete Konfigurationsspeicher bietet die folgenden Vorteile gegenüber anderen Konfigurationsspeicheroptionen.

- Sie müssen keine anderen Services wie Amazon Simple Storage Service (Amazon S3) oder Parameterspeicher einrichten und konfigurieren.
- Sie müssen keine AWS Identity and Access Management (IAM-) Berechtigungen konfigurieren, um den Konfigurationsspeicher verwenden zu können.
- Sie können Konfigurationen in YAML, JSON oder als Textdokumente speichern.
- Für die Nutzung des Speichers fallen keine Kosten an.
- Sie können eine Konfiguration erstellen und dem Speicher hinzufügen, wenn Sie ein Konfigurationsprofil erstellen.

Grundlegendes zu den in Amazon S3 gespeicherten Konfigurationen

Sie können Konfigurationen in einem Amazon Simple Storage Service (Amazon S3) -Bucket speichern. Wenn Sie das Konfigurationsprofil erstellen, geben Sie den URI für ein einzelnes S3-Objekt an in einem Bucket an. Sie geben auch den Amazon-Ressourcennamen (ARN) einer AWS Identity and Access Management (IAM) -Rolle an, die die AWS AppConfig Berechtigung zum Abrufen des Objekts erteilt. Bevor Sie ein Konfigurationsprofil für ein Amazon S3 S3-Objekt erstellen, sollten Sie die folgenden Einschränkungen beachten.

Einschränkung	Details
Größe	Konfigurationen, die als S3-Objekte gespeichert werden, können maximal 1 MB groß sein.
Objekt-Verschlüsselung	Ein Konfigurationsprofil kann auf SSE-S3- und SSE-KMS-verschlüsselte Objekte abzielen.
Speicherklassen	AWS AppConfig unterstützt die folgenden S3-Speicherklassen:`STANDARD`,,`INTELLIGENT_TIERING` , `REDUCED_REDUNDANCY` und. `STANDARD_IA` `ONEZONE_IA` Die folgenden Klassen werden nicht unterstüt

Einschränkung	Details
	zt: Alle S3-Glacier-Klassen (`GLACIER` und `DEEP_ARCHIVE`).
Versionsverwaltung	AWS AppConfig erfordert, dass das S3-Objekt Versionierung verwendet.

Konfiguration von Berechtigungen für eine als Amazon S3 S3-Objekt gespeicherte Konfiguration

Wenn Sie ein Konfigurationsprofil für eine als S3-Objekt gespeicherte Konfiguration erstellen, müssen Sie einen ARN für eine IAM-Rolle angeben, die die AWS AppConfig Berechtigung zum Abrufen des Objekts erteilt. Die Rolle muss die folgenden Berechtigungen enthalten:

Berechtigungen für den Zugriff auf das S3-Objekt

- s3: GetObject
- s3: GetObjectVersion

Berechtigungen zum Auflisten von S3-Buckets

s3: ListAllMyBuckets

Berechtigungen für den Zugriff auf den S3-Bucket, in dem das Objekt gespeichert ist

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

Gehen Sie wie folgt vor, um eine Rolle zu erstellen, mit AWS AppConfig der eine in einem S3-Objekt gespeicherte Konfiguration abgerufen werden kann.

Erstellen der IAM-Richtlinie für den Zugriff auf ein S3-Objekt

Gehen Sie wie folgt vor, um eine IAM-Richtlinie zu erstellen, mit der eine in einem S3-Objekt gespeicherte Konfiguration abgerufen werden kann AWS AppConfig .

Um eine IAM-Richtlinie für den Zugriff auf ein S3-Objekt zu erstellen

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Richtlinien und dann Richtlinie erstellen.
3. Wählen Sie auf der Seite Richtlinie erstellen die Registerkarte JSON aus.
4. Aktualisieren Sie die folgende Beispielrichtlinie mit Informationen zu Ihrem S3-Bucket und Konfigurationsobjekt. Fügen Sie dann die Richtlinie in das Textfeld auf der Registerkarte JSON ein. Ersetzen Sie *placeholder values* durch Ihre Informationen.

```
{
  "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. Wählen Sie Richtlinie prüfen.

6. Geben Sie auf der Seite Review policy (Prüfungsrichtlinie) im Feld Name einen Namen und anschließend eine Beschreibung ein.
7. Wählen Sie Richtlinie erstellen aus. Das System leitet Sie zur Seite Rollen zurück.

Erstellen der IAM-Rolle für den Zugriff auf ein S3-Objekt

Gehen Sie wie folgt vor, um eine IAM-Rolle zu erstellen, mit der eine in einem S3-Objekt gespeicherte Konfiguration abgerufen werden kann AWS AppConfig .

So erstellen Sie eine IAM-Rolle für den Zugriff auf ein Amazon S3 S3-Objekt

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie im Abschnitt Typ der vertrauenswürdigen Entität auswählen die Option AWS Service aus.
4. Wählen Sie im Abschnitt Einen Anwendungsfall auswählen unter Allgemeine Anwendungsfälle die Option EC2und anschließend Weiter: Berechtigungen aus.
5. Geben Sie auf der Seite Attach permissions policy (Berechtigungsrichtlinie anfügen) im Suchfeld den Namen der Richtlinie ein, die Sie im vorherigen Verfahren erstellt haben.
6. Wählen Sie diese Richtlinie aus und klicken Sie dann auf Next: Tags (Weiter: Tags).
7. Geben Sie auf der Seite Tags hinzufügen (optional) einen Schlüssel und einen optionalen Wert ein und wählen Sie dann Weiter: Überprüfen aus.
8. Geben Sie auf der Seite Review (Überprüfen) im Feld Role name (Rollenname) einen Namen und anschließend eine Beschreibung ein.
9. Wählen Sie Create role (Rolle erstellen) aus. Das System leitet Sie zur Seite Rollen zurück.
10. Wählen Sie auf der Seite Roles (Rollen) die gerade erstellte Rolle aus, um die Seite Summary (Übersicht) zu öffnen. Notieren Sie sich den Role Name (Rollenname) und Role ARN (Rollen-ARN). Sie geben den Rollen-ARN an, wenn Sie das Konfigurationsprofil später in diesem Thema erstellen.

Erstellen einer Vertrauensstellung

Gehen Sie wie folgt vor, um die Rolle, die Sie gerade erstellt haben, für AWS AppConfig als Vertrauensstellung zu konfigurieren.

So fügen Sie eine Vertrauensstellung hinzu:

1. Wählen Sie auf der Seite Summary für die eben erstellte Rolle die Registerkarte Trust Relationships und wählen Sie dann Edit Trust Relationship.
2. Löschen Sie `"ec2.amazonaws.com"` und fügen Sie `"appconfig.amazonaws.com"` hinzu, wie im folgenden Beispiel gezeigt.

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

3. Wählen Sie Update Trust Policy (Trust Policy aktualisieren).

Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils (Konsole)

Gehen Sie wie folgt vor, um ein AWS AppConfig Freiform-Konfigurationsprofil und (optional) eine Freiform-Konfiguration mithilfe der Konsole zu erstellen. AWS Systems Manager

Um ein Freiform-Konfigurationsprofil zu erstellen

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Anwendungen und dann eine Anwendung aus, in der Sie sie erstellt haben. Erstellen Sie einen Namespace für Ihre Anwendung in AWS AppConfig
3. Wählen Sie die Registerkarte Konfigurationsprofile und Feature-Flags und dann Konfiguration erstellen aus.
4. Wählen Sie im Abschnitt Konfigurationsoptionen die Option Freiform-Konfiguration aus.
5. Geben Sie unter Name des Konfigurationsprofils einen Namen für das Konfigurationsprofil ein.
6. (Optional) Erweitern Sie Beschreibung und geben Sie eine Beschreibung ein.

7. (Optional) Erweitern Sie Zusätzliche Optionen und füllen Sie bei Bedarf die folgenden Schritte aus.

   a. Wählen Sie im Abschnitt „Erweiterungen zuordnen“ eine Erweiterung aus der Liste aus.

   b. Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus und geben Sie dann einen Schlüssel und einen optionalen Wert an.

8. Wählen Sie Weiter aus.

9. Wählen Sie auf der Seite „Konfigurationsdaten angeben“ im Abschnitt „Konfigurationsdefinition“ eine Option aus.

10. Füllen Sie die Felder für die gewählte Option aus, wie in der folgenden Tabelle beschrieben.

Option ausgewählt	Details
AWS AppConfig gehostete Konfiguration	Wählen Sie entweder Text, JSON oder YAML und geben Sie Ihre Konfiguration in das Feld ein. Fahren Sie in diesem Verfahren mit Schritt 12 fort.
Amazon S3 S3-Objekt	Geben Sie die Objekt-URI in das Feld S3-Objektquelle ein und fahren Sie mit Schritt 11 in diesem Verfahren fort.
AWS CodePipeline	Wählen Sie Weiter und fahren Sie in diesem Verfahren mit Schritt 12 fort.
Secrets Manager geheim	Wählen Sie das Geheimnis aus der Liste aus. Fahren Sie in diesem Verfahren mit Schritt 11 fort.
AWS Systems Manager Parameter	Wählen Sie den Parameter aus der Liste aus und fahren Sie in diesem Verfahren mit Schritt 11 fort.
AWS Systems Manager document	1. Wählen Sie ein Dokument aus der Liste aus oder wählen Sie Neues Dokument erstellen.

Option ausgewählt	Details
	2. Wenn Sie Neues Dokument erstellen wählen, geben Sie unter Dokumentname einen Namen ein. Erweitern Sie optional den Versionsnamen und geben Sie einen Namen für die Dokumentversion ein. 3. Wählen Sie für Anwendungskonfigurationsschema entweder das JSON-Schema aus der Liste aus oder wählen Sie Schema erstellen. Wenn Sie Schema erstellen wählen, öffnet Systems Manager die Seite Schema erstellen. Geben Sie die Schemadetails ein und wählen Sie dann Anwendungskonfigurationsschema erstellen. 4. Wählen Sie im Abschnitt Content (Inhalt) entweder YAML oder JSON aus und geben Sie dann die Konfigurationsdaten in das Feld ein.

11. Wählen Sie im Abschnitt Servicerolle die Option Neue Servicerolle aus, um die IAM-Rolle zu AWS AppConfig erstellen, die den Zugriff auf die Konfigurationsdaten ermöglicht. AWS AppConfig füllt das Feld Rollenname automatisch auf der Grundlage des zuvor eingegebenen Namens aus. Oder wählen Sie Bestehende Servicerolle aus. Wählen Sie die Rolle in der Liste Role ARN (ARN der Rolle) aus.
12. Wählen Sie optional auf der Seite „Validatoren hinzufügen“ entweder JSON-Schema oder. AWS Lambda Wenn Sie JSON Schema (JSON-Schema) auswählen, geben Sie das JSON-Schema in das Feld ein. Wenn Sie AWS Lambda auswählen, wählen Sie die Funktion „Amazon Resource Name (ARN)“ und die Version aus der Liste aus.

> **Important**
>
> Konfigurationsdaten, die in SSM-Dokumenten gespeichert sind, müssen anhand eines zugehörigen JSON-Schemas validiert werden, bevor Sie die Konfiguration dem System hinzufügen können. SSM-Parameter erfordern keine Validierungsmethode,

wir empfehlen jedoch, eine Validierungsprüfung für neue oder aktualisierte SSM-Parameterkonfigurationen zu erstellen, indem Sie AWS Lambda

13. Wählen Sie Weiter aus.
14. Wählen Sie auf der Seite Überprüfen und speichern die Option Speichern und mit der Bereitstellung fortfahren aus.

Important

Wenn Sie ein Konfigurationsprofil für erstellt haben AWS CodePipeline, müssen Sie eine Pipeline erstellen CodePipeline , in der Sie AWS AppConfig als Bereitstellungsanbieter angeben. Sie müssen keine Leistung erbringenBereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig. Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschriebenKonfigurationsdaten werden ohne AWS AppConfig Agent abgerufen. Informationen zum Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter angegeben wird, finden Sie unter Tutorial: Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter verwendet wird im AWS CodePipeline Benutzerhandbuch.

Fahren Sie mit Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig fort.

Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie das AWS CLI (unter Linux oder Windows) verwenden oder AWS -Tools für PowerShell ein AWS AppConfig Freiform-Konfigurationsprofil erstellen. Wenn Sie möchten, können Sie AWS CloudShell damit die unten aufgeführten Befehle ausführen. Weitere Informationen finden Sie unter Was ist AWS CloudShell? im AWS CloudShell -Benutzerhandbuch.

Note

Für Freiformkonfigurationen, die im AWS AppConfig gehosteten Konfigurationsspeicher gehostet werden, geben Sie als `hosted` Standort-URI an.

Um ein Konfigurationsprofil mit dem zu erstellen AWS CLI

1. Öffnen Sie das AWS CLI.

2. Führen Sie den folgenden Befehl aus, um ein Freiform-Konfigurationsprofil zu erstellen.

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

Notieren Sie die folgenden wichtigen Informationen.

- Wenn Sie ein Konfigurationsprofil für erstellt haben AWS CodePipeline, müssen Sie eine Pipeline erstellen CodePipeline , in der Sie AWS AppConfig als Bereitstellungsanbieter angeben. Sie müssen keine Leistung erbringenBereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig. Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschriebenKonfigurationsdaten werden ohne AWS AppConfig Agent abgerufen. Informationen zum Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter angegeben wird, finden Sie unter Tutorial: Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter verwendet wird im AWS CodePipeline Benutzerhandbuch.
- Wenn Sie eine Konfiguration im AWS AppConfig gehosteten Konfigurationsspeicher erstellt haben, können Sie mithilfe der CreateHostedConfigurationVersionAPI-Operationen neue Versionen der Konfiguration erstellen. AWS CLI Einzelheiten und Beispielbefehle für diesen API-Vorgang finden Sie create-hosted-configuration-versionin der AWS CLI Befehlsreferenz.

Fahren Sie mit Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig fort.

Erstellen eines Konfigurationsprofils für nicht-native Datenquellen

AWS AppConfig unterstützt die Bereitstellung von Konfigurationsdaten aus fast jedem Datenspeicher. AWS AppConfig Unterstützt nativ die Bereitstellung von Konfigurationsdaten, die in den folgenden Diensten gespeichert sind:

- Der AWS AppConfig gehostete Konfigurationsspeicher
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager Parameterspeicher
- Systems Manager Dokumentenspeicher
- AWS CodePipeline

Wenn Ihre Konfigurationsdaten an einem Ort gespeichert sind, der nicht nativ von unterstützt wird AWS AppConfig, können Sie eine AWS AppConfig Erweiterung erstellen, um Ihre Daten aus der Quelle abzurufen. Mithilfe einer AWS AppConfig Erweiterung können Sie beispielsweise Konfigurationsdaten abrufen, die in Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB (DynamoDB),, oder einem lokalen Repository gespeichert sind GitHub GitLab, um nur einige zu nennen. Durch die Implementierung einer Erweiterung können Sie die AWS AppConfig Sicherheit und die DevOps Verbesserungen für Ihre Anwendungen und Ihre Computerumgebung nutzen. Sie können diese Methode auch verwenden, wenn Sie Konfigurationsdaten aus älteren Systemen in diese migrieren AWS AppConfig.

Das Erstellen eines Konfigurationsprofils für Datenquellen, die nicht nativ unterstützt werden, AWS AppConfig umfasst die folgenden Prozesse oder Aktionen:

1. Erstellen Sie eine AWS Lambda Funktion, die Daten aus Ihrer Datenquelle abruft. Solange eine Lambda-Funktion auf die Datenquelle zugreifen kann, kann Ihre AWS AppConfig Erweiterung die Daten abrufen.
2. Erstellen Sie eine benutzerdefinierte AWS AppConfig Erweiterung, die Ihre Lambda-Funktion aufruft. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.
3. Erstellen Sie ein AWS AppConfig formloses Konfigurationsprofil. Erstellen Sie insbesondere ein Konfigurationsprofil, das die AWS AppConfig gehostete Konfigurationsdefinition verwendet. Das Konfigurationsprofil fungiert als temporärer Datenspeicher, nachdem Ihre Lambda-Funktion Ihre Konfiguration aus Ihrer Quelle abgerufen hat. Ihre Anwendung ruft die Konfigurationsdaten aus dem AWS AppConfig gehosteten Konfigurationsspeicher ab. Weitere Informationen finden Sie unter Erstellen Sie ein frei formatiertes Konfigurationsprofil in AWS AppConfig.
4. Erstellen Sie eine Erweiterungszuordnung, die mithilfe des `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` Aktionspunkts ausgelöst wird. Weitere Informationen finden Sie unter Schritt 4: Erstellen Sie eine Erweiterungszuordnung für eine benutzerdefinierte Erweiterung AWS AppConfig.

Wenn Ihre Anwendung nach der Konfiguration eine neue Version der Konfigurationsdaten anfordert, ruft Lambda Ihre Konfigurationsdaten ab und zieht sie in das Konfigurationsprofil. AWS AppConfig speichert dann das Konfigurationsprofil und Ihre Drittanbieterdaten.

Wenn Sie bereit sind, können Sie das Konfigurationsprofil wie jede andere Art von Konfigurationsdaten für Ihre Anwendungen bereitstellen.

> **Note**
>
> Sie können wählen, ob Drittanbieterdaten entsprechend den vorhandenen Konfigurationsdaten eingefügt werden sollen oder dass der gesamte Inhalt der Konfigurationsdaten nur die Drittanbieterdaten enthält. Wenn Sie möchten, dass die Daten mit anderen vorhandenen Daten übereinstimmen, sollte diese Logik Teil der Lambda-Funktion sein, die die Daten aus der Drittanbieterquelle importiert.

Migration AWS AppConfig von veralteten und selbst entwickelten Konfigurationsdiensten

Wenn Sie damit begonnen haben, ältere Konfigurationsdaten oder Feature-Flags in einem anderen System zu verwenden AWS AppConfig und immer noch über ältere Konfigurationsdaten oder Feature-Flags verfügen, können Sie den weiter oben in diesem Thema beschriebenen Prozess verwenden, um von Ihrem alten System auf ein anderes System zu migrieren. AWS AppConfig Sie können eine Erweiterung erstellen, mit der Daten aus Ihrem Altsystem abgerufen und über dieses bereitgestellt werden. AWS AppConfig Auf diese AWS AppConfig Weise erhalten Sie alle Sicherheitsfunktionen und Vorteile, während Sie weiterhin Ihre veralteten Datenspeicher verwenden.

Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig

Nachdem Sie die erforderlichen Artefakte für die Arbeit mit Feature-Flags und Freiform-Konfigurationsdaten erstellt haben, können Sie eine neue Bereitstellung erstellen. Wenn Sie eine neue Bereitstellung erstellen, geben Sie die folgenden Informationen an:

- Eine Anwendungs-ID
- Eine Konfigurationsprofil-ID
- Eine Konfigurationsversion
- Eine Umgebungs-ID, in der Sie die Konfigurationsdaten bereitstellen möchten
- Eine Bereitstellungsstrategie-ID, die definiert, wie schnell die Änderungen wirksam werden sollen
- Eine AWS Key Management Service (AWS KMS) Schlüssel-ID zum Verschlüsseln der Daten mit einem vom Kunden verwalteten Schlüssel.

AWS AppConfig Führt beim Aufrufen der StartDeploymentAPI-Aktion die folgenden Aufgaben aus:

1. Ruft die Konfigurationsdaten mithilfe des Standort-URI im Konfigurationsprofil aus dem zugrunde liegenden Datenspeicher ab.
2. Überprüft mithilfe der Validatoren, die Sie bei der Erstellung Ihres Konfigurationsprofils angegeben haben, dass die Konfigurationsdaten syntaktisch und semantisch korrekt sind.
3. Speichert eine Kopie der Daten im Cache, sodass sie von Ihrer Anwendung abgerufen werden können. Diese zwischengespeicherte Kopie wird als bereitgestellte Daten bezeichnet.

Sie können Situationen vermeiden, in denen die Bereitstellung von Konfigurationsdaten zu Fehlern in Ihrer Anwendung führt, indem Sie eine Kombination aus AWS AppConfig Bereitstellungsstrategien und automatischen Rollbacks auf der Grundlage von CloudWatch Amazon-Alarmen verwenden. Eine Bereitstellungsstrategie ermöglicht es Ihnen, Änderungen an Produktionsumgebungen langsam innerhalb von Minuten oder Stunden zu veröffentlichen. Wenn nach der Konfiguration ein oder mehrere CloudWatch Alarme während einer Bereitstellung in den Alarmstatus wechseln, AWS AppConfig werden Ihre Konfigurationsdaten automatisch auf die vorherige Version zurückgesetzt. Weitere Informationen zu Bereitstellungsstrategien finden Sie unterArbeiten mit Bereitstellungsstrategien. Weitere Informationen zu automatischen Rollbacks finden Sie unterBereitstellungen im Hinblick auf automatisches Rollback überwachen.

Themen

- Arbeiten mit Bereitstellungsstrategien
- Bereitstellen einer Konfiguration
- Bereitstellen von AWS AppConfig Konfigurationen mit CodePipeline
- Konfiguration rückgängig machen

Arbeiten mit Bereitstellungsstrategien

Eine Bereitstellungsstrategie ermöglicht es Ihnen, Änderungen an Produktionsumgebungen langsam innerhalb von Minuten oder Stunden zu veröffentlichen. Eine AWS AppConfig Bereitstellungsstrategie definiert die folgenden wichtigen Aspekte einer Konfigurationsbereitstellung.

<table>
<tr><th>Einstellung</th><th>Beschreibung</th></tr>
<tr><td>Deployment type (Bereitstellungstyp)</td><td>Der Bereitstellungstyp definiert, wie die Konfiguration bereitgestellt oder eingeführt wird. AWS AppConfig unterstützt lineare und exponentielle Bereitstellungstypen.
Linear: AWS AppConfig Verarbeitet bei diesem Typ die Bereitstellung in Schritten des Wachstumsfaktors, der gleichmäßig über die Bereitstellung verteilt ist. Hier ist ein Beispiel für einen Zeitplan für eine 10-stündige Bereitstellung mit einem linearen Wachstum von 20%:<table><tr><th>Verstrichene Zeit</th><th>Fortschritt der Bereitstellung</th></tr><tr><td>0 Stunde</td><td>0%</td></tr><tr><td>2 Stunden</td><td>20 %</td></tr><tr><td>4 Stunden</td><td>40%</td></tr></table></td></tr>
</table>

<table>
<tr><th>Einstellung</th><th>Beschreibung</th></tr>
<tr><td></td><td>
<table>
<tr><th>Verstrichene Zeit</th><th>Fortschritt der Bereitstellung</th></tr>
<tr><td>6 Stunden</td><td>60%</td></tr>
<tr><td>8 Stunden</td><td>80%</td></tr>
<tr><td>10 Stunden</td><td>100 %</td></tr>
</table>

Exponentiell: Für diesen Typ verarbeitet AWS AppConfig die Bereitstellung exponentiell mit der folgenden Formel: <code>G*(2^N)</code>. In dieser Formel ist <code>G</code> der vom Benutzer angegebene Schrittprozentsatz und <code>N</code> die Anzahl der Schritte, bis die Konfiguration für alle Ziele bereitgestellt wird. Wenn Sie beispielsweise einen Wachstumsfaktor von 2 angeben, führt das System die Konfiguration wie folgt aus:

<pre>2*(2^0)
2*(2^1)
2*(2^2)</pre>
Zahlenmäßig ausgedrückt, wird die Bereitstellung wie folgt ausgeführt: 2 % der Ziele, 4 % der Ziele, 8 % der Ziele, Sie wird fortgesetzt, bis die Konfiguration für alle Ziele bereitgestellt wurde.

</td></tr>
</table>

Einstellung	Beschreibung
Schrittprozentsatz (Wachstumsfaktor)	Diese Einstellung gibt den Prozentsatz der Aufrufer an, für den die Bereitstellung bei den einzelnen Schritten ausgeführt werden soll. **Note** Im SDK und in der AWS AppConfig - API-Referenz wird `step percentage` als `growth factor` bezeichnet.
Deployment time (Bereitstellungszeit)	Diese Einstellung gibt einen Zeitraum an, in dem die AWS AppConfig Bereitstellung auf Hosts erfolgt. Dies ist kein Timeoutwert. Es ist ein Zeitfenster, in dem die Bereitstellung in Intervallen verarbeitet wird.
Bake time (Bake-Zeit)	Diese Einstellung legt fest, wie lange CloudWatch Amazon-Alarme nach der Bereitstellung der Konfiguration auf 100% ihrer Ziele AWS AppConfig überwacht werden, bevor die Bereitstellung als abgeschlossen betrachtet wird. Wird während dieser Zeit ein Alarm ausgelöst, setzt AWS AppConfig die Bereitstellung zurück. Sie müssen die Berechtigungen für das AWS AppConfig Rollback auf der Grundlage von CloudWatch Alarmen konfigurieren. Weitere Informationen finden Sie unter (Empfohlen) Konfigurieren Sie die Berechtigungen für das automatische Rollback.

Sie können eine vordefinierte Strategie auswählen, die im Lieferumfang enthalten ist, AWS AppConfig oder Ihre eigene Strategie erstellen.

Themen

- Verwenden vordefinierter Bereitstellungsstrategien
- Erstellen einer Bereitstellungsstrategie

Verwenden vordefinierter Bereitstellungsstrategien

AWS AppConfig umfasst vordefinierte Bereitstellungsstrategien, mit denen Sie eine Konfiguration schnell bereitstellen können. Anstatt eigene Strategien zu erstellen, können Sie beim Bereitstellen einer Konfiguration eine der folgenden Strategien auswählen.

Bereitstellungsstrategie	Beschreibung
AppConfig. Linear 20 6 Minuten PercentEvery	AWS empfohlen: Bei dieser Strategie wird die Konfiguration alle sechs Minuten auf 20% aller Ziele bereitgestellt, was einer 30-minütigen Bereitstellung entspricht. Das System überwacht 30 Minuten lang, CloudWatch ob Amazon-Alarme vorliegen. Wenn in dieser Zeit keine Alarme empfangen werden, ist die Bereitstellung abgeschlossen. Wenn während dieser Zeit ein Alarm ausgelöst wird, AWS AppConfig wird die Bereitstellung rückgängig gemacht. Wir empfehlen, diese Strategie für Produktionsbereitstellungen zu verwenden, da sie den AWS bewährten Methoden entspricht und aufgrund ihrer langen Dauer und Backzeit zusätzliche Aufmerksamkeit auf die Sicherheit bei der Bereitstellung legt.
AppConfig. Kanarische 10 Prozent 20 Minuten	AWS empfohlen: Diese Strategie verarbeitet die Bereitstellung exponentiell mit einem Wachstumsfaktor von 10 % über 20 Minuten. Das System überwacht 10 Minuten lang, ob CloudWatch Alarme vorliegen. Wenn in dieser Zeit keine Alarme

Bereitstellungsstrategie	Beschreibung
	empfangen werden, ist die Bereitstellung abgeschlossen. Wenn während dieser Zeit ein Alarm ausgelöst wird, AWS AppConfig wird die Bereitstellung rückgängig gemacht. Wir empfehlen, diese Strategie für Produktio nsbereitstellungen zu verwenden, da sie den AWS bewährten Methoden für Konfigura tionsbereitstellungen entspricht.
AppConfig.AllAtOnce	Schnell: Diese Strategie stellt die Konfiguration sofort für alle Ziele bereit. Das System überwacht 10 Minuten lang, ob CloudWatch Alarme vorliegen . Wenn in dieser Zeit keine Alarme empfangen werden, ist die Bereitstellung abgeschlossen. Wird während dieser Zeit ein Alarm ausgelöst, setzt AWS AppConfig die Bereitstellung zurück.
AppConfig. Linear 50 30 Sekunden PercentEv ery	Testen/Vorführung: Diese Strategie stellt die Konfiguration auf der Hälfte aller Ziele alle 30 Sekunden für eine Bereitstellung von einer Minute bereit. Das System überwacht 1 Minute lang, CloudWatch ob Amazon-Alarme vorliegen. Wenn in dieser Zeit keine Alarme empfangen werden, ist die Bereitstellung abgeschlossen. Wenn während dieser Zeit ein Alarm ausgelöst wird, AWS AppConfig wird die Bereitstellung rückgängig gemacht. Wir empfehlen, diese Strategie aufgrund der kurzen Dauer und Bake-Zeit nur für Test- oder Demonstrationszwecke zu verwenden.

Erstellen einer Bereitstellungsstrategie

Wenn Sie keine der vordefinierten Bereitstellungsstrategien verwenden möchten, können Sie Ihre eigene erstellen. Sie können maximal 20 Bereitstellungsstrategien erstellen. Beim Bereitstellen einer Konfiguration können Sie die für die Anwendung und Umgebung am besten geeignete Bereitstellungsstrategie auswählen.

Erstellen einer AWS AppConfig Bereitstellungsstrategie (Konsole)

Gehen Sie wie folgt vor, um mithilfe der AWS Systems Manager Konsole eine AWS AppConfig Bereitstellungsstrategie zu erstellen.

So erstellen Sie eine Bereitstellungsstrategie

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Deployment Strategies und anschließend Create Deployment Strategy aus.
3. Geben Sie unter Name einen Namen für die Bereitstellungsstrategie ein.
4. Geben Sie unter Description (Beschreibung) Informationen zur Bereitstellungsstrategie ein.
5. Wählen Sie unter Deployment type (Bereitstellungstyp) einen Typ aus.
6. Wählen Sie unter Step percentage (Schrittprozentsatz) den Prozentsatz der Aufrufer aus, für den die Bereitstellung bei den einzelnen Schritten der Bereitstellung ausgeführt werden soll.
7. Geben Sie unter Deployment time (Bereitstellungszeit) die Gesamtdauer der Bereitstellung in Minuten oder Stunden ein.
8. Geben Sie unter Backzeit die Gesamtzeit in Minuten oder Stunden ein, die für die Überwachung von CloudWatch Amazon-Alarmen benötigt wird, bevor Sie mit dem nächsten Schritt einer Bereitstellung fortfahren oder die Bereitstellung als abgeschlossen betrachten.
9. Geben Sie im Abschnitt Tags einen Schlüssel und einen optionalen Wert ein. Sie können maximal 50 Tags für eine Ressource angeben.
10. Wählen Sie Create deployment strategy (Bereitstellungsstrategie erstellen) aus.

> **Important**
>
> Wenn Sie ein Konfigurationsprofil für erstellt haben AWS CodePipeline, müssen Sie eine Pipeline erstellen CodePipeline , in der Sie AWS AppConfig als Bereitstellungsanbieter

angeben. Sie müssen keine Leistung erbringenBereitstellen einer Konfiguration. Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschriebenKonfigurationsdaten werden ohne AWS AppConfig Agent abgerufen. Informationen zum Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter angegeben wird, finden Sie unter Tutorial: Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter verwendet wird im AWS CodePipeline Benutzerhandbuch.

Fahren Sie mit Bereitstellen einer Konfiguration fort.

Erstellen einer AWS AppConfig Bereitstellungsstrategie (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie die AWS CLI (unter Linux oder Windows) verwenden oder AWS -Tools für PowerShell eine AWS AppConfig Bereitstellungsstrategie erstellen.

So erstellen Sie Schritt für Schritt eine Bereitstellungsstrategie

1. Öffnen Sie das AWS CLI.
2. Führen Sie den folgenden Befehl aus, um eine Bereitstellungsstrategie zu erstellen.

   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
```

Das System gibt unter anderem folgende Informationen zurück

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
```

Bereitstellen einer Konfiguration

Nachdem Sie die erforderlichen Artefakte für die Arbeit mit Feature-Flags und Freiform-Konfigurationsdaten erstellt haben, können Sie mit dem AWS Management Console AWS CLI, oder dem SDK eine neue Bereitstellung erstellen. Beim Starten einer Bereitstellung in wird der AWS AppConfig StartDeploymentAPI-Vorgang aufgerufen. Dieser Aufruf umfasst die Angabe IDs der AWS AppConfig Anwendung, der Umgebung, des Konfigurationsprofils und (optional) der Version der Konfigurationsdaten, die bereitgestellt werden sollen. Der Aufruf enthält auch die ID der zu verwendenden Bereitstellungsstrategie, die bestimmt, wie die Konfigurationsdaten bereitgestellt werden.

Wenn Sie Geheimnisse einsetzen AWS Secrets Manager, die in Objekten von Amazon Simple Storage Service (Amazon S3) gespeichert sind, die mit einem vom Kunden verwalteten Schlüssel verschlüsselt sind, oder sichere Zeichenkettenparameter, die im AWS Systems Manager Parameter Store gespeichert sind und mit einem vom Kunden verwalteten Schlüssel verschlüsselt sind, müssen Sie einen Wert für den `KmsKeyIdentifier` Parameter angeben. Wenn Ihre Konfiguration nicht verschlüsselt oder mit einem verschlüsselt ist Von AWS verwalteter Schlüssel, ist die Angabe eines Werts für den `KmsKeyIdentifier` Parameter nicht erforderlich.

> **Note**
>
> Bei dem Wert, den Sie angeben, `KmsKeyIdentifier` muss es sich um einen vom Kunden verwalteten Schlüssel handeln. Dabei muss es sich nicht um denselben Schlüssel handeln, den Sie zum Verschlüsseln Ihrer Konfiguration verwendet haben.

Wenn Sie eine Bereitstellung mit einem startenKmsKeyIdentifier, muss die Ihrem AWS Identity and Access Management (IAM-) Principal zugeordnete Berechtigungsrichtlinie den kms:GenerateDataKey Vorgang zulassen.

AWS AppConfig überwacht die Verteilung an alle Hosts und meldet den Status. Wenn eine Verteilung fehlschlägt, wird AWS AppConfig die Konfiguration zurückgesetzt.

> **Note**
>
> Sie können jeweils nur eine Konfiguration in einer Umgebung bereitstellen. Sie können jedoch jeweils eine Konfiguration gleichzeitig in verschiedenen Umgebungen bereitstellen.

Stellen Sie eine Konfiguration bereit (Konsole)

Gehen Sie wie folgt vor, um eine AWS AppConfig Konfiguration mithilfe der AWS Systems Manager Konsole bereitzustellen.

So stellen Sie eine Konfiguration über die Konsole bereit

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Anwendungen und dann eine Anwendung aus, in der Sie sie erstellt haben. Erstellen Sie einen Namespace für Ihre Anwendung in AWS AppConfig
3. Füllen Sie auf der Registerkarte Umgebungen das Optionsfeld für eine Umgebung aus, und wählen Sie dann Details anzeigen aus.
4. Klicken Sie auf Start deployment (Bereitstellung starten).
5. Wählen Sie unter Configuration (Konfiguration) eine Konfiguration in der Liste aus.
6. Verwenden Sie je nach Quelle Ihrer Konfiguration die Versionsliste, um die Version auszuwählen, die Sie bereitstellen möchten.
7. Wählen Sie unter Deployment strategy (Bereitstellungsstrategie) eine Strategie in der Liste aus.
8. (Optional) Geben Sie unter Bereitstellungsbeschreibung eine Beschreibung ein.
9. Wählen Sie für zusätzliche Verschlüsselungsoptionen einen AWS Key Management Service Schlüssel aus der Liste aus.

10. (Optional) Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen optionalen Wert ein. Sie können maximal 50 Tags für eine Ressource angeben.
11. Klicken Sie auf Start deployment (Bereitstellung starten).

Stellen Sie eine Konfiguration bereit (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie die AWS CLI (unter Linux oder Windows) verwenden oder AWS -Tools für PowerShell eine AWS AppConfig Konfiguration bereitstellen.

So stellen Sie eine Konfiguration Schritt für Schritt bereit

1. Öffnen Sie das AWS CLI.
2. Führen Sie den folgenden Befehl aus, um eine Konfiguration bereitzustellen.

   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
```

Das System gibt unter anderem folgende Informationen zurück

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
```

Bereitstellen von AWS AppConfig Konfigurationen mit CodePipeline

AWS AppConfig ist eine integrierte Bereitstellungsaktion für AWS CodePipeline (CodePipeline). CodePipeline ist ein vollständig verwalteter Continuous Delivery-Service, der Sie bei der

Automatisierung Ihrer Release-Pipelines für schnelle und zuverlässige Anwendungs- und Infrastrukturupdates unterstützt. CodePipeline automatisiert die Erstellungs-, Test- und Bereitstellungsphasen Ihres Release-Prozesses bei jeder Codeänderung auf der Grundlage des von Ihnen definierten Release-Modells. Weitere Informationen finden Sie unter Was ist AWS CodePipeline?

Die Integration von AWS AppConfig mit CodePipeline bietet die folgenden Vorteile:

- Kunden, die früher CodePipeline die Orchestrierung verwaltet haben, verfügen nun über eine einfache Möglichkeit, Konfigurationsänderungen an ihren Anwendungen vorzunehmen, ohne ihre gesamte Codebasis bereitstellen zu müssen.
- Kunden, die das System AWS AppConfig zur Verwaltung von Konfigurationsbereitstellungen verwenden möchten, aber nur eingeschränkt zur Verfügung stehen, weil ihr aktueller Code oder Konfigurationsspeicher AWS AppConfig nicht unterstützt wird, stehen jetzt zusätzliche Optionen zur Verfügung. CodePipeline unterstützt AWS CodeCommit GitHub, und BitBucket (um nur einige zu nennen).

> **Note**
>
> AWS AppConfig Die Integration mit CodePipeline wird nur dort unterstützt AWS-Regionen , wo sie verfügbar CodePipeline ist.

Wie funktioniert die Integration

Sie beginnen mit der Einrichtung und Konfiguration CodePipeline. Dazu gehört das Hinzufügen Ihrer Konfiguration zu einem CodePipeline Codespeicher mit Unterstützung. Als Nächstes richten Sie Ihre AWS AppConfig Umgebung ein, indem Sie die folgenden Aufgaben ausführen:

- Erstellen Sie einen Namespace und ein Konfigurationsprofil
- Wählen Sie eine vordefinierte Bereitstellungsstrategie oder erstellen Sie Ihre eigene

Nachdem Sie diese Aufgaben abgeschlossen haben, erstellen Sie eine Pipeline CodePipeline , in der Sie AWS AppConfig als Bereitstellungsanbieter angeben. Anschließend können Sie eine Änderung an Ihrer Konfiguration vornehmen und diese in Ihren CodePipeline Codespeicher hochladen. Durch das Hochladen der neuen Konfiguration wird automatisch eine neue Bereitstellung in CodePipeline gestartet. Nach Abschluss der Bereitstellung können Sie Ihre Änderungen überprüfen. Informationen

zum Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter angibt, finden Sie im AWS CodePipeline Benutzerhandbuch unter Tutorial: Erstellen einer Pipeline, die AWS AppConfig als Bereitstellungsanbieter verwendet wird.

Konfiguration rückgängig machen

Während einer Bereitstellung können Sie Situationen vermeiden, in denen fehlerhafte oder falsche Konfigurationsdaten Fehler in Ihrer Anwendung verursachen, indem Sie automatische Rollbacks verwenden (wenn während einer Bereitstellung ein Alarm ausgelöst wird) oder indem Sie die Konfigurationsdaten auf die vorherige Version zurücksetzen (wenn eine Bereitstellung erfolgreich abgeschlossen wurde).

Für automatische Rollbacks können Sie eine Kombination aus AWS AppConfig Bereitstellungsstrategien und CloudWatch Amazon-Alarmen verwenden. Wenn nach der Konfiguration ein oder mehrere CloudWatch Alarme während einer Bereitstellung in den `ALARM` Status wechseln, AWS AppConfig werden Ihre Konfigurationsdaten automatisch auf die vorherige Version zurückgesetzt, wodurch Anwendungsausfälle oder -fehler vermieden werden. Um zu beginnen, sehen Sie sich (Empfohlen) Konfigurieren Sie die Berechtigungen für das automatische Rollback an.

> **Note**
>
> Sie können eine Konfiguration auch rückgängig machen, indem Sie den StopDeploymentAPI-Vorgang aufrufen, während eine Bereitstellung noch läuft.

Unterstützt bei erfolgreich abgeschlossenen Bereitstellungen AWS AppConfig auch das Zurücksetzen von Konfigurationsdaten auf eine frühere Version, indem der `AllowRevert` Parameter zusammen mit dem StopDeploymentAPI-Vorgang verwendet wird. Für einige Kunden garantiert das Zurücksetzen auf eine vorherige Konfiguration nach einer erfolgreichen Bereitstellung, dass die Daten dieselben sind wie vor der Bereitstellung. Beim Zurücksetzen werden auch Alarmanzeigen ignoriert, wodurch verhindert werden kann, dass ein Rollforward während eines Anwendungsnotfalls fortgesetzt wird.

> **Important**
>
> Wenn Sie `StopDeployment` mit aktiviertem `AllowRevert` Parameter aufrufen, AWS AppConfig wird die Bereitstellung nur dann rückgängig gemacht, wenn die Bereitstellung

innerhalb der letzten 72 Stunden erfolgreich war. Nach 72 Stunden kann die Bereitstellung nicht mehr rückgängig gemacht werden. Sie müssen eine neue Bereitstellung erstellen.

Im Folgenden finden Sie eine Aufschlüsselung der `StopDeployment` Funktionen anhand verschiedener Situationen.

1. Wenn bei einer Bereitstellung, die gerade ausgeführt wird, aufgerufen `ROLLED_BACK` wird, `StopDeployment` lautet der resultierende Bereitstellungsstatus.
2. Wenn `StopDeployment` (with`AllowRevert`) bei einem Einsatz in Bearbeitung aufgerufen wird, lautet der resultierende Bereitstellungsstatus. `ROLLED_BACK`
3. Wenn `StopDeployment` bei einer abgeschlossenen Bereitstellung aufgerufen wird, `BadRequestException` wird a ausgelöst.
4. Wenn `StopDeployment` (with`AllowRevert`) bei einer abgeschlossenen Bereitstellung aufgerufen wird, lautet der resultierende Bereitstellungsstatus`REVERTED`.
5. Wenn `StopDeployment` (with`AllowRevert`) bei einer abgeschlossenen Bereitstellung nach 72 Stunden aufgerufen wird, `BadRequestException` wird a ausgelöst.

Sie können das verwenden AWS CLI , um die StopDeploymentOperation mit dem `AllowRevert` Parameter aufzurufen. Hier ist ein AWS CLI Beispielbefehl, der den `AllowRevert` Parameter enthält.

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

Feature-Flags und Konfigurationsdaten abrufen in AWS AppConfig

Ihre Anwendung ruft Feature-Flags und Freiform-Konfigurationsdaten ab, indem sie mithilfe des AWS AppConfig Datendienstes eine Konfigurationssitzung einrichtet. Es wird empfohlen, den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten zu verwenden. Der Agent (oder die AWS AppConfig Agent Lambda-Erweiterung für Lambda-Rechenumgebungen) verwaltet in Ihrem Namen eine Reihe von API-Aufrufen und Sitzungstoken. Auf oberster Ebene funktioniert der Prozess wie folgt:

1. Sie konfigurieren den AWS AppConfig Agenten als lokalen Host und lassen den Agenten AWS AppConfig nach Konfigurationsupdates fragen.
2. Der Agent ruft die StartConfigurationSessionund GetLatestConfigurationAPI-Aktionen auf und speichert Ihre Konfigurationsdaten lokal im Cache.
3. Um die Daten abzurufen, sendet Ihre Anwendung einen HTTP-Aufruf an den Localhost-Server. AWS AppConfig Der Agent unterstützt mehrere Anwendungsfälle, wie unter beschriebenWie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten.

Wenn Sie möchten, können Sie diese API-Aktionen manuell aufrufen, um eine Konfiguration abzurufen. Der API-Prozess funktioniert wie folgt:

1. Ihre Anwendung richtet mithilfe der `StartConfigurationSession` API-Aktion eine Konfigurationssitzung ein. Der Client Ihrer Sitzung ruft dann regelmäßig auf, um `GetLatestConfiguration` nach den neuesten verfügbaren Daten zu suchen und diese abzurufen.
2. Beim Aufrufen `StartConfigurationSession` sendet Ihr Code Kennungen (ID oder Name) einer AWS AppConfig Anwendung, einer Umgebung und eines Konfigurationsprofils, das von der Sitzung verfolgt wird.
3. AWS AppConfig Stellt als Antwort ein, das `InitialConfigurationToken` an den Client der Sitzung übergeben und verwendet werden soll, wenn er diese Sitzung `GetLatestConfiguration` zum ersten Mal aufruft.
4. Beim Aufrufen `GetLatestConfiguration` sendet Ihr Client-Code den neuesten `ConfigurationToken` Wert, den er hat, und empfängt als Antwort:

- `NextPollConfigurationToken`: der `ConfigurationToken` Wert, der beim nächsten Aufruf von verwendet werden soll`GetLatestConfiguration`.
- Die Konfiguration: Die neuesten Daten, die für die Sitzung vorgesehen sind. Dies kann leer sein, wenn der Client bereits über die neueste Version der Konfiguration verfügt.

Inhalt

- Was ist AWS AppConfig Agent?
- Wie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten
- AWS AppConfig Überlegungen zur Nutzung von Browsern und Mobilgeräten
- Konfigurationsdaten werden ohne AWS AppConfig Agent abgerufen

Was ist AWS AppConfig Agent?

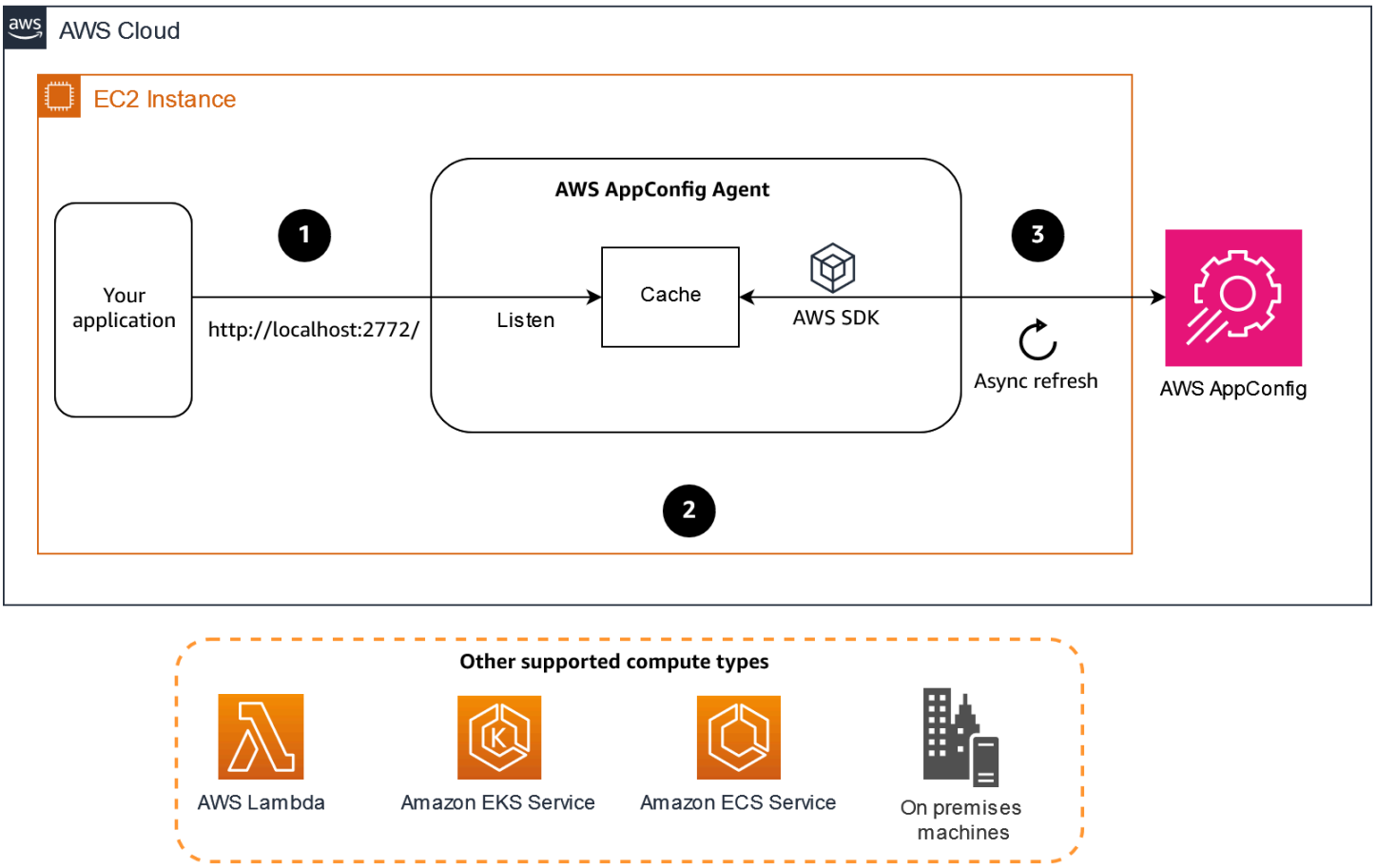
AWS AppConfig Agent ist ein von Amazon entwickelter und verwalteter Prozess zum Abrufen von Konfigurationsdaten von. AWS AppConfig Mit dem Agenten können Sie Konfigurationsdaten lokal zwischenspeichern und den AWS AppConfig Datenebenendienst asynchron nach Aktualisierungen abfragen. Dieser caching/polling Prozess stellt sicher, dass Ihre Konfigurationsdaten immer für Ihre Anwendung verfügbar sind, und minimiert gleichzeitig Latenz und Kosten. Der Agent ist nicht die einzige Möglichkeit, Konfigurationsdaten abzurufen AWS AppConfig, aber er ist die empfohlene Methode. Der Agent verbessert die Anwendungsverarbeitung und -verwaltung auf folgende Weise:

- Der Agent ruft in Ihrem Namen AWS AppConfig an, indem er einen AWS Identity and Access Management (IAM-) Prinzipal verwendet und einen lokalen Cache mit Konfigurationsdaten verwaltet. Durch das Abrufen von Konfigurationsdaten aus dem lokalen Cache benötigt Ihre Anwendung weniger Codeaktualisierungen zur Verwaltung von Konfigurationsdaten, ruft Konfigurationsdaten in Millisekunden ab und ist nicht von Netzwerkproblemen betroffen, die Aufrufe solcher Daten unterbrechen können.
- Der Agent bietet eine native Oberfläche für das Abrufen und Auflösen von Feature-Flags. AWS AppConfig
- Der sofort einsatzbereite Agent bietet bewährte Methoden für Caching-Strategien, Abfrageintervalle und die Verfügbarkeit lokaler Konfigurationsdaten und verfolgt gleichzeitig die für nachfolgende Serviceanfragen benötigten Konfigurationstoken.
- Während der Ausführung im Hintergrund fragt der Agent den AWS AppConfig Datenebenendienst regelmäßig nach Aktualisierungen der Konfigurationsdaten ab. Ihre Anwendung kann die Daten

abrufen, indem sie über Port 2772 (ein anpassbarer Standard-Portwert) eine Verbindung zu localhost herstellt und HTTP GET aufruft, um die Daten abzurufen.

> **Note**
>
> AWS AppConfig Der Agent speichert Daten im Cache, wenn der Dienst Ihre Konfigurationsdaten zum ersten Mal abruft. Aus diesem Grund ist der erste Aufruf zum Abrufen von Daten langsamer als nachfolgende Aufrufe.

Das folgende Diagramm zeigt, wie AWS AppConfig Agent funktioniert.



1. Ihre Anwendung fordert Konfigurationsdaten vom Agenten an.
2. Der Agent gibt Daten aus einem In-Memory-Cache zurück.

3. Der Agent fragt den AWS AppConfig Dienst asynchron in einem vordefinierten Rhythmus nach den neuesten Konfigurationsdaten ab. Die neuesten Konfigurationsdaten werden immer in einem Cache im Arbeitsspeicher gespeichert.

Wie benutzt man den AWS AppConfig Agenten zum Abrufen von Konfigurationsdaten

Der AWS AppConfig Agent ist die empfohlene Methode zum Abrufen von AWS AppConfig Feature-Flags oder Freiform-Konfigurationsdaten. Der Agent wird auf allen Arten von AWS Compute unterstützt, einschließlich Amazon EC2, Amazon ECS, Amazon EKS und Lambda. Nachdem Sie die anfängliche Einrichtung des Agenten abgeschlossen haben, ist es einfacher, den Agenten zum Abrufen von Konfigurationsdaten zu verwenden, als direkt anzurufen AWS AppConfig APIs. Der Agent implementiert automatisch bewährte Methoden und kann Ihre Nutzungskosten senken, AWS AppConfig da weniger API-Aufrufe zum Abrufen von Konfigurationen erforderlich sind.

Topics

- Verwenden Sie den AWS AppConfig Agenten mit AWS Lambda
- AWS AppConfig Agent mit Amazon EC2 - und lokalen Maschinen verwenden
- AWS AppConfig Agent mit Amazon ECS und Amazon EKS verwenden
- Feature-Flags für grundlegende und variantenreiche Funktionen werden abgerufen
- Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren
  - AWS AppConfig Agent zum Abrufen von Konfigurationen von mehreren Konten konfigurieren
  - Den AWS AppConfig Agenten so konfigurieren, dass er Konfigurationskopien auf die Festplatte schreibt
- Generierung eines Clients mithilfe der OpenAPI-Spezifikation
- Mit dem lokalen Entwicklungsmodus des Agenten AWS AppConfig arbeiten

Verwenden Sie den AWS AppConfig Agenten mit AWS Lambda

Eine AWS Lambda Erweiterung ist ein Begleitprozess, der die Funktionen einer Lambda-Funktion erweitert. Eine Erweiterung kann vor dem Aufruf einer Funktion beginnen, parallel zu einer Funktion ausgeführt werden und nach der Verarbeitung eines Funktionsaufrufs weiterlaufen. Im Wesentlichen ist eine Lambda-Erweiterung wie ein Client, der parallel zu einem Lambda-Aufruf ausgeführt wird.

Dieser parallele Client kann jederzeit während seines Lebenszyklus mit Ihrer Funktion verbunden werden.

Wenn Sie AWS AppConfig Feature-Flags oder andere dynamische Konfigurationsdaten in einer Lambda-Funktion verwenden, empfehlen wir Ihnen, die AWS AppConfig Agent Lambda-Erweiterung als Ebene zu Ihrer Lambda-Funktion hinzuzufügen. Dadurch wird das Aufrufen von Feature-Flags einfacher, und die Erweiterung selbst enthält bewährte Methoden, die die Verwendung vereinfachen und AWS AppConfig gleichzeitig die Kosten senken. Geringere Kosten ergeben sich aus weniger API-Aufrufen des AWS AppConfig Dienstes und kürzeren Verarbeitungszeiten für Lambda-Funktionen. Weitere Informationen zu Lambda-Erweiterungen finden Sie unter Lambda-Erweiterungen im AWS Lambda Developer Guide.

> **Note**
>
> AWS AppConfig ist eine Fähigkeit von. AWS Systems Manager AWS AppConfig Die Preisgestaltung basiert auf der Häufigkeit, mit der eine Konfiguration aufgerufen und empfangen wird. Ihre Kosten steigen, wenn Ihr Lambda mehrere Kaltstarts durchführt und häufig neue Konfigurationsdaten abruft.

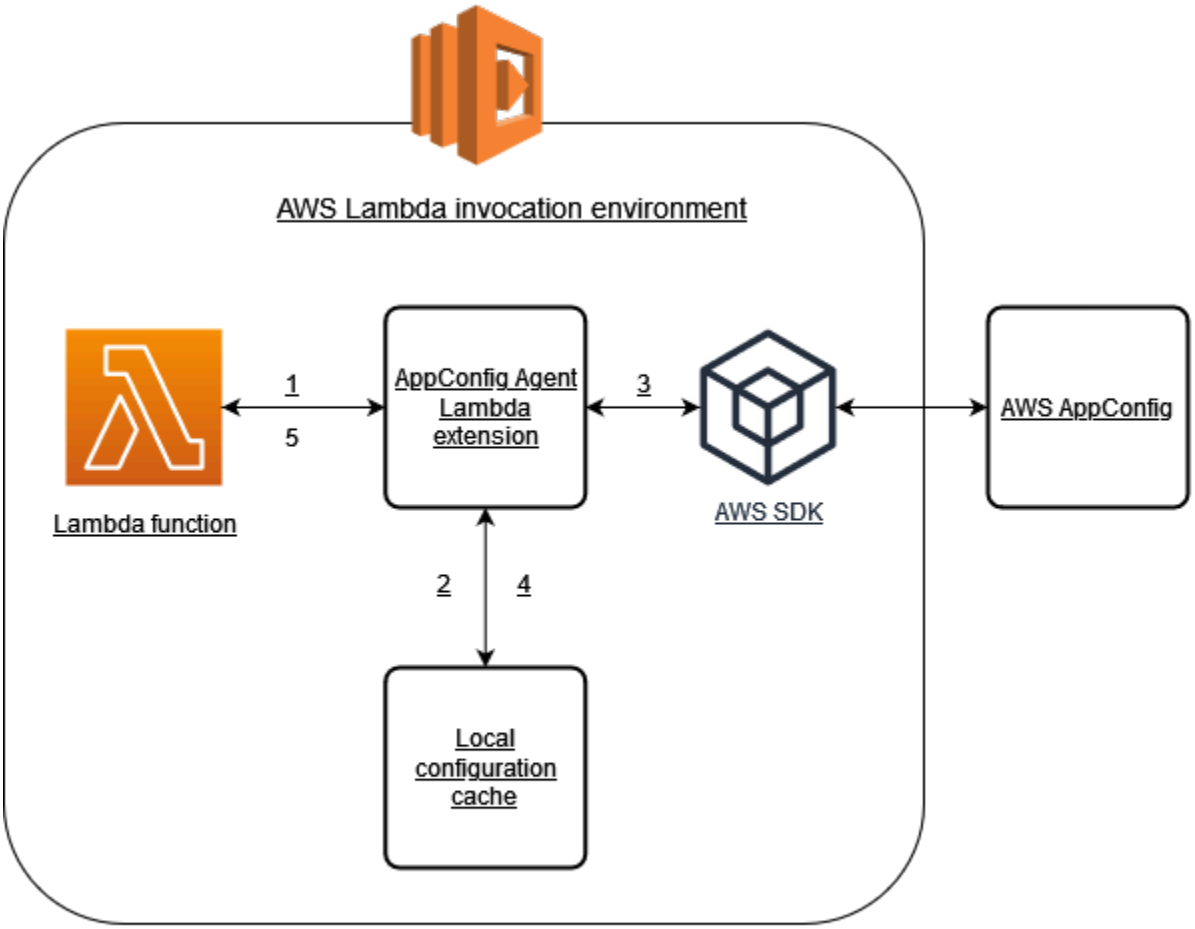
Themen

- Verstehen, wie die AWS AppConfig Agent Lambda-Erweiterung funktioniert
- Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung
- Konfiguration der AWS AppConfig Agent-Lambda-Erweiterung
- Grundlegendes zu den verfügbaren Versionen der AWS AppConfig Agent Lambda-Erweiterung

Verstehen, wie die AWS AppConfig Agent Lambda-Erweiterung funktioniert

Wenn Sie AWS AppConfig Konfigurationen für eine Lambda-Funktion ohne Lambda-Erweiterungen verwalten, müssen Sie Ihre Lambda-Funktion so konfigurieren, dass sie Konfigurationsupdates erhält, indem Sie sie in die StartConfigurationSessionund GetLatestConfigurationAPI-Aktionen integrieren.

Die Integration der AWS AppConfig Agent Lambda-Erweiterung in Ihre Lambda-Funktion vereinfacht diesen Prozess. Die Erweiterung kümmert sich darum, den AWS AppConfig Dienst aufzurufen, einen lokalen Cache mit abgerufenen Daten zu verwalten, die für die nächsten Serviceabrufe benötigten Konfigurationstoken zu verfolgen und regelmäßig im Hintergrund nach Konfigurationsupdates zu suchen. Das folgende Diagramm zeigt, wie es funktioniert.



1. Sie konfigurieren die AWS AppConfig Agent Lambda-Erweiterung als Ebene Ihrer Lambda-Funktion.
2. Um auf ihre Konfigurationsdaten zuzugreifen, ruft Ihre Funktion die AWS AppConfig Erweiterung an einem HTTP-Endpunkt auf, auf dem sie ausgeführt wird. `localhost:2772`
3. Die Erweiterung verwaltet einen lokalen Cache mit den Konfigurationsdaten. Wenn sich die Daten nicht im Cache befinden, ruft die Erweiterung auf, AWS AppConfig um die Konfigurationsdaten abzurufen.
4. Nach dem Empfang der Konfiguration vom Dienst speichert die Erweiterung sie im lokalen Cache und übergibt sie an die Lambda-Funktion.
5. AWS AppConfig Die Agent Lambda-Erweiterung sucht regelmäßig im Hintergrund nach Aktualisierungen Ihrer Konfigurationsdaten. Jedes Mal, wenn Ihre Lambda-Funktion aufgerufen wird, überprüft die Erweiterung die Zeit, die seit dem Abrufen einer Konfiguration vergangen ist.

Wenn die verstrichene Zeit länger als das konfigurierte Abfrageintervall ist, ruft die Erweiterung auf, AWS AppConfig um nach neu bereitgestellten Daten zu suchen, aktualisiert den lokalen Cache, falls eine Änderung vorgenommen wurde, und setzt die verstrichene Zeit zurück.

> **Note**
>
> - Lambda instanziiert separate Instances, die der Gleichzeitigkeitsstufe entsprechen, die Ihre Funktion benötigt. Jede Instance ist isoliert und verwaltet ihren eigenen lokalen Cache Ihrer Konfigurationsdaten. Weitere Informationen zu Lambda-Instanzen und Parallelität finden Sie unter Parallelität für eine Lambda-Funktion verwalten.
> - Wie lange es dauert, bis eine Konfigurationsänderung in einer Lambda-Funktion angezeigt wird, nachdem Sie eine aktualisierte Konfiguration von bereitgestellt haben, hängt von der Bereitstellungsstrategie ab AWS AppConfig, die Sie für die Bereitstellung verwendet haben, und dem Abfrageintervall, das Sie für die Erweiterung konfiguriert haben.

Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung

Um die AWS AppConfig Agent Lambda-Erweiterung verwenden zu können, müssen Sie die Erweiterung zu Ihrem Lambda hinzufügen. Dies kann geschehen, indem Sie die AWS AppConfig Agent Lambda-Erweiterung als Ebene zu Ihrer Lambda-Funktion hinzufügen oder indem Sie die Erweiterung für eine Lambda-Funktion als Container-Image aktivieren.

> **Note**
>
> Die AWS AppConfig Erweiterung ist laufzeitunabhängig und unterstützt alle Laufzeiten.

Bevor Sie beginnen

Gehen Sie wie folgt vor, bevor Sie die AWS AppConfig Agent Lambda-Erweiterung aktivieren:

- Organisieren Sie die Konfigurationen in Ihrer Lambda-Funktion so, dass Sie sie externalisieren können. AWS AppConfig
- Erstellen Sie AWS AppConfig Artefakte und Konfigurationsdaten, einschließlich Feature-Flags oder Freiform-Konfigurationsdaten. Weitere Informationen finden Sie unter Erstellen von Feature-Flags und Freiform-Konfigurationsdaten in AWS AppConfig.

- Fügen Sie der AWS Identity and Access Management (IAM `appconfig:StartConfigurationSession` -`appconfig:GetLatestConfiguration`) Richtlinie, die von der Lambda-Funktionsausführungsrolle verwendet wird, und hinzu. Weitere Informationen finden Sie unter AWS Lambda -Ausführungsrolle im AWS Lambda -Entwicklerhandbuch. Weitere Informationen zu AWS AppConfig Berechtigungen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS AppConfig in der Service Authorization Reference.

Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung mithilfe einer Ebene und eines ARN

Um die AWS AppConfig Agent Lambda-Erweiterung zu verwenden, fügen Sie die Erweiterung als Ebene zu Ihrer Lambda-Funktion hinzu. Informationen zum Hinzufügen einer Ebene zu Ihrer Funktion finden Sie unter Konfiguration von Erweiterungen im AWS Lambda Entwicklerhandbuch. Der Name der Erweiterung in der AWS Lambda Konsole lautet AWS- AppConfig -Extension. Beachten Sie auch, dass Sie beim Hinzufügen der Erweiterung als Ebene zu Ihrem Lambda einen Amazon-Ressourcennamen (ARN) angeben müssen. Wählen Sie einen ARN aus einer der folgenden Listen aus, der der Plattform entspricht und AWS-Region auf der Sie das Lambda erstellt haben.

- x86-64-Plattform
- ARM64plattform

Wenn Sie die Erweiterung testen möchten, bevor Sie sie zu Ihrer Funktion hinzufügen, können Sie anhand des folgenden Codebeispiels überprüfen, ob sie funktioniert.

```
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
```

Um es zu testen, erstellen Sie eine neue Lambda-Funktion für Python, fügen Sie die Erweiterung hinzu und führen Sie dann die Lambda-Funktion aus. Nachdem Sie die Lambda-Funktion ausgeführt haben, gibt die AWS AppConfig Lambda-Funktion die Konfiguration zurück, die Sie für den Pfad http://localhost:2772 angegeben haben. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole im AWS Lambda Entwicklerhandbuch.

Important

Sie können die Protokolldaten für die AWS AppConfig Agent Lambda-Erweiterung in den AWS Lambda Protokollen anzeigen. Protokolleinträgen wird das Präfix vorangestellt. `appconfig agent` Ein Beispiel:

```
[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)
```

Konfiguration der AWS AppConfig Agent-Lambda-Erweiterung

Sie können die Erweiterung konfigurieren, indem Sie die folgenden AWS Lambda Umgebungsvariablen ändern. Weitere Informationen finden Sie unter Verwenden von AWS Lambda Umgebungsvariablen im AWS Lambda Entwicklerhandbuch.

Konfigurationsdaten werden vorab abgerufen

Die Umgebungsvariable `AWS_APPCONFIG_EXTENSION_PREFETCH_LIST` kann die Startzeit Ihrer Funktion verbessern. Wenn die AWS AppConfig Agent Lambda-Erweiterung initialisiert ist, ruft sie die angegebene Konfiguration ab, AWS AppConfig bevor Lambda beginnt, Ihre Funktion zu initialisieren und Ihren Handler aufzurufen. In einigen Fällen sind die Konfigurationsdaten bereits im lokalen Cache verfügbar, bevor Ihre Funktion sie anfordert.

Um die Prefetch-Fähigkeit zu verwenden, setzen Sie den Wert der Umgebungsvariablen auf den Pfad, der Ihren Konfigurationsdaten entspricht. Wenn Ihre Konfiguration beispielsweise einem Anwendungs-, Umgebungs- und Konfigurationsprofil mit den Namen „my_application“, „my_environment“ und „my_configuration_data“ entspricht, wäre der Pfad. `/applications/my_application/environments/my_environment/configurations/my_configuration_data` Sie können mehrere Konfigurationselemente angeben, indem Sie sie als kommagetrennte Liste auflisten (wenn Sie einen Ressourcennamen haben, der ein Komma enthält, verwenden Sie den ID-Wert der Ressource anstelle ihres Namens).

Zugreifen auf Konfigurationsdaten von einem anderen Konto

Die AWS AppConfig Agent Lambda-Erweiterung kann Konfigurationsdaten von einem anderen Konto abrufen, indem sie eine IAM-Rolle angibt, die Berechtigungen für die Daten gewährt. Gehen Sie folgendermaßen vor, um dies einzurichten:

1. Erstellen Sie in dem AWS AppConfig Konto, in dem die Konfigurationsdaten verwaltet werden, eine Rolle mit einer Vertrauensrichtlinie, die dem Konto, auf dem die Lambda-Funktion ausgeführt wird, Zugriff auf die `appconfig:GetLatestConfiguration` Aktionen `appconfig:StartConfigurationSession` und gewährt, zusammen mit den Teil- oder Vollzugriff, die den AWS AppConfig Konfigurationsressourcen ARNs entsprechen.
2. Fügen Sie in dem Konto, auf dem die Lambda-Funktion ausgeführt wird, der Lambda-Funktion die `AWS_APPCONFIG_EXTENSION_ROLE_ARN` Umgebungsvariable mit dem ARN der in Schritt 1 erstellten Rolle hinzu.
3. (Optional) Bei Bedarf kann mithilfe der `AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID` Umgebungsvariablen eine externe ID angegeben werden. In ähnlicher Weise kann ein Sitzungsname mithilfe der `AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME` Umgebungsvariablen konfiguriert werden.

> **Note**
>
> Notieren Sie die folgenden Informationen:
>
> - Die AWS AppConfig Agent Lambda-Erweiterung kann nur Daten von einem Konto abrufen. Wenn Sie eine IAM-Rolle angeben, kann die Erweiterung keine Konfigurationsdaten von dem Konto abrufen, in dem die Lambda-Funktion ausgeführt wird.
> - AWS Lambda protokolliert Informationen über die AWS AppConfig Agent Lambda-Erweiterung und die Lambda-Funktion mithilfe von Amazon CloudWatch Logs.
> - Die folgende Tabelle enthält eine Spalte mit Beispielwerten. Je nach Bildschirmauflösung müssen Sie möglicherweise zum Ende der Tabelle und dann nach rechts scrollen, um die Spalte anzuzeigen.

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTE`	Diese Umgebungsvariable gibt den Port an, auf dem der lokale	2772	2772

Umgebungsvariable	Details	Standardwert	Beispielwerte
`NSION_HTT P_PORT`	HTTP-Server läuft, der die Erweiterung hostet.		
`AWS_APPCO NFIG_EXTE NSION_LOG _LEVEL`	Diese Umgebungs variable gibt den Detaillierungsgrad an, den der Agent protokolliert. Jede Ebene umfasst die aktuelle Ebene und alle höheren Ebenen. Der Wert untersche idet nicht zwischen Groß- und Kleinschr eibung. Die Protokoll ebenen, von den meisten bis hin zu den am wenigsten detaillierten`trace`, lauten wie folgt: `debug`,`info`,`warn`,,`err` undnone. Das `trace` Protokoll enthält detaillierte Informati onen, einschließlich Zeitinformationen, über den Agenten.	info	Spur debug info warnen error tödlich Keine

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS`	Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, die die Erweiterung zum Abrufen von AWS AppConfig Konfigurationen verwendet.	3	3
`AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS`	Diese Umgebungsvariable steuert, wie oft der Agent AWS AppConfig nach aktualisierten Konfigurationsdaten fragt. Sie können eine Anzahl von Sekunden für das Intervall angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: s für Sekunden, m für Minuten und h für Stunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardmäßig Sekunden. Beispielsweise ergeben 60, 60 Sekunden und 1 m dasselbe Abfragein tervall.	45	45 45 s 5m 1 h

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS`	Diese Umgebungsvariable steuert die maximale Zeit (in Millisekunden), für die die Erweiterung AWS AppConfig beim Aktualisieren von Daten im Cache auf eine Antwort wartet. Wenn innerhalb der angegebenen Zeit AWS AppConfig nicht reagiert wird, überspringt die Erweiterung dieses Abfrageintervall und gibt die zuvor aktualisierten zwischengespeicherten Daten zurück.	3000 ms	3000 300 ms 5s

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_PREFETCH_LIST`	Diese Umgebungsvariable gibt die Konfigurationsdaten an, nach denen der Agent fragt, AWS AppConfig sobald er gestartet wird. In einer durch Kommas getrennten Liste können mehrere Konfigurationsbezeichner angegeben werden. Durch das Vorabrufen von Konfigurationsdaten von AWS AppConfig kann die Kaltstartzeit Ihrer Funktion erheblich reduziert werden.	Keine	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Konfiguration 1, ::Konfiguration 2 MyEnv MyApp MyEnv

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_PROXY_HEADERS`	Diese Umgebungsvariable gibt Header an, die für den Proxy erforderlich sind, auf den in der Umgebungsvariablen verwiesen wird. `AWS_APPCONFIG_EXTENSION_PROXY_URL` Der Wert ist eine durch Kommas getrennte Liste von Headern.	Keine	Header: Wert h1: v1, h2: v2
`AWS_APPCONFIG_EXTENSION_PROXY_URL`	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen von der AWS AppConfig Erweiterung zu verwendet werden soll AWS-Services. `HTTPS`und `HTTP` URLs werden unterstützt.	Keine	http://localhost:7474 https://my-proxy.example.com

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_ROLE_ARN`	Diese Umgebungsvariable gibt den ARN der IAM-Rolle an, der einer Rolle entspricht, die von der AWS AppConfig Erweiterung zum Abrufen der Konfiguration übernommen werden sollte.	Keine	arn:aws:iam: :123456789012:role/ MyRole
`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`	Diese Umgebungsvariable gibt die externe ID an, die in Verbindung mit der angenommenen Rolle ARN verwendet werden soll.	Keine	MyExternalId
`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	Keine	AWSAppConfigAgentSession

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_SERVICE_REGION`	Diese Umgebungsvariable gibt eine alternative Region an, die die Erweiterung verwenden soll, um den AWS AppConfig Dienst aufzurufen. Wenn sie nicht definiert ist, verwendet die Erweiterung den Endpunkt in der aktuellen Region.	Keine	us-east-1 eu-west-1
`AWS_APPCONFIG_EXTENSION_MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er zusätzliche konfigurationsspezifische Funktionen wie das Abrufen mehrerer Konten und das Speichern der Konfiguration auf der Festplatte nutzt. Weitere Informationen zu diesen Funktionen finden Sie unter Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren.	Keine	Wenn die Konfiguration als Manifest verwendet AWS AppConfig wird:. `MyApp:MyEnv:MyManifestConfig` Beim Laden des Manifests von der Festplatte: `file:/path/to/manifest.json`

Umgebungsvariable	Details	Standardwert	Beispielwerte
`AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er wartet, bis das Manifest verarbeitet ist, bevor der Start abgeschlossen wird.	true	true false

Grundlegendes zu den verfügbaren Versionen der AWS AppConfig Agent Lambda-Erweiterung

Dieses Thema enthält Informationen zu den Versionen der AWS AppConfig Agent-Lambda-Erweiterung. Die AWS AppConfig Agent Lambda-Erweiterung unterstützt Lambda-Funktionen, die für die Plattformen x86-64 und ARM64 (Graviton2) entwickelt wurden. Damit Ihre Lambda-Funktion ordnungsgemäß funktioniert, muss sie so konfiguriert sein, dass sie den spezifischen Amazon-Ressourcennamen (ARN) für den AWS-Region Ort verwendet, an dem sie derzeit gehostet wird. Sie können die ARN-Details später in diesem Abschnitt anzeigen AWS-Region .

Important

Beachten Sie die folgenden wichtigen Details zur AWS AppConfig Agent Lambda-Erweiterung.

- Die `GetConfiguration` API-Aktion wurde am 28. Januar 2022 als veraltet eingestuft. Aufrufe zum Empfangen von Konfigurationsdaten sollten stattdessen `StartConfigurationSession` und `GetLatestConfiguration` APIs verwenden. Wenn Sie eine Version der AWS AppConfig Agent Lambda-Erweiterung verwenden, die vor dem 28. Januar 2022 erstellt wurde, müssen Sie möglicherweise die Berechtigung für die neue APIs Erweiterung konfigurieren. Weitere Informationen finden Sie unter Konfigurationsdaten werden ohne AWS AppConfig Agent abgerufen.
- AWS AppConfig unterstützt alle unter aufgeführten Versionen. Ältere Erweiterungsversionen Wir empfehlen, regelmäßig auf die neueste Version zu aktualisieren, um die Vorteile der Erweiterungen nutzen zu können.

Themen

- AWS AppConfig Versionshinweise zur Agent Lambda Extension
- Finden Sie die Versionsnummer Ihrer Lambda-Erweiterung
- x86-64-Plattform
- ARM64plattform
- Ältere Erweiterungsversionen

AWS AppConfig Versionshinweise zur Agent Lambda Extension

In der folgenden Tabelle werden Änderungen beschrieben, die an den letzten Versionen der AWS AppConfig Lambda-Erweiterung vorgenommen wurden.

Version	Startdatum	Hinweise
2.0.2037	12.05.2025	Es wurde ein `/ping` Pfad hinzugefügt, der eine einfache Integritätsprüfung ermöglicht, die die Version des Agenten zurückgibt. Enthält auch kleinere Verbesserungen und Fehlerkorrekturen.
2.0.1079	12.12.2024	Kleinere Verbesserungen und Bugfixes.
2.0.719	08.08.2024	Kleinere Verbesserungen und Bugfixes.
2.0.678	23.07.2024	Verbesserungen bei der Unterstützung von Feature-Flag-Zielen, -Varianten und -Splits. Weitere Informationen finden Sie unter Feature-Flags mit mehreren Varianten erstellen.

Version	Startdatum	Hinweise
2.0.501	01.07.2024	Kleinere Verbesserungen und Bugfixes.
2.0.358	01.12.2023	Unterstützung für die folgenden Abruffunktionen wurde hinzugefügt: • Abruf mehrerer Konten: Verwenden Sie den AWS AppConfig Agenten von einem Primärkonto oder vom Abruf aus AWS-Konto , um Konfigurationsdaten von Konten mehrerer Anbieter abzurufen. • Konfigurationskopie auf Festplatte schreiben: Verwenden Sie den AWS AppConfig Agenten, um Konfigurationsdaten auf die Festplatte zu schreiben. Diese Funktion ermöglich t Kunden mit Anwendung en, die Konfigurationsdaten von der Festplatte lesen, die Integration in Anwendungen AWS AppConfig.
2.0.181	14.08.2023	Unterstützung für Israel (Tel Aviv) AWS-Region il-central-1 hinzugefügt.

Version	Startdatum	Hinweise
2.0.165	21.02.2023	Kleinere Fehlerbehebungen. Die Nutzung von Erweiterungen wird über die Konsole nicht mehr auf bestimmte Runtime-Versionen beschränkt. AWS Lambda Unterstützung für Folgendes AWS-Regionen wurde hinzugefügt: • Naher Osten (VAE), me-central-1 • Asien-Pazifik (Hyderabad), ap-south-2 • Asien-Pazifik (Melbourne), ap-southeast-4 • Europa (Spanien), eu-south-2 • Europa (Zürich), eu-central-2

Version	Startdatum	Hinweise
2.0.122	23.08.2022	Unterstützung für einen Tunneling-Proxy hinzugefügt, der mit den Umgebungsvariablen und konfiguriert werden kann. `AWS_APPCONFIG_EXTENSION_PROXY_URL` `AWS_APPCONFIG_EXTENSION_PROXY_HEADERS` .NET 6 als Runtime hinzugefügt. Weitere Hinweise zu Umgebungsvariablen finden Sie unter Konfiguration der AWS AppConfig Agent-Lambda-Erweiterung.
2.0.58	05.03.2022	Verbesserte Unterstützung für Graviton2 (ARM64) -Prozessoren in Lambda.

Version	Startdatum	Hinweise
2.0.45	15.03.2022	Unterstützung für das Aufrufen eines einzelnen Feature-F lags wurde hinzugefügt. Bisher riefen Kunden Feature-Flags auf, die in einem Konfigura tionsprofil gruppiert waren, und mussten die Antwort clientseitig analysieren. Mit dieser Version können Kunden beim Aufrufen des HTTP-Localhost-Endpunkts einen `flag=<flag-name>` Parameter verwenden, um den Wert eines einzelnen Flags abzurufen. Außerdem wurde anfängliche Unterstüt zung für Graviton2 () ARM64 - Prozessoren hinzugefügt.

Finden Sie die Versionsnummer Ihrer Lambda-Erweiterung

Gehen Sie wie folgt vor, um die Versionsnummer Ihrer aktuell konfigurierten AWS AppConfig Agent Lambda-Erweiterung zu ermitteln. Damit Ihre Lambda-Funktion ordnungsgemäß funktioniert, muss sie so konfiguriert sein, dass sie den spezifischen Amazon-Ressourcennamen (ARN) für den AWS-Region Ort verwendet, an dem sie derzeit gehostet wird.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie die Lambda-Funktion aus, der Sie die `AWS-AppConfig-Extension` Ebene hinzufügen möchten.
3. Wählen Sie im Bereich Ebenen die Option Ebene hinzufügen aus.
4. Wählen Sie im Abschnitt „Ebene auswählen“ in der Ebenenliste die AWS Option AWS-AppConfig -Extension aus.
5. Verwenden Sie die Versionsliste, um eine Versionsnummer auszuwählen.

6. Wählen Sie Hinzufügen aus.
7. Verwenden Sie die Registerkarte Test, um die Funktion zu testen.
8. Sehen Sie sich nach Abschluss des Tests die Protokollausgabe an. Suchen Sie im Abschnitt Details der Ausführung nach der Version der AWS AppConfig Agent-Lambda-Erweiterung. Diese Version muss mit der URLs für diese Version erforderlichen Version übereinstimmen.

x86-64-Plattform

Wenn Sie die Erweiterung als Ebene zu Ihrem Lambda hinzufügen, müssen Sie einen ARN angeben. Wählen Sie aus der folgenden Tabelle einen ARN aus, der dem Ort entspricht, AWS-Region an dem Sie das Lambda erstellt haben. Diese ARNs sind für Lambda-Funktionen vorgesehen, die für die x86-64-Plattform entwickelt wurden.

Version 2.0.2037

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:207`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:162`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:258`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:262`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:152`

Region	ARN
Kanada West (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:57
Europa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:189
Europa (Zürich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:106
Europa (Irland)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:189
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:133
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:162
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:259
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:140
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:102

Region	ARN
China (Peking)	`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`
Asien-Pazifik (Hongkong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:142`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:155`
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:165`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:159`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:156`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:199`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:150`

Region	ARN
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:78`
Asien-Pazifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:55`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:175`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:104`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:215`
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:120`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:154`

Region	ARN
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:110`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:109`

ARM64plattform

Wenn Sie die Erweiterung als Ebene zu Ihrem Lambda hinzufügen, müssen Sie einen ARN angeben. Wählen Sie aus der folgenden Tabelle einen ARN aus, der dem Ort entspricht, AWS-Region an dem Sie das Lambda erstellt haben. Diese ARNs sind für Lambda-Funktionen vorgesehen, die für die ARM64 Plattform entwickelt wurden.

Version 2.0.2037

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:140`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:114`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:135`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:164`

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

Region	ARN
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:63
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:70
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:108
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:73
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:74
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:108
Asien-Pazifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:142
Asien-Pazifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:87
Asien-Pazifik (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:63

Region	ARN
Asien-Pazifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:30`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:117`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:62`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:103`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:80`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:76`
Naher Osten (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 (Peking)	`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 (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:56`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:55`

Ältere Erweiterungsversionen

In diesem Abschnitt sind die ARNs und AWS-Regionen für ältere Versionen der AWS AppConfig Lambda-Erweiterung aufgeführt. Diese Liste enthält nicht Informationen für alle früheren Versionen der AWS AppConfig Agent Lambda-Erweiterung, sie wird jedoch aktualisiert, wenn neue Versionen veröffentlicht werden.

Themen

- Ältere Erweiterungsversionen (x86-64-Plattform)
- Ältere Erweiterungsversionen (ARM64 Plattform)

Ältere Erweiterungsversionen (x86-64-Plattform)

In den folgenden Tabellen sind ältere Versionen der AWS-Regionen AWS AppConfig Agent Lambda-Erweiterung aufgeführt ARNs , die für die x86-64-Plattform entwickelt wurde.

Datum, das durch eine neuere Erweiterung ersetzt wurde: 20.05.2025

Version 2.0.1079

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:174`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:133`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:223`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:230`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:123`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:159`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:77`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:160`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:121
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:133
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:225
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:111
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:74
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:113
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:126

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

Region	ARN
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:179`
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:91`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:125`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:80`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:80`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 12.12.2024

Version 2.0.719

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:173`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:132`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:221`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:229`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:121`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:158`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:75`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:159`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:120
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:132
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:224
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:110
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:72
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:112
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:125

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

Region	ARN
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:178`
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:90`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:124`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:79`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:79`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 08/08/2024

Version 2.0.678

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:167`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:126`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:213`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:223`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:116`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:21`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:152`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:70`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:153`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:114
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:126
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:218
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:104
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:67
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:106
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:119

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

Region	ARN
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:84`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:118`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:73`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 23.07.2024

Version 2.0.501

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:153`

Region	ARN
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:112`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:195`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:210`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:101`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:136`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:53`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:144`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:99`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:111`

Region	ARN
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:201
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:89
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:50
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:91
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:104
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:114
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:107

Region	ARN
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:112`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:142`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:98`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:26`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:125`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:53`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:155`
Afrika (Kapstadt)	`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
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:68`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:103`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:59`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:59`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 07/01/2024

Version 2.0.358

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141`

Region	ARN
USA West (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161
Kanada (Zentral)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93
Europa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106
Europa (Zürich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47
Europa (Irland)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83

Region	ARN
Europa (Spain)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44`
China (Peking)	`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`
Asien-Pazifik (Hongkong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98`
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106`

Region	ARN
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128`
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85`

Region	ARN
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 12.01.2023

Version 2.0.181

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:113`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:81`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81`

Region	ARN
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142`
Europa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73`
Europa (Spain)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29`
China (Peking)	`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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91
Asien-Pazifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93
Asien-Pazifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64
Asien-Pazifik (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5

Region	ARN
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113`
Afrika (Kapstadt)	`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`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 14.08.2023

Version 2.0.165

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82

Region	ARN
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110`

Region	ARN
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.02.2023

Version 2.0.122

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59`

Region	ARN
USA West (Nordkalifornien)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93
USA West (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114
Kanada (Zentral)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59
Europe (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70
Europa (Irland)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54

Region	ARN
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64
Asien-Pazifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70
Asien-Pazifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37

Region	ARN
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 23.08.2022

Version 2.0.58

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`

Region	ARN
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98`

Region	ARN
Europa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47`
China (Peking)	`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`
Asien-Pazifik (Hongkong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49`
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59`

Region	ARN
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.04.2022

Version 2.0.45

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50`

Region	ARN
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50

Region	ARN
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 15.03.2022

Version 2.0.30

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48`

Region	ARN
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44
China (Peking)	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
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45

Region	ARN
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20`

Ältere Erweiterungsversionen (ARM64 Plattform)

In den folgenden Tabellen sind ältere Versionen der AWS-Regionen AWS AppConfig Agent Lambda-Erweiterung aufgeführt ARNs , die für die ARM64 Plattform entwickelt wurden.

Datum, das durch eine neuere Erweiterung ersetzt wurde: 20.05.2025

Version 2.0.1079

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:107`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:85`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:100`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:132`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:43`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:18`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:102`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:35`

Region	ARN
Europa (Irland)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:98
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:73
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:52
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:84
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:39
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:35
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:41
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:79
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:44

Region	ARN
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:45
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:86
Asien-Pazifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:108
Asien-Pazifik (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:58
Asien-Pazifik (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:34
Asien-Pazifik (Malaysia)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1
Asien-Pazifik (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:88
Asien-Pazifik (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:33
Südamerika (São Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:67

Region	ARN
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:51`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:47`
Naher Osten (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 (Peking)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:28`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:26`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 12.12.2024

Version 2.0.678

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:106`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:84`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:98`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:131`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:41`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:17`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:101`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:33`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:97`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:72
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:51
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:83
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:38
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:33
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:40
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:78
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:43
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:44

Region	ARN
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:84`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:107`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:57`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:33`
Asien-Pazifik (Malaysia)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:87`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:32`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:66`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:50`

Region	ARN
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:46`
Naher Osten (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 (Peking)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:26`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:24`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:25`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:25`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 08/08/2024

Version 2.0.678

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:100`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:78`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:90`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:125`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:11`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:36`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:95`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:28`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:91`

Region	ARN
Europa (London)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:66
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:45
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:77
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:32
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:28
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:34
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:72
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:37
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:38

Region	ARN
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:79`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:101`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:51`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:27`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:81`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:26`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:60`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:44`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:40`

Region	ARN
Naher Osten (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 (Peking)	`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`
AWS GovCloud (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:19`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:19`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 23.07.2024

Version 2.0.501

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:86`

Region	ARN
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:64`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:72`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:112`
Kanada West (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:1`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:21`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:79`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:11`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:82`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:51`

Region	ARN
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:30
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:60
Europa (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:17
Europa (Spain)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:11
Asien-Pazifik (Hongkong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:19
Asien-Pazifik (Tokio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:57
Asien-Pazifik (Seoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:22
Asien-Pazifik (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:22
Asien-Pazifik (Singapur)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:64

Region	ARN
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:85`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:35`
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:11`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:67`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:11`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:43`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:30`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:24`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:31`

Region	ARN
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:11`
China (Peking)	`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 (US-Ost)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:5`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:5`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 07/01/2024

Version 2.0.358

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45`

Region	ARN
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49`
Europa (Zürich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18`

Region	ARN
Europa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11`
Europa (Spain)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5`
Asien-Pazifik (Hongkong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51`
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16`

Region	ARN
Asien-Pazifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49`
Asien-Pazifik (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11`
Naher Osten (VAE)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:5`
Naher Osten (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`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 12/01/2023

Version 2.0.181

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:46`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:33`
USA West (Nordkalifornien)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:1`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:48`
Kanada (Zentral)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:1`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:36`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:48`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1`

Region	ARN
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Hongkong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37`
Asien-Pazifik (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36`
Asien-Pazifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1`

Region	ARN
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36`
Südamerika (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1`
Afrika (Kapstadt)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1`
Naher Osten (Bahrain)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1`

Datum ersetzt durch eine neuere Erweiterung: 30.03.2023

Version 2.0.165

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45`

Region	ARN
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34`

Datum ersetzt durch eine neuere Erweiterung: 21.02.2023

Version 2.0.122

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:15`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:11`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:16`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:13`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:20`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:11`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:15`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:16`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13`

Region	ARN
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:13`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 23.08.2022

Version 2.0.58

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3`
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2`

Region	ARN
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2`

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.04.2022

Version 2.0.45

Region	ARN
USA Ost (Nord-Virginia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1`
USA Ost (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1`
USA West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2`

Region	ARN
Europe (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Irland)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6`
Europa (London)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Tokio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Singapur)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2`
Asien-Pazifik (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1`
Asien-Pazifik (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1`

AWS AppConfig Agent mit Amazon EC2 - und lokalen Maschinen verwenden

AWS AppConfig Mithilfe von AWS AppConfig Agent können Sie Anwendungen integrieren, die auf Ihren Amazon Elastic Compute Cloud (Amazon EC2) Linux-Instances ausgeführt werden. Der Agent verbessert die Anwendungsverarbeitung und -verwaltung auf folgende Weise:

- Der Agent ruft in Ihrem Namen AWS AppConfig an, indem er eine AWS Identity and Access Management (IAM-) Rolle verwendet und einen lokalen Cache mit Konfigurationsdaten verwaltet. Durch das Abrufen von Konfigurationsdaten aus dem lokalen Cache benötigt Ihre Anwendung weniger Codeaktualisierungen zur Verwaltung der Konfigurationsdaten, ruft Konfigurationsdaten in Millisekunden ab und ist nicht von Netzwerkproblemen betroffen, die Aufrufe solcher Daten unterbrechen können. *
- Der Agent bietet eine native Oberfläche zum Abrufen und Auflösen von AWS AppConfig Feature-Flags.
- Der sofort einsatzbereite Agent bietet bewährte Methoden für Caching-Strategien, Abfrageintervalle und die Verfügbarkeit lokaler Konfigurationsdaten und verfolgt gleichzeitig die für nachfolgende Serviceanfragen benötigten Konfigurationstoken.
- Während der Ausführung im Hintergrund fragt der Agent die AWS AppConfig Datenebene regelmäßig nach Aktualisierungen der Konfigurationsdaten ab. Ihre Anwendung kann die Daten abrufen, indem sie über Port 2772 (ein anpassbarer Standard-Portwert) eine Verbindung zu localhost herstellt und HTTP GET aufruft, um die Daten abzurufen.

*AWS AppConfig Der Agent speichert Daten im Cache, wenn der Dienst Ihre Konfigurationsdaten zum ersten Mal abruft. Aus diesem Grund ist der erste Aufruf zum Abrufen von Daten langsamer als nachfolgende Aufrufe.

Themen

- Schritt 1: (Erforderlich) Ressourcen erstellen und Berechtigungen konfigurieren
- Schritt 2: (Erforderlich) AWS AppConfig Agent auf EC2 Amazon-Instances installieren und starten
- Schritt 3: (Optional, aber empfohlen) Senden von Protokolldateien an CloudWatch Logs
- Schritt 4: (Optional) Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon EC2
- Schritt 5: (Erforderlich) Abrufen von Konfigurationsdaten
- Schritt 6 (optional, aber empfohlen): Automatisieren von Agent-Updates AWS AppConfig

Schritt 1: (Erforderlich) Ressourcen erstellen und Berechtigungen konfigurieren

Für die Integration AWS AppConfig mit Anwendungen, die auf Ihren EC2 Amazon-Instances ausgeführt werden, müssen Sie AWS AppConfig Artefakte und Konfigurationsdaten erstellen, einschließlich Feature-Flags oder Freiform-Konfigurationsdaten. Weitere Informationen finden Sie unter Erstellen von Feature-Flags und Freiform-Konfigurationsdaten in AWS AppConfig.

Um Konfigurationsdaten abzurufen, die von gehostet werden AWS AppConfig, müssen Ihre Anwendungen mit Zugriff auf die AWS AppConfig Datenebene konfiguriert sein. Um Ihren Anwendungen Zugriff zu gewähren, aktualisieren Sie die IAM-Berechtigungsrichtlinie, die der EC2 Amazon-Instance-Rolle zugewiesen ist. Insbesondere müssen Sie der Richtlinie die `appconfig:GetLatestConfiguration` Aktionen `appconfig:StartConfigurationSession` und hinzufügen. Ein Beispiel:

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

Weitere Informationen zum Hinzufügen von Berechtigungen zu einer Richtlinie finden Sie unter Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im IAM-Benutzerhandbuch.

Schritt 2: (Erforderlich) AWS AppConfig Agent auf EC2 Amazon-Instances installieren und starten

AWS AppConfig Der Agent wird in einem Amazon Simple Storage Service (Amazon S3) -Bucket gehostet, der von verwaltet wird AWS. Verwenden Sie das folgende Verfahren, um die neueste Version des Agenten auf Ihrer Linux-Instance zu installieren. Wenn Ihre Anwendung auf mehrere Instanzen verteilt ist, müssen Sie dieses Verfahren für jede Instanz ausführen, die die Anwendung hostet.

> **Note**
>
> Notieren Sie die folgenden Informationen:
>
> - AWS AppConfig Der Agent ist für Linux-Betriebssysteme mit Kernel-Version 4.15 oder höher verfügbar. Debian-basierte Systeme wie Ubuntu werden nicht unterstützt.
> - Der Agent unterstützt x86_64 und Architekturen. ARM64

- Für verteilte Anwendungen empfehlen wir, die Installations- und Startbefehle zu den EC2 Amazon-Benutzerdaten Ihrer Auto Scaling Scaling-Gruppe hinzuzufügen. Wenn Sie dies tun, führt jede Instance die Befehle automatisch aus. Weitere Informationen finden Sie unter Befehle auf Ihrer Linux-Instance beim Start ausführen im EC2 Amazon-Benutzerhandbuch. Weitere Informationen finden Sie unter Tutorial: Benutzerdaten konfigurieren, um den Ziellebenszyklusstatus über Instance-Metadaten abzurufen im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.
- Die Verfahren in diesem Thema beschreiben, wie Sie Aktionen wie die Installation des Agenten durchführen, indem Sie sich bei der Instance anmelden, um den Befehl auszuführen. Sie können die Befehle von einem lokalen Client-Computer aus ausführen und eine oder mehrere Instances als Ziel verwenden, indem Sie Run Command verwenden, ein Tool in AWS Systems Manager. Weitere Informationen finden Sie unter AWS Systems Manager Run Command im Benutzerhandbuch für AWS Systems Manager .
- AWS AppConfig Agent auf Amazon EC2 Linux-Instances ist ein `systemd` Service.

Um AWS AppConfig Agent auf einer Instance zu installieren und zu starten

1. Melden Sie sich bei Ihrer Linux-Instanz an.
2. Öffnen Sie ein Terminal und führen Sie einen der folgenden Befehle mit Administratorrechten aus:

   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
   ```

   Wenn Sie eine bestimmte Version von AWS AppConfig Agent installieren möchten, ersetzen Sie `latest` die URL durch eine bestimmte Versionsnummer. Hier ist ein Beispiel für 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. Führen Sie den folgenden Befehl aus, um den Agenten zu starten:

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

4. Führen Sie den folgenden Befehl aus, um zu überprüfen, ob der Agent ausgeführt wird:

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

Bei Erfolg gibt der Befehl Informationen wie die folgenden zurück:

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

> **Note**
>
> Um den Agenten zu stoppen, führen Sie den folgenden Befehl aus:
>
> ```
> sudo systemctl stop aws-appconfig-agent
> ```

Schritt 3: (Optional, aber empfohlen) Senden von Protokolldateien an CloudWatch Logs

Standardmäßig veröffentlicht der AWS AppConfig Agent Protokolle auf STDERR. Systemd leitet STDOUT und STDERR für alle Dienste, die auf der Linux-Instanz ausgeführt werden, an das Systemd-Journal weiter. Sie können Protokolldaten im Systemd-Journal anzeigen und verwalten, wenn Sie AWS AppConfig Agent nur auf einer oder zwei Instanzen ausführen. Eine bessere Lösung, eine Lösung, die wir für verteilte Anwendungen dringend empfehlen, besteht darin, Protokolldateien auf die Festplatte zu schreiben und dann den CloudWatch Amazon-Agenten zu verwenden, um die Protokolldaten in die AWS Cloud hochzuladen. Darüber hinaus können Sie den CloudWatch Agenten so konfigurieren, dass er alte Protokolldateien von Ihrer Instance löscht, wodurch verhindert wird, dass Ihrer Instance der Speicherplatz ausgeht.

Um die Protokollierung auf der Festplatte zu aktivieren, müssen Sie die `LOG_PATH` Umgebungsvariable festlegen, wie unter beschriebenSchritt 4: (Optional) Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon EC2.

Informationen zu den ersten Schritten mit dem CloudWatch Agenten finden Sie unter Sammeln von Metriken und Protokollen von EC2 Amazon-Instances und lokalen Servern mit dem CloudWatch Agenten im CloudWatch Amazon-Benutzerhandbuch. Sie können Quick Setup, ein Tool in Systems Manager, verwenden, um den CloudWatch Agenten schnell zu installieren. Weitere Informationen finden Sie im AWS Systems Manager Benutzerhandbuch unter Quick Setup Host Management.

> **Warning**
>
> Wenn Sie sich dafür entscheiden, Protokolldateien auf die Festplatte zu schreiben, ohne den CloudWatch Agenten zu verwenden, müssen Sie alte Protokolldateien löschen. AWS AppConfig Der Agent rotiert die Protokolldateien automatisch jede Stunde. Wenn Sie alte Protokolldateien nicht löschen, kann es sein, dass Ihrer Instanz der Speicherplatz ausgeht.

Nachdem Sie den CloudWatch Agenten auf Ihrer Instanz installiert haben, erstellen Sie eine CloudWatch Agenten-Konfigurationsdatei. In der Konfigurationsdatei wird dem CloudWatch Agenten erklärt, wie er mit AWS AppConfig Agent-Protokolldateien arbeiten soll. Weitere Informationen zum Erstellen einer CloudWatch Agent-Konfigurationsdatei finden Sie unter CloudWatch Agenten-Konfigurationsdatei erstellen.

Fügen Sie der CloudWatch Agent-Konfigurationsdatei auf der Instanz den folgenden `logs` Abschnitt hinzu und speichern Sie Ihre Änderungen:

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

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

Wenn der Wert gleich `auto_removal` ist`true`, löscht der CloudWatch Agent automatisch rotierte AWS AppConfig Agenten-Protokolldateien.

Schritt 4: (Optional) Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon EC2

Sie können AWS AppConfig Agent for Amazon mithilfe EC2 von Umgebungsvariablen konfigurieren. Um Umgebungsvariablen für einen `systemd` Service festzulegen, erstellen Sie eine Drop-In-Unit-Datei. Das folgende Beispiel zeigt, wie Sie eine Drop-In-Unit-Datei erstellen, um die AWS AppConfig Agenten-Protokollierungsebene auf festzulegen. `DEBUG`

Beispiel für die Erstellung einer Drop-In-Unit-Datei für Umgebungsvariablen

1. Loggen Sie sich in Ihre Linux-Instanz ein.
2. Öffnen Sie ein Terminal und führen Sie den folgenden Befehl mit Administratorrechten aus. Der Befehl erstellt ein Konfigurationsverzeichnis:

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

3. Führen Sie den folgenden Befehl aus, um die Drop-In-Unit-Datei zu erstellen. *`file_name`*Ersetzen Sie es durch einen Namen für die Datei. Die Erweiterung muss wie folgt lauten`.conf`:

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

4. Geben Sie Informationen in die Drop-In-Unit-Datei ein. Im folgenden Beispiel wird ein `Service` Abschnitt hinzugefügt, der eine Umgebungsvariable definiert. Im Beispiel wird die Protokollebene AWS AppConfig des Agenten auf festgelegt`DEBUG`.

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

5. Führen Sie den folgenden Befehl aus, um die Systemd-Konfiguration neu zu laden:

   ```
   sudo systemctl daemon-reload
   ```

6. Führen Sie den folgenden Befehl aus, um den Agenten neu zu starten AWS AppConfig :

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

Sie können AWS AppConfig Agent for Amazon konfigurieren, EC2 indem Sie die folgenden Umgebungsvariablen in einer Drop-In-Unit-Datei angeben.

> **Note**
>
> Die folgende Tabelle enthält eine Spalte mit Beispielwerten. Je nach Bildschirmauflösung müssen Sie möglicherweise zum Ende der Tabelle und dann nach rechts scrollen, um die Spalte anzuzeigen.

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`ACCESS_TOKEN`	Diese Umgebungsvariable definiert ein Token, das bereitgestellt werden muss, wenn Konfigurationsdaten vom Agent-HTTP-Server angefordert werden. Der Wert des Tokens muss im Autorisierungsheader für HTTP-Anfragen mit dem Autorisierungstyp festgelegt werden`Bearer`. Ein Beispiel. `GET /applications/my_app/...`	Keine	MyAccessToken

Umgebungsvariable	Details	Standardwert	Beispielwert (e)

	`Host: localhost:2772`
	
	`Authorization: Bearer <token value>`		

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`BACKUP_DIRECTORY`	Diese Umgebungsvariable ermöglicht es dem AWS AppConfig Agenten, eine Sicherungskopie jeder abgerufenen Konfiguration im angegebenen Verzeichnis zu speichern. **Important** Auf der Festplatte gesicherte Konfigurationen sind nicht verschlüsselt. Wenn Ihre Konfiguration vertrauliche Daten enthält, AWS AppConfig empfiehlt Ihnen, bei Ihren Dateisystemberechtigungen das Prinzip der geringsten Rechte	Keine	/path/to/backups

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
	anzuwende n. Weitere Informati onen finden Sie unter Sicherhei t in AWS AppConfig.		
`HTTP_PORT`	Diese Umgebungs variable gibt den Port an, auf dem der HTTP-Server für den Agenten läuft.	2772	2772

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
LOG_LEVEL	Diese Umgebungsvariable gibt den Detaillierungsgrad an, den der Agent protokolliert. Jede Ebene umfasst die aktuelle Ebene und alle höheren Ebenen. Der Wert unterscheidet nicht zwischen Groß- und Kleinschreibung. Die Protokollebenen, von den meisten bis hin zu den am wenigsten detaillierten `trace`, `debug` `info`, `warn`, `error` und `none`. Das `trace` Protokoll enthält detaillierte Informationen, einschließlich Zeitinformationen, über den Agenten.	info	Spur debug info warnen error tödlich Keine
LOG_PATH	Der Speicherort auf der Festplatte, in den Protokolle geschrieben werden. Wenn nicht angegeben, werden Protokolle auf stderr geschrieben.	Keine	/.log path/to/logs/agent

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er zusätzliche konfigurationsspezifische Funktionen wie das Abrufen mehrerer Konten und das Speichern der Konfiguration auf der Festplatte nutzt. Weitere Informationen zu diesen Funktionen finden Sie unter Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren.	Keine	Wenn die Konfiguration als Manifest verwendet AWS AppConfig wird:. `MyApp:MyEnv:MyManifestConfig` Beim Laden des Manifests von der Festplatte: `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, von denen der Agent Konfigurationen AWS AppConfig abruft.	3	3

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`POLL_INTERVAL`	Diese Umgebungsvariable steuert, wie oft der Agent AWS AppConfig nach aktualisierten Konfigurationsdaten fragt. Sie können eine Anzahl von Sekunden für das Intervall angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: s für Sekunden, m für Minuten und h für Stunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardmäßig Sekunden. Beispielsweise ergeben 60, 60 Sekunden und 1 m dasselbe Abfrageintervall.	45 Sekunden	45 45 Sekunden 5m 1 h

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PREFETCH_LIST	Diese Umgebungsvariable gibt die Konfigurationsdaten an, die der Agent anfordert, AWS AppConfig sobald er gestartet wird. In einer durch Kommas getrennten Liste können mehrere Konfigurationsbezeichner angegeben werden.	Keine	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Konfiguration 1, ::Konfiguration 2 MyEnv MyApp MyEnv

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PRELOAD_BACKUPS	Wenn auf gesetzt`true`, lädt der AWS AppConfig Agent die im gefundenen Konfigurations-Backups in den BACKUP_DIRECTORY Arbeitsspeicher und überprüft sofort, ob eine neuere Version des Dienstes existiert. Wenn diese Option auf gesetzt ist`false`, lädt der AWS AppConfig Agent den Inhalt einer Konfigurationssicherung nur dann, wenn er keine Konfigurationsdaten vom Dienst abrufen kann, z. B. wenn ein Problem mit Ihrem Netzwerk vorliegt.	true	true false

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PROXY_HEADERS	Diese Umgebungsvariable gibt Header an, die von dem Proxy benötigt werden, auf den in der PROXY_URL Umgebungsvariablen verwiesen wird. Der Wert ist eine durch Kommas getrennte Liste von Headern.	Keine	Header: Wert h1: v1, h2: v2
PROXY_URL	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen vom Agenten zu verwendet werden soll AWS-Services, einschließlich AWS AppConfig. HTTPSund HTTP URLs werden unterstützt.	Keine	http://localhost:7474 https://my-proxy.example.com

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
REQUEST_TIMEOUT	Diese Umgebungsvariable steuert, wie lange der Agent auf eine Antwort wartet. AWS AppConfig Wenn der Dienst nicht antwortet, schlägt die Anfrage fehl. Wenn es sich bei der Anfrage um den ersten Datenabruf handelt, gibt der Agent einen Fehler an Ihre Anwendung zurück. Wenn das Timeout während einer Hintergrundüberprüfung auf aktualisierte Daten auftritt, protokolliert der Agent den Fehler und versucht es nach einer kurzen Verzögerung erneut. Sie können die Anzahl der Millisekunden für das Timeout angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: ms für Millisekunden	3000 ms	3000 3000 ms 5s

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
	und s für Sekunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardm äßig Millisekunden. Beispiel: 5000, 5000 ms und 5 Sekunden führen zu demselben Wert für das Anforderungs-Timeout.		
ROLE_ARN	Diese Umgebungs variable gibt den Amazon-Ressourcenn amen (ARN) einer IAM-Rolle an. AWS AppConfig Der Agent übernimmt diese Rolle, um Konfigura tionsdaten abzurufen.	Keine	arn:aws:i am: :12345678 9012:role/ MyRole
ROLE_EXTE RNAL_ID	Diese Umgebungs variable gibt die externe ID an, die mit dem angenomme nen Rollen-ARN verwendet werden soll.	Keine	MyExternalId

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`ROLE_SESSION_NAME`	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	Keine	AWSAppConfigAgentSession
`SERVICE_REGION`	Diese Umgebungsvariable gibt eine Alternative an AWS-Region , die der AWS AppConfig Agent verwendet, um den AWS AppConfig Dienst aufzurufen. Wenn diese Option nicht definiert ist, versucht der Agent, die aktuelle Region zu ermitteln. Wenn dies nicht möglich ist, kann der Agent nicht gestartet werden.	Keine	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er wartet, bis das Manifest verarbeitet ist, bevor der Start abgeschlossen wird.	true	true false

Schritt 5: (Erforderlich) Abrufen von Konfigurationsdaten

Sie können Konfigurationsdaten mithilfe eines HTTP-Localhost-Aufrufs vom AWS AppConfig Agenten abrufen. Die folgenden Beispiele werden `curl` mit einem HTTP-Client verwendet. Sie können den Agenten mit jedem verfügbaren HTTP-Client, der von Ihrer Anwendungssprache unterstützt wird, oder mit verfügbaren Bibliotheken, einschließlich eines AWS SDK, aufrufen.

Um den vollständigen Inhalt einer bereitgestellten Konfiguration abzurufen

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

Um ein einzelnes Flag und seine Attribute aus einer AWS AppConfig Konfiguration des Typs abzurufen **`Feature Flag`**

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

Um von einer AWS AppConfig Konfiguration des Typs aus auf mehrere Flags und ihre Attribute zuzugreifen **`Feature Flag`**

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

Schritt 6 (optional, aber empfohlen): Automatisieren von Agent-Updates AWS AppConfig

AWS AppConfig Der Agent wird regelmäßig aktualisiert. Um sicherzustellen, dass Sie die neueste Version von AWS AppConfig Agent auf Ihren Instances ausführen, empfehlen wir Ihnen, die folgenden Befehle zu Ihren EC2 Amazon-Benutzerdaten hinzuzufügen. Sie können die Befehle zu den Benutzerdaten entweder in der Instance oder in der EC2 Auto Scaling Scaling-Gruppe hinzufügen. Das Skript installiert und startet bei jedem Start oder Neustart einer Instanz die neueste Version des Agenten.

```
#!/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
```

AWS AppConfig Agent mit Amazon ECS und Amazon EKS verwenden

Mithilfe AWS AppConfig von Agent können Sie Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) integrieren AWS AppConfig . Der Agent fungiert als Sidecar-Container, der neben Ihren Amazon ECS- und Amazon EKS-Containeranwendungen ausgeführt wird. Der Agent verbessert die Verarbeitung und Verwaltung containerisierter Anwendungen auf folgende Weise:

- Der Agent ruft in Ihrem Namen AWS AppConfig an, indem er eine AWS Identity and Access Management (IAM-) Rolle verwendet und einen lokalen Cache mit Konfigurationsdaten verwaltet. Durch das Abrufen von Konfigurationsdaten aus dem lokalen Cache benötigt Ihre Anwendung weniger Codeaktualisierungen zur Verwaltung der Konfigurationsdaten, ruft Konfigurationsdaten in Millisekunden ab und ist nicht von Netzwerkproblemen betroffen, die Aufrufe solcher Daten unterbrechen können. *
- Der Agent bietet eine native Oberfläche zum Abrufen und Auflösen von AWS AppConfig Feature-Flags.
- Der sofort einsatzbereite Agent bietet bewährte Methoden für Caching-Strategien, Abfrageintervalle und die Verfügbarkeit lokaler Konfigurationsdaten und verfolgt gleichzeitig die für nachfolgende Serviceanfragen benötigten Konfigurationstoken.
- Während der Ausführung im Hintergrund fragt der Agent die AWS AppConfig Datenebene regelmäßig nach Aktualisierungen der Konfigurationsdaten ab. Ihre containerisierte Anwendung kann die Daten abrufen, indem sie über Port 2772 (ein anpassbarer Standard-Portwert) eine Verbindung zu localhost herstellt und HTTP GET aufruft, um die Daten abzurufen.
- AWS AppConfig Der Agent aktualisiert die Konfigurationsdaten in Ihren Containern, ohne diese Container neu starten oder recyceln zu müssen.

* Der AWS AppConfig Agent speichert Daten im Cache, wenn der Service Ihre Konfigurationsdaten zum ersten Mal abruft. Aus diesem Grund ist der erste Aufruf zum Abrufen von Daten langsamer als nachfolgende Aufrufe.

Bevor Sie beginnen

Für die Integration AWS AppConfig mit Ihren Containeranwendungen müssen Sie AWS AppConfig Artefakte und Konfigurationsdaten, einschließlich Feature-Flags oder Freiform-Konfigurationsdaten, erstellen. Weitere Informationen finden Sie unter Erstellen von Feature-Flags und Freiform-Konfigurationsdaten in AWS AppConfig.

Um Konfigurationsdaten abzurufen, die von gehostet werden AWS AppConfig, müssen Ihre Containeranwendungen mit Zugriff auf die AWS AppConfig Datenebene konfiguriert sein. Um Ihren Anwendungen Zugriff zu gewähren, aktualisieren Sie die IAM-Berechtigungsrichtlinie, die von Ihrer Containerdienst-IAM-Rolle verwendet wird. Insbesondere müssen Sie der Richtlinie die `appconfig:GetLatestConfiguration` Aktionen `appconfig:StartConfigurationSession` und hinzufügen. Zu den IAM-Rollen des Containerdienstes gehören:

- Die Amazon ECS-Aufgabenrolle
- Die Amazon EKS-Knotenrolle
- Die AWS Fargate Pod-Ausführungsrolle (wenn Ihre Amazon EKS-Container Fargate für die Rechenverarbeitung verwenden)

Weitere Informationen zum Hinzufügen von Berechtigungen zu einer Richtlinie finden Sie unter Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im IAM-Benutzerhandbuch.

Themen

- Den AWS AppConfig Agenten für die Amazon ECS-Integration starten
- Den AWS AppConfig Agenten für die Amazon EKS-Integration starten
- (Optional) Wird AWS AppConfig als DaemonSet in Amazon EKS ausgeführt
- (Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS
- Abrufen von Konfigurationsdaten für Anwendungen, die in Amazon ECS und Amazon EKS ausgeführt werden

Den AWS AppConfig Agenten für die Amazon ECS-Integration starten

Der AWS AppConfig Agent-Sidecar-Container ist automatisch in Ihrer Amazon ECS-Umgebung verfügbar. Um ihn zu verwenden, müssen Sie ihn starten, wie im folgenden Verfahren beschrieben.

Um Amazon ECS (Konsole) zu starten

1. Öffnen Sie die Konsole auf https://console.aws.amazon.com/ecs/Version 2.
2. Wählen Sie im Navigationsbereich Task definitions (Aufgabendefinitionen) aus.
3. Wählen Sie die Aufgabendefinition für Ihre Anwendung und dann die neueste Version aus.
4. Wählen Sie Neue Revision erstellen, Neue Revision erstellen aus.
5. Wählen Sie Weitere Container hinzufügen.
6. Geben Sie unter Name einen eindeutigen Namen für den AWS AppConfig Agent-Container ein.
7. Geben Sie für Bild-URI Folgendes ein: **`public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x`**
8. Wählen Sie für Essential-Container die Option Ja aus.
9. Wählen Sie im Abschnitt Portzuordnungen die Option Portzuordnung hinzufügen aus.
10. Geben Sie für Container-Port den Wert ein. **`2772`**

> **Note**
>
> AWS AppConfig Der Agent wird standardmäßig auf Port 2772 ausgeführt. Sie können einen anderen Port angeben.

11. Wählen Sie Erstellen aus. Amazon ECS erstellt eine neue Container-Revision und zeigt die Details an.
12. Wählen Sie im Navigationsbereich Clusters und dann Ihren Anwendungscluster in der Liste aus.
13. Wählen Sie auf der Registerkarte Dienste den Dienst für Ihre Anwendung aus.
14. Wählen Sie Aktualisieren.
15. Wählen Sie unter Bereitstellungskonfiguration für Revision die neueste Revision aus.
16. Wählen Sie Aktualisieren. Amazon ECS stellt die neueste Aufgabendefinition bereit.
17. Nach Abschluss der Bereitstellung können Sie auf der Registerkarte Konfiguration und Aufgaben überprüfen, ob der AWS AppConfig Agent ausgeführt wird. Wählen Sie auf der Registerkarte Aufgaben die laufende Aufgabe aus.
18. Vergewissern Sie sich, dass der AWS AppConfig Agent-Container im Abschnitt Container aufgeführt ist.
19. Um zu überprüfen, ob der AWS AppConfig Agent gestartet wurde, wählen Sie die Registerkarte Logs. Suchen Sie für den AWS AppConfig Agent-Container nach einer Aussage wie

der folgenden: `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Notieren Sie die folgenden Informationen:
>
> - AWS AppConfig Agent ist ein lang andauernder Prozess. Als bewährte Methode für Amazon ECS-Container sollten Sie Integritätsprüfungen für Ihre Container konfigurieren und insbesondere die Container-Abhängigkeit auf den Zustand HEALTHY setzen. Weitere Informationen finden Sie ContainerDependencyin der Amazon Elastic Container Service API-Referenz.
> - Sie können das Standardverhalten von AWS AppConfig Agent anpassen, indem Sie Umgebungsvariablen eingeben oder ändern. Hinweise zu den verfügbaren Umgebungsvariablen finden Sie unter(Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS. Informationen zum Ändern von Umgebungsvariablen in Amazon ECS finden Sie unter Übergeben von Umgebungsvariablen an einen Container im Amazon Elastic Container Service Developer Guide.

Den AWS AppConfig Agenten für die Amazon EKS-Integration starten

Der AWS AppConfig Agent-Sidecar-Container ist automatisch in Ihrer Amazon EKS-Umgebung verfügbar. Um ihn zu verwenden, müssen Sie ihn starten. Das folgende Verfahren beschreibt, wie Sie das Amazon `kubectl` EKS-Befehlszeilentool verwenden, um den Agenten zu starten.

> **Note**
>
> Bevor Sie fortfahren, stellen Sie sicher, dass Ihre `kubeconfig` Datei auf dem neuesten Stand ist. Weitere Informationen zum Erstellen oder Bearbeiten einer `kubeconfig` Datei finden Sie unter Erstellen oder Aktualisieren einer kubeconfig-Datei für einen Amazon EKS-Cluster im Amazon EKS-Benutzerhandbuch.

So starten Sie den AWS AppConfig Agenten (kubectl-Befehlszeilentool)

1. Öffnen Sie das Manifest für Ihre Anwendung und stellen Sie sicher, dass Ihre Amazon EKS-Anwendung als Einzelcontainer-Bereitstellung ausgeführt wird. Der Inhalt der Datei sollte etwa wie folgt aussehen.

```
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. Fügen Sie die Details der AWS AppConfig Agent-Container-Definition zu Ihrem Bereitstellungsmanifest hinzu.

```
- 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**
>
> Notieren Sie die folgenden Informationen:
>
> - AWS AppConfig Der Agent wird standardmäßig auf Port 2772 ausgeführt. Sie können einen anderen Port angeben.
> - Sie können das Standardverhalten des AWS AppConfig Agenten anpassen, indem Sie Umgebungsvariablen eingeben. Weitere Informationen finden Sie unter (Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS.
> - Geben Sie für *AWS-Region* den AWS-Region Code an (z. B.`us-west-1`), mit dem der AWS AppConfig Agent die Konfigurationsdaten abruft.

3. Führen Sie den folgenden `kubectl` Befehl aus, um die Änderungen auf Ihren Cluster anzuwenden. *my-deployment*Ersetzen Sie es durch den Namen Ihres Bereitstellungsmanifests.

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

4. Stellen Sie nach Abschluss der Bereitstellung sicher, dass der AWS AppConfig Agent ausgeführt wird. Verwenden Sie den folgenden Befehl, um die Anwendungs-Pod-Protokolldatei anzuzeigen.

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

Suchen Sie für den AWS AppConfig Agent-Container nach einer Anweisung wie der folgenden: `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Sie können das Standardverhalten des AWS AppConfig Agenten anpassen, indem Sie Umgebungsvariablen eingeben oder ändern. Hinweise zu den verfügbaren Umgebungsvariablen finden Sie unter(Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS.

(Optional) Wird AWS AppConfig als DaemonSet in Amazon EKS ausgeführt

Mit Amazon EKS können Sie AWS AppConfig Agent als Sidecar ausführen, was zu einem Agent-Container pro Anwendungs-Pod führt. Oder, wenn Sie es vorziehen, können Sie AWS AppConfig Agent als ausführen DaemonSet, was zu einem Agenten-Container pro Knoten in Ihrem Cluster führt.

> **Note**
>
> Wenn Sie AWS AppConfig Agent als ausführen DaemonSet, wird der Agent in einem separaten Pod ausgeführt, was bedeutet, dass Sie mit Aufrufen von nicht darauf zugreifen können`localhost`. Sie müssen die IP-Adresse des Agenten-Pods eingeben oder auf andere Weise ermitteln, um ihn aufrufen zu können.

Um den AWS AppConfig Agenten als auszuführen DaemonSet, erstellen Sie eine Manifestdatei mit dem folgenden Inhalt. Ersetzen Sie *`highlighted`* Text durch Details zu Ihrer Anwendung und Umgebung. Geben Sie für *`AWS-Region`* einen AWS-Region Code an (z. B.`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
```

Führen Sie den folgenden Befehl aus, um den AWS AppConfig Agenten DaemonSet auf Ihren Cluster anzuwenden. *aws_appconfig_agent_daemonset*Ersetzen Sie es durch den Namen Ihres DaemonSet Manifests.

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

(Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS

Sie können AWS AppConfig Agent konfigurieren, indem Sie die folgenden Umgebungsvariablen für Ihren Agent-Container ändern.

> **Note**
>
> Die folgende Tabelle enthält eine Spalte mit Beispielwerten. Je nach Bildschirmauflösung müssen Sie möglicherweise zum Ende der Tabelle und dann nach rechts scrollen, um die Spalte anzuzeigen.

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
ACCESS_TOKEN	Diese Umgebungs variable definiert ein Token, das bereitges tellt werden muss, wenn Konfigura tionsdaten vom Agent-HTTP-Server angefordert werden. Der Wert des Tokens muss im Autorisie	Keine	MyAccessToken

Umgebungsvariable	Details	Standardwert	Beispielwert (e)

	rungsheader für HTTP-Anfragen mit dem Autorisierungstyp festgelegt werden`Bearer`. Ein Beispiel.
	`GET /applications/my_app/...`
	`Host: localhost:2772`
	`Authorization: Bearer <token value>`		

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`BACKUP_DIRECTORY`	Diese Umgebungsvariable ermöglicht es dem AWS AppConfig Agenten, eine Sicherungskopie jeder abgerufenen Konfiguration im angegebenen Verzeichnis zu speichern. **Important** Auf der Festplatte gesicherte Konfigurationen sind nicht verschlüsselt. Wenn Ihre Konfiguration vertrauliche Daten enthält, AWS AppConfig empfiehlt Ihnen, bei Ihren Dateisystemberechtigungen das Prinzip der geringsten Rechte	Keine	/path/to/backups

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
	anzuwende n. Weitere Informati onen finden Sie unter Sicherhei t in AWS AppConfig.		
HTTP_PORT	Diese Umgebungs variable gibt den Port an, auf dem der HTTP-Server für den Agenten läuft.	2772	2772

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
LOG_LEVEL	Diese Umgebungsvariable gibt den Detaillierungsgrad an, den der Agent protokolliert. Jede Ebene umfasst die aktuelle Ebene und alle höheren Ebenen. Der Wert unterscheidet nicht zwischen Groß- und Kleinschreibung. Die Protokollebenen, von den meisten bis hin zu den am wenigsten detaillierten`trace`, lauten wie folgt: `debug`,`info`,`warn`,,`err`undnone. Das `trace` Protokoll enthält detaillierte Informationen, einschließlich Zeitinformationen, über den Agenten.	info	Spur debug info warnen error tödlich Keine
LOG_PATH	Der Speicherort auf der Festplatte, in den Protokolle geschrieben werden. Wenn nicht angegeben, werden Protokolle auf stderr geschrieben.	Keine	/.log path/to/logs/agent

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er zusätzliche konfigurationsspezifische Funktionen wie das Abrufen mehrerer Konten und das Speichern der Konfiguration auf der Festplatte nutzt. Weitere Informationen zu diesen Funktionen finden Sie unter Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren.	Keine	Wenn die Konfiguration als Manifest verwendet AWS AppConfig wird:. `MyApp:MyEnv:MyManifestConfig` Beim Laden des Manifests von der Festplatte: `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, von denen der Agent Konfigurationen AWS AppConfig abruft.	3	3

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
POLL_INTERVAL	Diese Umgebungsvariable steuert, wie oft der Agent AWS AppConfig nach aktualisierten Konfigurationsdaten fragt. Sie können eine Anzahl von Sekunden für das Intervall angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: s für Sekunden, m für Minuten und h für Stunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardmäßig Sekunden. Beispielsweise ergeben 60, 60 Sekunden und 1 m dasselbe Abfrageintervall.	45 Sekunden	45 45 Sekunden 5m 1 h

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PREFETCH_LIST	Diese Umgebungsvariable gibt die Konfigurationsdaten an, die der Agent anfordert, AWS AppConfig sobald er gestartet wird. In einer durch Kommas getrennten Liste können mehrere Konfigurationsbezeichner angegeben werden.	Keine	MyApp:MyEnv:MyConfig abcd123:efgh456:ijkl789 MyApp::Konfiguration 1, ::Konfiguration 2 MyEnv MyApp MyEnv

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PRELOAD_BACKUPS	Wenn auf gesetzt`true`, lädt der AWS AppConfig Agent die im gefundenen Konfigurations-Backups in den BACKUP_DIRECTORY Arbeitsspeicher und überprüft sofort, ob eine neuere Version des Dienstes existiert. Wenn diese Option auf gesetzt ist`false`, lädt der AWS AppConfig Agent den Inhalt einer Konfigurationssicherung nur dann, wenn er keine Konfigurationsdaten vom Dienst abrufen kann, z. B. wenn ein Problem mit Ihrem Netzwerk vorliegt.	true	true false

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
PROXY_HEADERS	Diese Umgebungsvariable gibt Header an, die von dem Proxy benötigt werden, auf den in der PROXY_URL Umgebungsvariablen verwiesen wird. Der Wert ist eine durch Kommas getrennte Liste von Headern.	Keine	Header: Wert h1: v1, h2: v2
PROXY_URL	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen vom Agenten zu verwendet werden soll AWS-Services, einschließlich AWS AppConfig. HTTPSund HTTP URLs werden unterstützt.	Keine	http://localhost:7474 https://my-proxy.example.com

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`REQUEST_TIMEOUT`	Diese Umgebungsvariable steuert, wie lange der Agent auf eine Antwort wartet. AWS AppConfig Wenn der Dienst nicht antwortet, schlägt die Anfrage fehl. Wenn es sich bei der Anfrage um den ersten Datenabruf handelt, gibt der Agent einen Fehler an Ihre Anwendung zurück. Wenn das Timeout während einer Hintergrundüberprüfung auf aktualisierte Daten auftritt, protokolliert der Agent den Fehler und versucht es nach einer kurzen Verzögerung erneut. Sie können die Anzahl der Millisekunden für das Timeout angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: ms für Millisekunden	3000 ms	3000 3000 ms 5s

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
	und s für Sekunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardm äßig Millisekunden. Beispiel: 5000, 5000 ms und 5 Sekunden führen zu demselben Wert für das Anforderungs-Timeout.		
`ROLE_ARN`	Diese Umgebungs variable gibt den Amazon-Ressourcenn amen (ARN) einer IAM-Rolle an. AWS AppConfig Der Agent übernimmt diese Rolle, um Konfigura tionsdaten abzurufen.	Keine	arn:aws:i am: :12345678 9012:role/ MyRole
`ROLE_EXTE RNAL_ID`	Diese Umgebungs variable gibt die externe ID an, die mit dem angenomme nen Rollen-ARN verwendet werden soll.	Keine	MyExternalId

Umgebungsvariable	Details	Standardwert	Beispielwert (e)
`ROLE_SESSION_NAME`	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	Keine	AWSAppConfigAgentSession
`SERVICE_REGION`	Diese Umgebungsvariable gibt eine Alternative an AWS-Region , die der AWS AppConfig Agent verwendet, um den AWS AppConfig Dienst aufzurufen. Wenn diese Option nicht definiert ist, versucht der Agent, die aktuelle Region zu ermitteln. Wenn dies nicht möglich ist, kann der Agent nicht gestartet werden.	Keine	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er wartet, bis das Manifest verarbeitet ist, bevor der Start abgeschlossen wird.	true	true false

Abrufen von Konfigurationsdaten für Anwendungen, die in Amazon ECS und Amazon EKS ausgeführt werden

Sie können Konfigurationsdaten von AWS AppConfig Agent für Anwendungen abrufen, die in Amazon ECS und Amazon EKS ausgeführt werden, indem Sie einen HTTP-Localhost-Aufruf verwenden. Die folgenden Beispiele werden `curl` mit einem HTTP-Client verwendet. Sie können den Agenten mit jedem verfügbaren HTTP-Client aufrufen, der von Ihrer Anwendungssprache oder verfügbaren Bibliotheken unterstützt wird.

> **Note**
>
> Wenn Ihre Anwendung einen Schrägstrich verwendet, z. B. „test-backend/test-service“, müssen Sie die URL-Kodierung verwenden, um Konfigurationsdaten abzurufen.

Um den vollständigen Inhalt einer bereitgestellten Konfiguration abzurufen

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

Um ein einzelnes Flag und seine Attribute aus einer AWS AppConfig Konfiguration des Typs abzurufen **`Feature Flag`**

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

Um von einer AWS AppConfig Konfiguration des Typs aus auf mehrere Flags und ihre Attribute zuzugreifen **`Feature Flag`**

```

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

Der Aufruf gibt Konfigurationsmetadaten in HTTP-Headern zurück, einschließlich der Konfigurationsversion, des Inhaltstyps und der Bezeichnung der Konfigurationsversion (falls zutreffend). Der Hauptteil der Agentenantwort enthält den Konfigurationsinhalt. Ein Beispiel:

```
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
```

Feature-Flags für grundlegende und variantenreiche Funktionen werden abgerufen

Bei Feature-Flag-Konfigurationen (Typkonfigurationen`AWS.AppConfig.FeatureFlags`) ermöglicht Ihnen der AWS AppConfig Agent, ein einzelnes Flag oder eine Teilmenge von Flags in einer Konfiguration abzurufen. Das Abrufen von einem oder zwei Flags ist nützlich, wenn Ihr Anwendungsfall nur einige Flags aus dem Konfigurationsprofil verwenden muss. Die folgenden Beispiele verwenden cURL.

> **Note**
>
> Die Möglichkeit, ein einzelnes Feature-Flag oder eine Teilmenge von Flags in einer Konfiguration aufzurufen, ist nur in der AWS AppConfig Agent-Version 2.0.45 und höher verfügbar.

Sie können AWS AppConfig Konfigurationsdaten von einem lokalen HTTP-Endpunkt abrufen. Verwenden Sie den `?flag=`*`FLAG_KEY`* Abfrageparameter für ein AWS AppConfig Konfigurationsprofil, um auf ein bestimmtes Flag oder eine Liste von Flags zuzugreifen.

Um ein einzelnes Flag und seine Attribute abzurufen

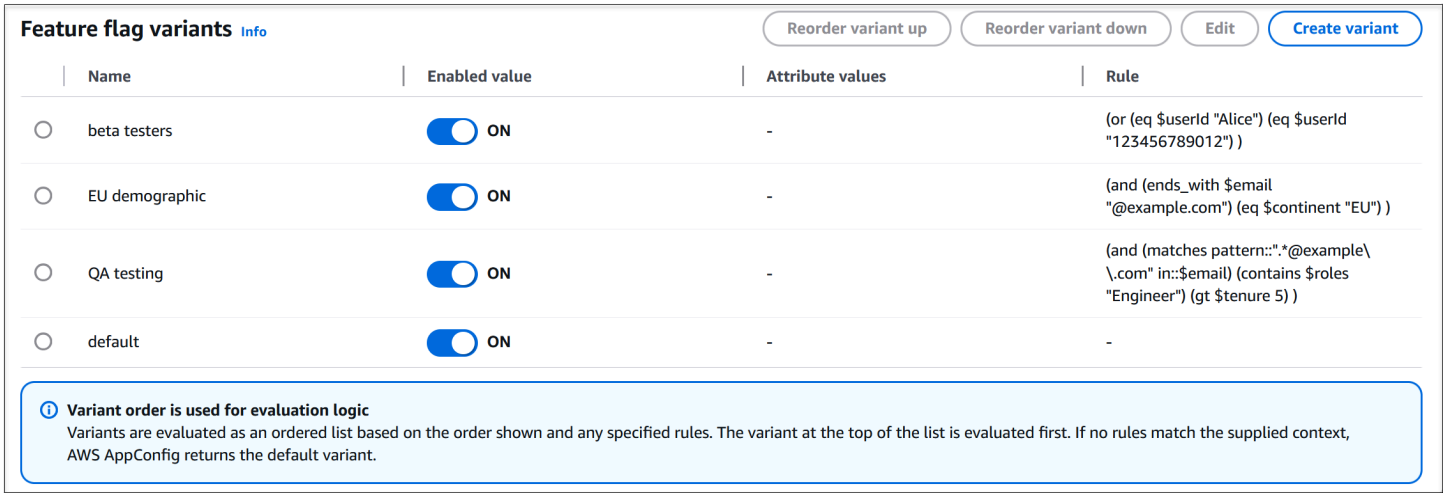
```
curl "http://localhost:2772/applications/APPLICATION_NAME/
environments/ENVIRONMENT_NAME/configurations/CONFIGURATION_NAME?flag=FLAG_KEY"
```

Um mehrere Flags und ihre Attribute abzurufen

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

Um Feature-Flag-Varianten basierend auf dem Anruferkontext abzurufen

Die folgenden Python-Beispiele zeigen, wie Feature-Flag-Varianten basierend auf dem Aufruferkontext abgerufen werden. Um am besten zu veranschaulichen, wie diese Aufrufe getätigt werden, werden in diesem Abschnitt Beispielanrufe verwendet, die auf einem Szenario basieren, in dem ein Kunde die folgenden Varianten erstellt hat:



> **Note**
>
> Um Flag-Varianten abzurufen, müssen Sie die neueste Version von AWS AppConfig Agent in Ihrer Computerumgebung verwenden. Weitere Informationen finden Sie in den folgenden Themen, in denen beschrieben wird, wie Sie den Agenten für jede der folgenden Computerumgebungen aktualisieren, installieren oder hinzufügen:
>
> - Für Lambda-Rechenumgebungen: Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung
> - Für EC2 Amazon-Rechenumgebungen: Schritt 2: (Erforderlich) AWS AppConfig Agent auf EC2 Amazon-Instances installieren und starten
> - Für Amazon ECS-Rechenumgebungen: Den AWS AppConfig Agenten für die Amazon ECS-Integration starten
> - Für Amazon EKS-Rechenumgebungen: Den AWS AppConfig Agenten für die Amazon EKS-Integration starten

So rufen Sie Flag-Daten mithilfe des Anruferkontextes von jane_doe@example.org ab (wer hat sich nicht für das Beta-Programm angemeldet):

```
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}
}
```

So rufen Sie Flaggendaten mithilfe des Anruferkontextes von jane_doe@example.org ab (der sich für das Beta-Programm angemeldet hat):

```
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}
}
```

So rufen Sie Flaggendaten mithilfe des Anruferkontextes von jane_doe@qa-testers.example.org ab (ein Qualitätssicherungstester bei Example Organization):

```
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}
}
```

Um Flaggendaten ohne Anruferkontext abzurufen (was die Standardvariante zurückgibt)

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

Um Flaggendaten für ein Szenario zur Verkehrsaufteilung abzurufen, um festzustellen, ob 1 von 10 zufälligen Anrufern die Variante „Stichprobenpopulation“ erhält

```
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}
}
```

Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren

AWS AppConfig Der Agent bietet die folgenden zusätzlichen Funktionen, mit denen Sie Konfigurationen für Ihre Anwendungen abrufen können.

- AWS AppConfig Agent zum Abrufen von Konfigurationen von mehreren Konten konfigurieren: Verwenden Sie den AWS AppConfig Agenten von einem Primärserver oder einem Abruf aus AWS-Konto , um Konfigurationsdaten von Konten mehrerer Anbieter abzurufen.
- Den AWS AppConfig Agenten so konfigurieren, dass er Konfigurationskopien auf die Festplatte schreibt: Verwenden Sie den AWS AppConfig Agenten, um Konfigurationsdaten auf die Festplatte zu schreiben. Diese Funktion ermöglicht Kunden mit Anwendungen, die Konfigurationsdaten von der Festplatte lesen, die Integration in diese Anwendungen AWS AppConfig.

Agentenmanifeste verstehen

Um diese AWS AppConfig Agent-Funktionen zu aktivieren, erstellen Sie ein Manifest. Ein Manifest besteht aus einer Reihe von Konfigurationsdaten, die Sie angeben, um die Aktionen zu steuern, die der Agent ausführen kann. Ein Manifest ist in JSON geschrieben. Es enthält eine Reihe von Schlüsseln der obersten Ebene, die verschiedenen Konfigurationen entsprechen, mit AWS AppConfig denen Sie sie bereitgestellt haben.

Ein Manifest kann mehrere Konfigurationen enthalten. Darüber hinaus kann jede Konfiguration im Manifest eine oder mehrere Agentenfunktionen identifizieren, die für die angegebene Konfiguration verwendet werden sollen. Der Inhalt des Manifests verwendet das folgende Format:

```
{
    "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"
        }
    }
}
```

Hier ist ein JSON-Beispiel für ein Manifest mit zwei Konfigurationen. Die erste Konfiguration (*MyApp*) verwendet keine AWS AppConfig Agentenfunktionen. Die zweite Konfiguration (*My2ndApp*) verwendet die Funktionen zum Kopieren der Konfiguration auf Festplatte und zum Abrufen mehrerer Konten:

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

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

Wie stellt man ein Agentenmanifest bereit

Sie können das Manifest als Datei an einem Ort speichern, an dem der AWS AppConfig Agent es lesen kann. Sie können das Manifest auch als AWS AppConfig Konfiguration speichern und den Agenten darauf verweisen. Um ein Agentenmanifest bereitzustellen, müssen Sie eine `MANIFEST` Umgebungsvariable mit einem der folgenden Werte festlegen:

Speicherort des Manifests	Wert der Umgebungsvariablen	Anwendungsfall
Datei	Datei:/path/to/agent-manifest.json	Verwenden Sie diese Methode, wenn sich Ihr Manifest nicht oft ändert.
AWS AppConfig Konfiguration	*application-name*:*environment-name*:*configuration-name*	Verwenden Sie diese Methode für dynamische Updates. Sie können ein in AWS AppConfig einer Konfiguration gespeichertes Manifest auf die gleiche Weise aktualisieren und bereitstellen, wie Sie andere AWS AppConfig Konfigurationen speichern.
Umgebungsvariable	Inhalt des Manifests (JSON)	Verwenden Sie diese Methode, wenn sich Ihr Manifest nicht oft ändert.

Speicherort des Manifests	Wert der Umgebungsvariablen	Anwendungsfall
		Diese Methode ist in Container umgebungen nützlich, in denen es einfacher ist, eine Umgebungsvariable festzuleg en, als eine Datei verfügbar zu machen.

Weitere Informationen zum Festlegen von Variablen für AWS AppConfig Agent finden Sie im entsprechenden Thema für Ihren Anwendungsfall:

- Konfiguration der AWS AppConfig Agent-Lambda-Erweiterung
- AWS AppConfig Agent mit Amazon verwenden EC2
- AWS AppConfig Agent mit Amazon ECS und Amazon EKS verwenden

AWS AppConfig Agent zum Abrufen von Konfigurationen von mehreren Konten konfigurieren

Sie können den AWS AppConfig Agenten so konfigurieren, dass er Konfigurationen von mehreren abruft, AWS-Konten indem Sie die Überschreibungen der Anmeldeinformationen in das AWS AppConfig Agent-Manifest eingeben. Zu den Überschreibungen von Anmeldeinformationen gehören der Amazon-Ressourcenname (ARN) einer AWS Identity and Access Management (IAM) -Rolle, eine Rollen-ID, ein Sitzungsname und eine Dauer, für die der Agent die Rolle übernehmen kann.

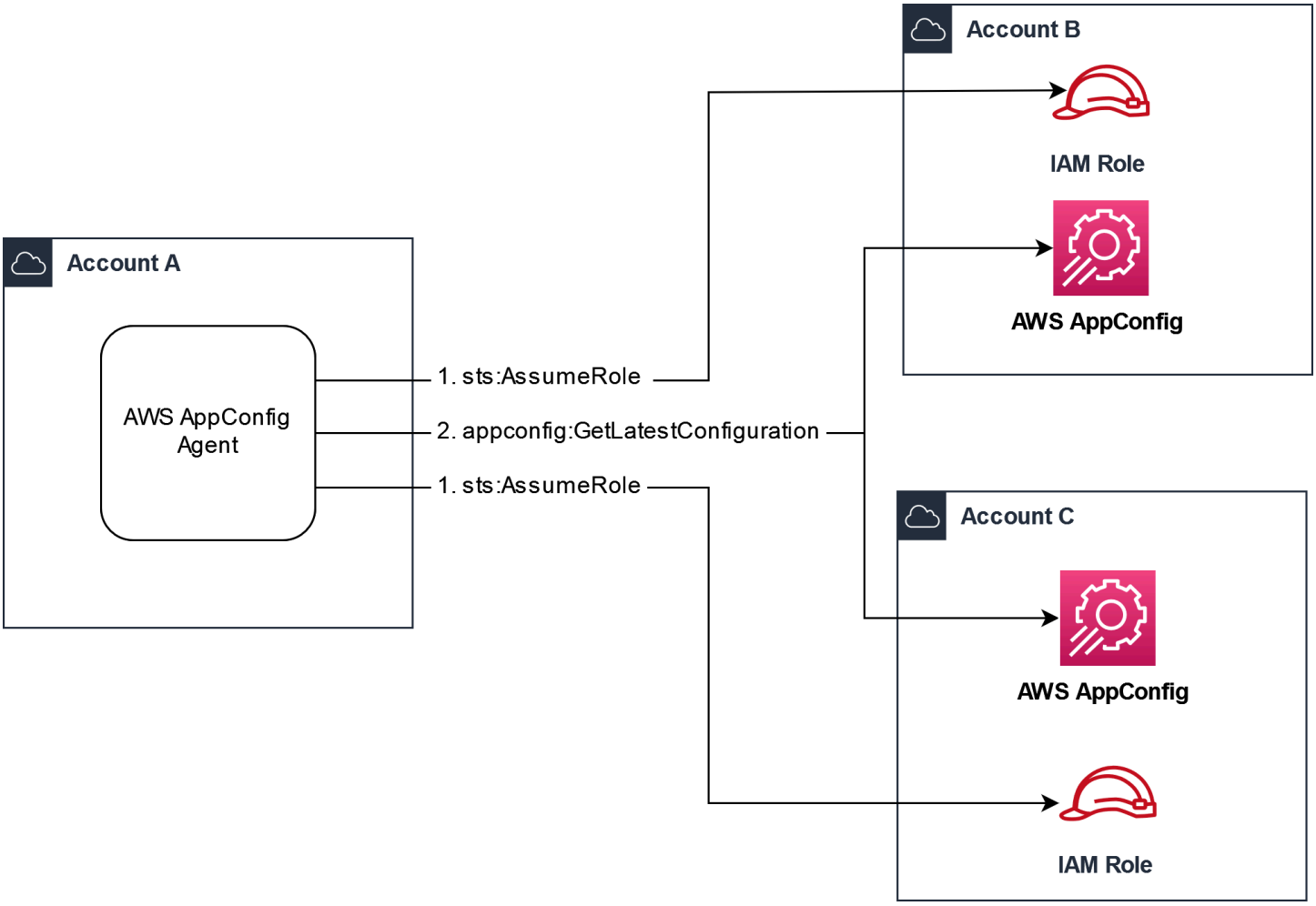
Sie geben diese Details im Manifest im Abschnitt „Anmeldeinformationen“ ein. Der Abschnitt „Anmeldeinformationen“ verwendet das folgende Format:

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

Ein Beispiel:

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

Vor dem Abrufen einer Konfiguration liest der Agent die Anmeldeinformationen für die Konfiguration aus dem Manifest und nimmt dann die für diese Konfiguration angegebene IAM-Rolle an. Sie können einen anderen Satz von Überschreibungen für Anmeldeinformationen für verschiedene Konfigurationen in einem einzigen Manifest angeben. Das folgende Diagramm zeigt, wie der AWS AppConfig Agent, während er in Konto A (dem Abrufkonto) ausgeführt wird, separate Rollen annimmt, die für die Konten B und C (die Lieferantenkonten) angegeben sind, und dann den GetLatestConfigurationAPI-Vorgang aufruft, um Konfigurationsdaten aus der AWS AppConfig Ausführung in diesen Konten abzurufen:



Konfigurieren Sie Berechtigungen zum Abrufen von Konfigurationsdaten aus Lieferantenkonten

AWS AppConfig Der Agent, der im Abrufkonto ausgeführt wird, benötigt die Berechtigung, Konfigurationsdaten von den Herstellerkonten abzurufen. Sie erteilen dem Agenten die entsprechende Berechtigung, indem Sie in jedem der Lieferantenkonten eine AWS Identity and Access Management (IAM-) Rolle erstellen. AWS AppConfig Der Agent im Abrufkonto übernimmt diese Rolle, um Daten von Lieferantenkonten abzurufen. Gehen Sie wie in diesem Abschnitt beschrieben vor, um eine IAM-Berechtigungsrichtlinie und eine IAM-Rolle zu erstellen und dem Manifest Agentenüberschreibungen hinzuzufügen.

Bevor Sie beginnen

Sammeln Sie die folgenden Informationen, bevor Sie eine Berechtigungsrichtlinie und eine Rolle in IAM erstellen.

- Die IDs für jeden AWS-Konto. Das Abrufkonto ist das Konto, das andere Konten für Konfigurationsdaten aufruft. Die Lieferantenkonten sind die Konten, die Konfigurationsdaten an das Abrufkonto weiterleiten.
- Der Name der IAM-Rolle, die von AWS AppConfig im Abrufkonto verwendet wird. Hier ist eine Liste der Rollen AWS AppConfig, die standardmäßig verwendet werden:
  - AWS AppConfig Verwendet für Amazon Elastic Compute Cloud (Amazon EC2) die Instanzrolle.
  - For AWS AppConfig verwendet AWS Lambda die Lambda-Ausführungsrolle.
  - AWS AppConfig Verwendet für Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) die Container-Rolle.

  Wenn Sie den AWS AppConfig Agenten für die Verwendung einer anderen IAM-Rolle konfiguriert haben, indem Sie die `ROLE_ARN` Umgebungsvariable angegeben haben, notieren Sie sich diesen Namen.

Erstellen Sie die Berechtigungsrichtlinie

Gehen Sie wie folgt vor, um mithilfe der IAM-Konsole eine Berechtigungsrichtlinie zu erstellen. Führen Sie das Verfahren für jeden Vorgang aus AWS-Konto , der die Konfigurationsdaten für das Abrufkonto bereitstellt.

So erstellen Sie eine IAM-Richtlinie

1. Melden Sie sich AWS Management Console bei einem Lieferantenkonto an.
2. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
3. Wählen Sie im Navigationsbereich Richtlinien und dann Richtlinie erstellen.
4. Wählen Sie die JSON-Option.
5. Ersetzen Sie im Policy-Editor das Standard-JSON durch die folgende Richtlinienanweisung. Aktualisieren Sie jedes *`example resource placeholder`* mit den Kontodetails des Anbieters.

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

```
        ],
        "Resource":
 "arn:partition:appconfig:region:vendor_account_ID:application/
vendor_application_ID/environment/vendor_environment_ID/
configuration/vendor_configuration_ID"
    }
   ]
}
```

Ein Beispiel:

```
{
    "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. Wählen Sie Weiter aus.
7. Geben Sie im Feld Richtlinienname einen Namen ein.
8. (Optional) Fügen Sie unter Tags hinzufügen ein oder mehrere Tag-Schlüssel-Wertepaare hinzu, um den Zugriff auf diese Richtlinie zu organisieren, nachzuverfolgen oder zu kontrollieren.
9. Wählen Sie Richtlinie erstellen aus. Das System führt Sie zur Seite Policies (Richtlinien) zurück.
10. Wiederholen Sie diesen Vorgang in allen Fällen, in AWS-Konto denen die Konfigurationsdaten für das Abrufkonto ausgegeben werden.

Erstellen Sie die IAM-Rolle

Gehen Sie wie folgt vor, um mithilfe der IAM-Konsole eine IAM-Rolle zu erstellen. Führen Sie das Verfahren in jedem Abschnitt aus AWS-Konto , der die Konfigurationsdaten für das Abrufkonto verkauft.

So erstellen Sie eine IAM-Rolle

1. Melden Sie sich AWS Management Console bei einem Lieferantenkonto an.
2. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
3. Wählen Sie im Navigationsbereich Rollen und dann Richtlinie erstellen aus.
4. Wählen Sie für Vertrauenswürdige Entität die Option AWS-Konto aus.
5. Wählen Sie in dem AWS-KontoAbschnitt „Andere“ aus AWS-Konto.
6. Geben Sie im Feld Konto-ID die Konto-ID für den Abruf ein.
7. (Optional) Wählen Sie als bewährte Sicherheitsmethode für diese Rolle die Option Externe ID erforderlich aus und geben Sie eine Zeichenfolge ein.
8. Wählen Sie Weiter aus.
9. Verwenden Sie auf der Seite „Berechtigungen hinzufügen“ das Suchfeld, um die Richtlinie zu finden, die Sie im vorherigen Verfahren erstellt haben. Aktivieren Sie das Kontrollkästchen neben dem Namen.
10. Wählen Sie Weiter aus.
11. Geben Sie in Role name (Name der Rolle) einen Namen ein.
12. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
13. Wählen Sie für Schritt 1: Vertrauenswürdige Entitäten auswählen die Option Bearbeiten aus. Ersetzen Sie die standardmäßige JSON-Vertrauensrichtlinie durch die folgende Richtlinie. Aktualisieren Sie sie jeweils *example resource placeholder* mit Informationen aus Ihrem Abrufkonto.

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

14. (Optional) Fügen Sie für Tags ein oder mehrere Tag-Schlüssel-Wert-Paare hinzu, um den Zugriff für diese Rolle zu organisieren, nachzuverfolgen oder zu steuern.
15. Wählen Sie Create role (Rolle erstellen) aus. Das System leitet Sie zur Seite Rollen zurück.
16. Suchen Sie nach der Rolle, die Sie gerade erstellt haben. Wählen Sie diese aus. Kopieren Sie im Abschnitt ARN den ARN. Sie geben diese Informationen im nächsten Verfahren an.

Fügen Sie dem Manifest Überschreibungen für Anmeldeinformationen hinzu

Nachdem Sie die IAM-Rolle in Ihrem Lieferantenkonto erstellt haben, aktualisieren Sie das Manifest im Abrufkonto. Fügen Sie insbesondere den Anmeldeinformationsblock und den IAM-Rollen-ARN zum Abrufen von Konfigurationsdaten aus dem Lieferantenkonto hinzu. Hier ist das JSON-Format:

```
{
    "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"
        }
    }
}
```

Ein Beispiel:

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

Stellen Sie sicher, dass der Abruf mehrerer Konten funktioniert

Sie können überprüfen, ob der Agent in der Lage ist, Konfigurationsdaten von mehreren Konten abzurufen, indem Sie die AWS AppConfig Agentenprotokolle überprüfen. Das `INFO` Level-Log für die abgerufenen Anfangsdaten für '`YourApplicationNameYourEnvironmentName::YourConfigurationName`' ist der beste Indikator für erfolgreiche Abrufe. Wenn Abrufe fehlschlagen, sollte ein Ebenenprotokoll mit Angabe der `ERROR` Fehlerursache angezeigt werden. Hier ist ein Beispiel für einen erfolgreichen Abruf von einem Lieferantenkonto:

```
[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
```

Den AWS AppConfig Agenten so konfigurieren, dass er Konfigurationskopien auf die Festplatte schreibt

Sie können den AWS AppConfig Agenten so konfigurieren, dass er automatisch eine Kopie einer Konfiguration im Klartext auf der Festplatte speichert. Diese Funktion ermöglicht Kunden mit Anwendungen, die Konfigurationsdaten von der Festplatte lesen, die Integration in Anwendungen AWS AppConfig.

Diese Funktion ist nicht für die Verwendung als Funktion zur Sicherung der Konfiguration konzipiert. AWS AppConfig Der Agent liest nicht aus den auf die Festplatte kopierten Konfigurationsdateien. Informationen zum Sichern von Konfigurationen auf der Festplatte finden Sie in den `BACKUP_DIRECTORY` `PRELOAD_BACKUP` Umgebungsvariablen Using AWS AppConfig Agent with Amazon EC2 oder Using AWS AppConfig Agent with Amazon ECS and Amazon EKS.

> **Warning**
>
> Beachten Sie die folgenden wichtigen Informationen zu dieser Funktion:
>
> - Auf der Festplatte gespeicherte Konfigurationen werden im Klartext gespeichert und sind für Menschen lesbar. Aktivieren Sie diese Funktion nicht für Konfigurationen, die vertrauliche Daten enthalten.
> - Diese Funktion schreibt auf die lokale Festplatte. Verwenden Sie das Prinzip der geringsten Rechte für Dateisystemberechtigungen. Weitere Informationen finden Sie unter Implementieren des Zugriffs mit geringsten Berechtigungen.

Kopieren Sie die Konfiguration auf die Festplatte, um das Schreiben der Konfiguration zu aktivieren

1. Bearbeiten Sie das Manifest.
2. Wählen Sie die Konfiguration aus, die Sie AWS AppConfig auf die Festplatte schreiben möchten, und fügen Sie ein `writeTo` Element hinzu. Ein Beispiel:

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

   Ein Beispiel:

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

3. Speichern Sie Ihre Änderungen. Die Datei configuration.json wird jedes Mal aktualisiert, wenn neue Konfigurationsdaten bereitgestellt werden.

Stellen Sie sicher, dass das Schreiben der Konfigurationskopie auf die Festplatte funktioniert

Anhand der AWS AppConfig Agentenprotokolle können Sie überprüfen, ob Kopien einer Konfiguration auf die Festplatte geschrieben werden. Der `INFO` Protokolleintrag mit der Formulierung „INFO hat die Konfiguration '*application*:*environment*:*configuration*' nach*file_path*" geschrieben, weist darauf hin, dass der AWS AppConfig Agent Konfigurationskopien auf die Festplatte schreibt.

Ein Beispiel:

```
[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
```

Generierung eines Clients mithilfe der OpenAPI-Spezifikation

Sie können die folgende YAML-Spezifikation für OpenAPI verwenden, um ein SDK mit einem Tool wie OpenAPI Generator zu erstellen. Sie können diese Spezifikation so aktualisieren, dass sie hartcodierte Werte für Anwendung, Umgebung oder Konfiguration enthält. Sie können auch zusätzliche Pfade hinzufügen (wenn Sie mehrere Konfigurationstypen haben) und Konfigurationsschemas einbeziehen, um konfigurationsspezifische typisierte Modelle für Ihre SDK-Clients zu generieren. Weitere Informationen zu OpenAPI (auch bekannt als Swagger) finden Sie in der OpenAPI-Spezifikation.

```
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
```

Mit dem lokalen Entwicklungsmodus des Agenten AWS AppConfig arbeiten

AWS AppConfig Der Agent unterstützt einen lokalen Entwicklungsmodus. Wenn Sie den lokalen Entwicklungsmodus aktivieren, liest der Agent Konfigurationsdaten aus einem angegebenen Verzeichnis auf der Festplatte. Er ruft keine Konfigurationsdaten von ab AWS AppConfig. Sie

können Konfigurationsbereitstellungen simulieren, indem Sie Dateien im angegebenen Verzeichnis aktualisieren. Wir empfehlen den lokalen Entwicklungsmodus für die folgenden Anwendungsfälle:

- Testen Sie verschiedene Konfigurationsversionen, bevor Sie sie mithilfe von bereitstellen AWS AppConfig.
- Testen Sie verschiedene Konfigurationsoptionen für eine neue Funktion, bevor Sie Änderungen in Ihr Code-Repository übernehmen.
- Testen Sie verschiedene Konfigurationsszenarien, um sicherzustellen, dass sie erwartungsgemäß funktionieren.

> **Warning**
>
> Verwenden Sie den lokalen Entwicklungsmodus nicht in Produktionsumgebungen. Dieser Modus unterstützt keine wichtigen AWS AppConfig Sicherheitsfunktionen wie Bereitstellungsvalidierung und automatische Rollbacks.

Gehen Sie wie folgt vor, um den AWS AppConfig Agenten für den lokalen Entwicklungsmodus zu konfigurieren.

So konfigurieren Sie den AWS AppConfig Agenten für den lokalen Entwicklungsmodus

1. Installieren Sie den Agenten mit der für Ihre Computerumgebung beschriebenen Methode. AWS AppConfig Der Agent arbeitet mit den folgenden Funktionen AWS-Services:
   - AWS Lambda
   - Amazon EC2
   - Amazon ECS und Amazon EKS
2. Wenn der Agent läuft, beenden Sie ihn.
3. `LOCAL_DEVELOPMENT_DIRECTORY`Zur Liste der Umgebungsvariablen hinzufügen. Geben Sie ein Verzeichnis im Dateisystem an, das dem Agenten Leserechte gewährt. Beispiel, `/tmp/local_configs`.
4. Erstellen Sie eine Datei in dem Verzeichnis. Der Dateiname muss das folgende Format haben:

   ```
   application_name:environment_name:configuration_profile_name
   ```

Ein Beispiel:

```
Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
```

> **Note**
>
> - Beispiele für Feature-Flags, die Sie einer Datei in Ihrem `LOCAL_DEVELOPMENT_DIRECTORY` Verzeichnis hinzufügen können, finden Sie unterBeispiele für Feature-Flags für den lokalen Entwicklungsmodus des AWS AppConfig Agenten.
> - (Optional) Sie können den Inhaltstyp, den der Agent für Ihre Konfigurationsdaten zurückgibt, anhand der Erweiterung, die Sie der Datei geben, steuern. Wenn Sie die Datei beispielsweise mit der Erweiterung.json benennen, gibt der Agent den Inhaltstyp zurück, `application/json` wenn Ihre Anwendung ihn anfordert. Wenn Sie die Erweiterung weglassen, verwendet der Agent sie `application/octet-stream` für den Inhaltstyp. Wenn Sie eine genaue Steuerung benötigen, können Sie eine Erweiterung im Format `.`*`type`*`%`*`subtype`* bereitstellen. Der Agent gibt einen Inhaltstyp von zurück`type/subtype`.

5. Führen Sie den folgenden Befehl aus, um den Agenten neu zu starten und die Konfigurationsdaten anzufordern.

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

Der Agent sucht in dem für den Agenten angegebenen Abfrageintervall nach Änderungen an der lokalen Datei. Wenn das Abfrageintervall nicht angegeben ist, verwendet der Agent das Standardintervall von 45 Sekunden. Diese Überprüfung im Abfrageintervall stellt sicher, dass sich der Agent in einer lokalen Entwicklungsumgebung genauso verhält wie bei der Konfiguration für die Interaktion mit dem AWS AppConfig Dienst.

> **Note**
>
> Um eine neue Version einer lokalen Entwicklungskonfigurationsdatei bereitzustellen, aktualisieren Sie die Datei mit neuen Daten.

Beispiele für Feature-Flags für den lokalen Entwicklungsmodus des AWS AppConfig Agenten

Dieser Abschnitt enthält Beispiele für Feature-Flags, die Sie mit AWS AppConfig Agent im lokalen Entwicklungsmodus verwenden können. Im lokalen Entwicklungsmodus werden Feature-Flag-Daten im Abrufzeitformat der Daten erwartet. Das Abrufzeitformat ist das Format, das zurückgegeben wird, wenn das Flag von der GetLatestConfigurationAPI abgerufen wird. Es enthält nur den Wert des Flags. Das Abrufzeitformat beinhaltet nicht die vollständige Definition eines Flags (wie sie an die API übergeben wurde). CreateHostedConfigurationVersion Die vollständige Definition eines Flags enthält auch Informationen wie Attributnamen und -werte, Einschränkungen und den aktivierten Status des Flags.

Themen

- Beispiele für Flaggen mit grundlegenden Funktionen
- Beispiele für Merkmalsflaggen mit mehreren Varianten

Beispiele für Flaggen mit grundlegenden Funktionen

Verwenden Sie die folgenden Beispiele für grundlegende Feature-Flags mit AWS AppConfig Agent im lokalen Entwicklungsmodus.

> **Note**
>
> Wenn Sie möchten, dass der Agent den Inhaltstyp Ihrer lokalen Feature-Flag-Daten als meldet `application/json` (wie beim Abrufen von Flag-Daten aus einer Umgebung, die sich nicht AWS AppConfig im lokalen Entwicklungsmodus befindet), müssen Ihre lokalen Feature-Flag-Dateien die Erweiterung.json verwenden. Beispiel, `Local:MyFeatureFlags:SampleB1.json`.

Beispiel 1: Ein einzelnes Flag, das eine Aktualisierung der Benutzeroberfläche darstellt.

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

Beispiel 2: Mehrere Flags stehen für betriebsbereite Feature-Flags.

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

Beispiele für Merkmalsflaggen mit mehreren Varianten

Das Abrufzeitformat einer Feature-Flag-Konfiguration, die mindestens ein Feature-Flag mit mehreren Varianten enthält, wird als Amazon Ion-Daten statt als JSON-Daten dargestellt. In diesem Format werden Flags mit mehreren Varianten als Liste mit Anmerkungen dargestellt, und grundlegende Flags werden als kommentierte Zeichenfolge dargestellt. Die Listenelemente eines Flags mit mehreren Varianten sind entweder ein Tupel (eine Liste mit einer Länge von zwei), das eine einzelne Variante darstellt, oder eine Zeichenfolge, die die Standardvariante darstellt. Innerhalb eines Variantentupels ist das erste Element ein S-Ausdruck, der die Regel der Variante darstellt, und das zweite Element ist eine Zeichenfolge, die den Inhalt der Variante darstellt.

Damit der Agent diese Dateien richtig interpretieren kann, müssen Ihre lokalen Feature-Flag-Dateien die folgende Erweiterung haben:. `application%ion%type=AWS.AppConfig.FeatureFlags`. Beispiel, `Local:MyFeatureFlags:SampleMV1.application%ion %type=AWS.AppConfig.FeatureFlags`.

Beispiel 1: Eine Flagge mit mehreren Varianten, die eine gestaffelte Version einer neuen Funktion darstellt.

```
'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}'''
]
```

Beispiel 2: Mehrere Flags, die je nach Benutzer-ID unterschiedliche UX-Displays repräsentieren. Die ersten beiden Flaggen sind variantenreich und die letzte Flagge ist einfach.

```
'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 Überlegungen zur Nutzung von Browsern und Mobilgeräten

Mithilfe von Feature-Flags können Sie die Benutzererfahrung Ihrer Webseiten und mobilen Anwendungen im Handumdrehen aktualisieren, ohne den Aufwand, das Risiko oder die Starrheit einer App Store-Version. Mithilfe von Feature-Flags können Sie eine Änderung an Ihrer Benutzerbasis schrittweise zu einem Zeitpunkt Ihrer Wahl veröffentlichen. Wenn Sie auf einen Fehler stoßen, können Sie die Änderung sofort rückgängig machen, ohne dass Benutzer auf eine

neue Softwareversion aktualisieren müssen. Kurz gesagt, Feature-Flags bieten mehr Kontrolle und Flexibilität bei der Implementierung von Änderungen an Ihrer Anwendung.

In den folgenden Abschnitten werden wichtige Überlegungen zur Verwendung von AWS AppConfig Feature-Flags auf Webseiten und Mobilgeräten beschrieben.

Themen

- Konfigurationsdaten und Abrufen von Flaggen
- Authentifizierung und Amazon Cognito
- Caching
- Segmentierung
- Bandbreite (mobile Anwendungsfälle)
- Zusätzliche Anwendungsfälle kennzeichnen

Konfigurationsdaten und Abrufen von Flaggen

Für Browser- und mobile Anwendungsfälle entscheiden sich viele Kunden dafür, eine Proxyschicht zwischen dem Internet oder der mobilen Anwendung und AWS AppConfig zu verwenden. Dadurch wird Ihr AWS AppConfig Anrufvolumen von der Größe Ihrer Benutzerbasis entkoppelt, was die Kosten senkt. Außerdem können Sie den AWS AppConfig Agenten nutzen, der die Leistung beim Abrufen von Flaggen optimiert und Funktionen wie Flags mit mehreren Varianten unterstützt. AWS AppConfig empfiehlt die Verwendung AWS Lambda zur Erstellung des Proxys. Anstatt Flags direkt von abzurufen AWS AppConfig, konfigurieren Sie die AWS AppConfig Lambda-Erweiterung so, dass Ihre Feature-Flags innerhalb einer Lambda-Funktion abgerufen werden. Schreiben Sie die Funktion so, dass sie AWS AppConfig Abrufparameter aus der Ereignisanforderung akzeptiert und die entsprechenden Konfigurationsdaten in der Lambda-Antwort zurückgibt. Stellen Sie Ihren Proxy mithilfe der Lambda-Funktion URLs dem Internet zur Verfügung.

Nachdem Sie Ihren Proxy konfiguriert haben, sollten Sie die Häufigkeit berücksichtigen, mit der Sie Daten abrufen. Für mobile Anwendungen sind in der Regel keine Abfrageintervalle mit hoher Frequenz erforderlich. Konfigurieren Sie den AWS AppConfig Agenten so, dass Daten AWS AppConfig häufiger aktualisiert werden als Ihre Anwendung vom Proxy.

Authentifizierung und Amazon Cognito

Die Lambda-Funktion URLs unterstützt zwei Formen der Zugriffskontrolle, `AWS_IAM` und`NONE`. Verwenden Sie diese Option, `NONE` wenn Sie es vorziehen, Ihre eigene Authentifizierung und

Autorisierung in Ihrer Lambda-Funktion zu implementieren. `NONE`ist auch die empfohlene Option, wenn Ihr Anwendungsfall es ermöglicht, Ihren Endpunkt der Öffentlichkeit zugänglich zu machen und Ihre Konfigurationsdaten keine sensiblen Daten enthalten. Verwenden Sie für alle anderen Anwendungsfälle. `AWS_IAM`

> **Important**
>
> Wenn Sie Ihren Endpunkt ohne Authentifizierung dem Internet zugänglich machen, stellen Sie sicher, dass Ihre Konfigurationsdaten keine vertraulichen Daten wie personenbezogene Daten (PII) IDs, Benutzer- oder unveröffentlichte Feature-Namen preisgeben.

Wenn Sie sich für die Verwendung entscheiden`AWS_IAM`, müssen Sie die Anmeldeinformationen mit Amazon Cognito verwalten. Um mit Amazon Cognito zu beginnen, erstellen Sie einen Identitätspool. Ein Identitätspool ermöglicht es Ihnen, kurzfristige Anmeldeinformationen für Ihre Anwendung an authentifizierte Benutzer oder Gastbenutzer zu verkaufen. Sie müssen dem Identitätspool Rollen hinzufügen, die es Benutzern ermöglichen, die Funktion `InvokeFunctionUrl` for your Lambda zu verwenden. Auf diese Weise können Instanzen Ihrer Anwendung auf die Anmeldeinformationen zugreifen, die zum Abrufen Ihrer Konfigurationsdaten erforderlich sind.

Wenn Sie in Ihrer Anwendung mit Amazon Cognito arbeiten, sollten Sie die Verwendung von AWS Amplify. Amplify vereinfacht mobile/web Anwendungsinteraktionen mit Amazon Cognito AWS und bietet integrierten Support für Amazon Cognito.

Caching

Bei der Verwendung AWS AppConfig sollten Sie Ihre Konfigurationsdaten immer lokal auf dem Gerät oder im Browser zwischenspeichern. Das Caching bietet folgende Vorteile:

- Verbessert die Leistung, indem Latenz und Batterieverbrauch reduziert werden
- Bietet Stabilität, indem Abhängigkeiten vom Netzwerkzugriff beseitigt werden
- Senkt die Kosten, indem die Häufigkeit des Datenabrufs reduziert wird

Für mobile Anwendungsfälle empfehlen wir, speicherinterne und persistente Caches auf dem Gerät zu implementieren. Konfigurieren Sie Ihre Anwendung so, dass versucht wird, die gewünschte Konfiguration aus dem In-Memory-Cache abzurufen, und falls erforderlich, auf den Abruf von Ihrem Proxy zurückgreifen. Aktualisieren Sie nach erfolgreichem Abruf von Ihrem Proxy den In-Memory-

Cache und speichern Sie die Konfiguration anschließend auf dem Gerät. Verwenden Sie einen Hintergrundprozess, um den Cache zu durchlaufen und jede Konfiguration zu aktualisieren. Wenn die Konfiguration nach dem Start der Anwendung zum ersten Mal abgerufen wird und ein Abruf nicht erfolgreich ist, verwenden Sie die persistente Konfiguration (und verwenden Sie sie, um den In-Memory-Cache zu laden).

Segmentierung

Wenn Sie Feature-Flags verwenden, möchten Sie das Feature-Flagging-Erlebnis möglicherweise auf Ihren Kundenstamm aufteilen. Geben Sie dazu Kontext für Ihre Aufrufe zum Abrufen von Kennzeichnungen an und konfigurieren Sie Regeln, um je nach dem bereitgestellten Kontext verschiedene Varianten Ihrer Feature-Flags zurückzugeben. Beispielsweise haben Sie möglicherweise eine Feature-Flag-Variante für iOS 18.X-Benutzer, eine Variante für iOS 17.X-Benutzer und eine Standardflagge für alle anderen Versionen von iOS. Mit Varianten können Sie jede iOS-Version Ihrer Anwendung so konfigurieren, dass sie auf dieselbe Konfiguration in derselben Umgebung abzielt. Basierend auf dem im Abrufaufruf angegebenen Kontext (z. B. „Version“: „i OS18 .1") erhalten die Geräte jedoch die entsprechende Variante der Konfiguration.

> **Note**
>
> Wenn Sie AWS AppConfig Feature-Flag-Varianten für einen mobilen Anwendungsfall verwenden, müssen Sie den AWS AppConfig Agenten und einen Proxy zum Abrufen von Feature-Flags verwenden.

Wenn Sie den AWS AppConfig Agenten nicht zum Abrufen von Feature-Flags verwenden möchten, können Sie AWS AppConfig Umgebungen für eine einfache Segmentierung mit geringer Kardinalität nutzen. Eine Umgebung ist eine logische Bereitstellungsgruppe für Ihre Ziele. Sie können Ihre Konfigurationen nicht nur in Entwicklungs-, Test- und Produktionsumgebungen unterteilen, sondern auch Ihren Kundenstamm unterteilen, indem Sie für Mobilgeräte spezifische Umgebungen einrichten, wie z. B. Gerätetyp (Tablet oder Telefon) oder Betriebssystem-Hauptversionen. In separaten Umgebungen können Sie dieselben oder unterschiedliche Sätze von Konfigurationsdaten bereitstellen, um die speziellen Anforderungen Ihres Kundenstamms zu erfüllen.

Bandbreite (mobile Anwendungsfälle)

Generell sollten Sie versuchen, die Größe der einzelnen Flaggensätze klein zu halten. Mobile Anwendungsfälle sind in der Regel mit geringen Bandbreitenbeschränkungen verbunden. Wenn

Sie die Größe Ihrer Daten minimieren, können Sie für eine einheitliche Benutzererfahrung sorgen. Bedenken Sie außerdem, dass das Zwischenspeichern auf dem Gerät von entscheidender Bedeutung ist, da mobile Geräte häufig zwischen Umgebungen mit geringer und ohne Bandbreite betrieben werden. Anwendungscode, der problemlos fehlschlägt, wenn keine Konfigurationsdaten abgerufen werden können, ist ebenfalls von entscheidender Bedeutung.

Zusätzliche Anwendungsfälle kennzeichnen

Die Leistungsfähigkeit von Feature-Flags geht weit über den Komfort bei der Veröffentlichung von Funktionen hinaus. Seit langem bestehende Betriebsflags können verwendet werden, um den Betriebszustand Ihrer Anwendung zu verbessern. Sie können beispielsweise einen Schalter für die Leistungsüberwachung einrichten, der während eines Ereignisses zusätzliche Metriken und Debugging-Daten ausgibt. Alternativ können Sie die Aktualisierungsraten Ihrer Anwendungen für einen Teil Ihres Kundenstamms beibehalten und anpassen.

Konfigurationsdaten werden ohne AWS AppConfig Agent abgerufen

Die empfohlene Methode zum Abrufen von Konfigurationsdaten von AWS AppConfig ist die Verwendung des von Amazon entwickelten und verwalteten AWS AppConfig Agenten. Mit dem Agenten können Sie Konfigurationsdaten lokal zwischenspeichern und den AWS AppConfig Datenebenendienst asynchron nach Aktualisierungen abfragen. Dieser caching/polling Prozess stellt sicher, dass Ihre Konfigurationsdaten immer für Ihre Anwendung verfügbar sind, und minimiert gleichzeitig Latenz und Kosten. Wenn Sie den Agenten nicht verwenden möchten, können Sie die Öffentlichkeit APIs direkt vom AWS AppConfig Datenebenendienst aus anrufen.

Der Datenebenendienst verwendet zwei API-Aktionen StartConfigurationSessionund GetLatestConfiguration. Der Datenebenendienst verwendet außerdem separate Endpunkte als die AWS AppConfig Steuerungsebene.

> **Note**
>
> Der Datenebenendienst ersetzt den vorherigen Prozess des Abrufs von Konfigurationsdaten mithilfe der `GetConfiguration` API-Aktion. Die `GetConfiguration` API ist veraltet.

Funktionsweise

So funktioniert der Prozess des direkten Aufrufs AWS AppConfig APIs mithilfe des Datenebenendienstes.

Ihre Anwendung ruft Konfigurationsdaten ab, indem sie zunächst mithilfe der StartConfigurationSessionAPI-Operation eine Konfigurationssitzung einrichtet. Der Client Ihrer Sitzung ruft dann regelmäßig auf, um GetLatestConfigurationnach den neuesten verfügbaren Daten zu suchen und diese abzurufen.

Wenn Sie anrufen`StartConfigurationSession`, sendet Ihr Code die folgenden Informationen:

- Identifikatoren (ID oder Name) einer AWS AppConfig Anwendung, einer Umgebung und eines Konfigurationsprofils, das in der Sitzung verfolgt wird.
- (Optional) Die Mindestzeit, die der Client der Sitzung zwischen Aufrufen an `GetLatestConfiguration` warten muss.

AWS AppConfig Stellt als Antwort eine `InitialConfigurationToken` bereit, die dem Client der Sitzung übergeben und verwendet werden soll, wenn er diese Sitzung `GetLatestConfiguration` zum ersten Mal aufruft.

> **Important**
>
> Dieses Token sollte bei Ihrem ersten Aufruf von nur einmal verwendet werden`GetLatestConfiguration`. Sie müssen das neue Token in der `GetLatestConfiguration` Antwort (`NextPollConfigurationToken`) bei jedem nachfolgenden Aufruf von verwenden`GetLatestConfiguration`. Um Anwendungsfälle mit langen Umfragen zu unterstützen, sind die Token bis zu 24 Stunden gültig. Wenn ein `GetLatestConfiguration` Anruf ein abgelaufenes Token verwendet, kehrt das System zurück`BadRequestException`.

Wenn Sie anrufen`GetLatestConfiguration`, sendet Ihr Client-Code den neuesten `ConfigurationToken` Wert, den er hat, und empfängt ihn als Antwort:

- `NextPollConfigurationToken`: der `ConfigurationToken` Wert, der beim nächsten Aufruf von verwendet werden soll`GetLatestConfiguration`.
- `NextPollIntervalInSeconds`: Die Dauer, für die der Client warten soll, bevor er seinen nächsten Anruf tätigt`GetLatestConfiguration`.

- Die Konfiguration: Die neuesten Daten, die für die Sitzung vorgesehen sind. Dies kann leer sein, wenn der Client bereits über die neueste Version der Konfiguration verfügt.

> **Important**
>
> Notieren Sie die folgenden wichtigen Informationen.
>
> - Die StartConfigurationSessionAPI sollte nur einmal pro Anwendung, Umgebung, Konfigurationsprofil und Client aufgerufen werden, um eine Sitzung mit dem Dienst einzurichten. Dies erfolgt in der Regel beim Start Ihrer Anwendung oder unmittelbar vor dem ersten Abruf einer Konfiguration.
> - Wenn Ihre Konfiguration mithilfe von bereitgestellt wird`KmsKeyIdentifier`, muss Ihre Anforderung zum Empfang der Konfiguration die Berechtigung zum Aufrufen `kms:Decrypt` enthalten. Weitere Informationen finden Sie unter Decrypt in der AWS Key Management Service API-Referenz.
> - Der API-Vorgang, der zuvor zum Abrufen von Konfigurationsdaten verwendet wurde`GetConfiguration`, ist veraltet. Der `GetConfiguration` API-Vorgang unterstützt keine verschlüsselten Konfigurationen.

(Beispiel) Abrufen einer Konfiguration durch Aufrufen AWS AppConfig APIs

Das folgende AWS CLI Beispiel zeigt, wie Konfigurationsdaten mithilfe der AWS AppConfig Daten `StartConfigurationSession` - und `GetLatestConfiguration` API-Operationen abgerufen werden. Mit dem ersten Befehl wird eine Konfigurationssitzung gestartet. Dieser Aufruf beinhaltet die IDs (oder Namen) der AWS AppConfig Anwendung, die Umgebung und das Konfigurationsprofil. Die API gibt einen zurück, der zum Abrufen Ihrer Konfigurationsdaten `InitialConfigurationToken` verwendet wurde.

```
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
```

Das System gibt Informationen im folgenden Format zurück.

```
{
```

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

Verwenden Sie nach dem Start einer Sitzung InitialConfigurationTokenden Befehl to call, GetLatestConfigurationum Ihre Konfigurationsdaten abzurufen. Die Konfigurationsdaten werden in der `mydata.json` Datei gespeichert.

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

Der erste Aufruf von `GetLatestConfiguration` verwendet das `ConfigurationToken` erhaltene von`StartConfigurationSession`. Die folgenden Informationen werden zurückgegeben.

```
{
    "NextPollConfigurationToken" : next configuration token,
    "ContentType" : content type of configuration,
    "NextPollIntervalInSeconds" : 60
}
```

Nachfolgende Aufrufe von `GetLatestConfiguration` müssen `NextPollConfigurationToken` aus der vorherigen Antwort resultieren.

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

Important

Beachten Sie die folgenden wichtigen Details zum `GetLatestConfiguration` API-Vorgang:

- Die `GetLatestConfiguration` Antwort enthält einen `Configuration` Abschnitt, in dem die Konfigurationsdaten angezeigt werden. Der `Configuration` Abschnitt wird nur angezeigt, wenn das System neue oder aktualisierte Konfigurationsdaten findet. Wenn das System keine neuen oder aktualisierten Konfigurationsdaten findet, sind die `Configuration` Daten leer.
- Sie erhalten `ConfigurationToken` in jeder Antwort von ein neues`GetLatestConfiguration`.

- Wir empfehlen, die Abfragehäufigkeit Ihrer `GetLatestConfiguration`-API-Aufrufe basierend auf Ihrem Budget, der erwarteten Häufigkeit der Konfigurationsbereitstellungen und der Anzahl der Ziele für eine Konfiguration zu optimieren.

Erweiterung von AWS AppConfig Workflows mithilfe von Erweiterungen

Eine Erweiterung erweitert Ihre Fähigkeit, Logik oder Verhalten an verschiedenen Stellen während des AWS AppConfig Workflows zur Erstellung oder Bereitstellung einer Konfiguration einzufügen. Beispielsweise können Sie Erweiterungen verwenden, um die folgenden Arten von Aufgaben auszuführen (um nur einige zu nennen):

- Senden Sie eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS) -Thema, wenn ein Konfigurationsprofil bereitgestellt wird.
- Säubern Sie den Inhalt eines Konfigurationsprofils nach sensiblen Daten, bevor eine Bereitstellung beginnt.
- Erstelle oder aktualisiere ein Atlassian Jira-Problem, wenn eine Änderung an einem Feature-Flag vorgenommen wird.
- Füge Inhalte aus einem Service oder einer Datenquelle mit deinen Konfigurationsdaten zusammen, wenn du ein Deployment startest.
- Sichern Sie eine Konfiguration in einem Amazon Simple Storage Service (Amazon S3) -Bucket, wann immer eine Konfiguration bereitgestellt wird.

Sie können diese Arten von Aufgaben AWS AppConfig Anwendungen, Umgebungen und Konfigurationsprofilen zuordnen.

Inhalt

- AWS AppConfig Erweiterungen verstehen
- Mit erstellten Erweiterungen AWS arbeiten
- Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig

AWS AppConfig Erweiterungen verstehen

In diesem Thema werden Konzepte und Terminologie von AWS AppConfig Erweiterungen vorgestellt. Die Informationen werden im Kontext der einzelnen Schritte behandelt, die für die Einrichtung und Verwendung von AWS AppConfig Erweiterungen erforderlich sind.

Themen

- Schritt 1: Ermitteln Sie, was Sie mit Erweiterungen machen möchten
- Schritt 2: Ermitteln Sie, wann die Erweiterung ausgeführt werden soll
- Schritt 3: Erstellen Sie eine Erweiterungszuordnung
- Schritt 4: Stellen Sie eine Konfiguration bereit und überprüfen Sie, ob die Erweiterungsaktionen ausgeführt wurden

Schritt 1: Ermitteln Sie, was Sie mit Erweiterungen machen möchten

Möchtest du eine Benachrichtigung an einen Webhook erhalten, der jedes Mal, wenn eine AWS AppConfig Bereitstellung abgeschlossen ist, Nachrichten an Slack sendet? Möchten Sie ein Konfigurationsprofil in einem Amazon Simple Storage Service (Amazon S3) -Bucket sichern, bevor eine Konfiguration bereitgestellt wird? Möchten Sie die Konfigurationsdaten nach vertraulichen Informationen durchsuchen, bevor die Konfiguration bereitgestellt wird? Sie können Erweiterungen verwenden, um diese Art von Aufgaben und mehr auszuführen. Sie können benutzerdefinierte Erweiterungen erstellen oder die erstellten Erweiterungen verwenden, AWS die im Lieferumfang enthalten sind. AWS AppConfig

> **Note**
>
> In den meisten Anwendungsfällen müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine AWS Lambda Funktion erstellen, die alle in der Erweiterung definierten Berechnungen und Verarbeitungen durchführt. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.

Die folgenden erstellten AWS Erweiterungen können Ihnen helfen, Konfigurationsbereitstellungen schnell in andere Dienste zu integrieren. Sie können diese Erweiterungen in der AWS AppConfig Konsole verwenden oder indem Sie API-Aktionen für Erweiterungen direkt über das AWS CLI, AWS -Tools für PowerShell, oder das SDK aufrufen.

Erweiterung	Beschreibung
Amazon: CloudWatch Offensichtlich A/B-Tests	Diese Erweiterung ermöglicht es Ihrer Anwendung, Benutzersitzungen lokal Varianten zuzuweisen, anstatt den EvaluateFeatureVorgang aufzurufen. Weitere Informati

Erweiterung	Beschreibung
	onen finden Sie unter Verwenden der Amazon CloudWatch Evidenly-Erweiterung.
AWS AppConfig Bereitstellungsereignisse für EventBridge	Diese Erweiterung sendet Ereignisse an den EventBridge Standard-Event-Bus, wenn eine Konfiguration bereitgestellt wird.
AWS AppConfig Bereitstellungsereignisse für Amazon Simple Notification Service (Amazon SNS)	Diese Erweiterung sendet Nachrichten an ein Amazon SNS SNS-Thema, das Sie angeben, wenn eine Konfiguration bereitgestellt wird.
AWS AppConfig Bereitstellungsereignisse für Amazon Simple Queue Service (Amazon SQS)	Diese Erweiterung stellt Nachrichten in Ihre Amazon SQS SQS-Warteschlange, wenn eine Konfiguration bereitgestellt wird.
Integrationserweiterung — Atlassian Jira	Mit dieser Erweiterung kannst du Probleme erstellen und aktualisieren AWS AppConfig , wenn du Änderungen an einem Feature-Flag vornimmst.

Schritt 2: Ermitteln Sie, wann die Erweiterung ausgeführt werden soll

Eine Erweiterung definiert eine oder mehrere Aktionen, die sie während eines AWS AppConfig Workflows ausführt. Die AWS verfasste `AWS AppConfig deployment events to Amazon SNS` Erweiterung umfasst beispielsweise eine Aktion zum Senden einer Benachrichtigung an ein Amazon SNS SNS-Thema. Jede Aktion wird entweder aufgerufen, wenn Sie mit einem Prozess interagieren AWS AppConfig oder wenn ein Prozess in AWS AppConfig Ihrem Namen ausgeführt wird. Diese Punkte werden Aktionspunkte genannt. AWS AppConfig Erweiterungen unterstützen die folgenden Aktionspunkte:

PRE_*-Aktionspunkte: Für `PRE_*` Aktionspunkte konfigurierte Erweiterungsaktionen werden nach der Überprüfung der Anfrage angewendet, jedoch bevor die Aktivität AWS AppConfig ausgeführt wird, die dem Namen des Aktionspunkts entspricht. Diese Aktionsaufrufe werden gleichzeitig mit einer Anfrage verarbeitet. Wenn mehr als eine Anforderung gestellt wird, werden Aktionsaufrufe nacheinander ausgeführt. Beachten Sie auch, dass `PRE_*` Aktionspunkte den Inhalt einer Konfiguration empfangen

und ändern können. PRE_*Aktionspunkte können auch auf einen Fehler reagieren und verhindern, dass eine Aktion ausgeführt wird.

- PRE_CREATE_HOSTED_CONFIGURATION_VERSION
- PRE_START_DEPLOYMENT

ON_*-Aktionspunkte: Eine Erweiterung kann auch parallel zu einem AWS AppConfig Workflow ausgeführt werden, indem ein ON_* Aktionspunkt verwendet wird. ON_*Aktionspunkte werden asynchron aufgerufen. ON_*Aktionspunkte erhalten nicht den Inhalt einer Konfiguration. Wenn bei einer Erweiterung während eines ON_* Aktionspunkts ein Fehler auftritt, ignoriert der Dienst den Fehler und setzt den Workflow fort.

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

AT_*-Aktionspunkte: Für AT_* Aktionspunkte konfigurierte Erweiterungsaktionen werden synchron und parallel zu einem Workflow aufgerufen. AWS AppConfig Wenn bei einer Erweiterung während eines AT_* Aktionspunkts ein Fehler auftritt, stoppt der Dienst den Workflow und setzt die Bereitstellung zurück.

- AT_DEPLOYMENT_TICK

Schritt 3: Erstellen Sie eine Erweiterungszuordnung

Um eine Erweiterung zu erstellen oder eine AWS erstellte Erweiterung zu konfigurieren, definieren Sie die Aktionspunkte, die eine Erweiterung aufrufen, wenn eine bestimmte AWS AppConfig Ressource verwendet wird. Sie können sich beispielsweise dafür entscheiden, die AWS AppConfig deployment events to Amazon SNS Erweiterung auszuführen und Benachrichtigungen zu einem Amazon SNS SNS-Thema zu erhalten, wenn eine Konfigurationsbereitstellung für eine bestimmte Anwendung gestartet wird. Die Definition, welche Aktionspunkte eine Erweiterung für eine bestimmte AWS AppConfig Ressource aufrufen, wird als Erweiterungszuordnung bezeichnet. Eine

Erweiterungszuordnung ist eine bestimmte Beziehung zwischen einer Erweiterung und einer AWS AppConfig Ressource, z. B. einer Anwendung oder einem Konfigurationsprofil.

Eine einzelne AWS AppConfig Anwendung kann mehrere Umgebungen und Konfigurationsprofile enthalten. Wenn Sie einer Anwendung oder einer Umgebung eine Erweiterung zuordnen, AWS AppConfig ruft sie die Erweiterung für alle Workflows auf, die sich auf die Anwendungs- oder Umgebungsressourcen beziehen, sofern zutreffend.

Nehmen wir beispielsweise an, Sie haben eine AWS AppConfig Anwendung aufgerufen MobileApps , die ein Konfigurationsprofil namens AccessList enthält. Nehmen wir an, die MobileApps Anwendung umfasst Beta-, Integrations- und Produktionsumgebungen. Sie erstellen eine Erweiterungszuordnung für die AWS verfasste Amazon SNS SNS-Benachrichtigungserweiterung und ordnen die Erweiterung der Anwendung zu. MobileApps Die Amazon SNS SNS-Benachrichtigungserweiterung wird immer dann aufgerufen, wenn die Konfiguration für die Anwendung in einer der drei Umgebungen bereitgestellt wird.

> **Note**
>
> Sie müssen keine Erweiterung erstellen, um AWS erstellte Erweiterungen verwenden zu können, aber Sie müssen eine Erweiterungszuordnung erstellen.

Schritt 4: Stellen Sie eine Konfiguration bereit und überprüfen Sie, ob die Erweiterungsaktionen ausgeführt wurden

Wenn Sie eine Zuordnung erstellt haben, wenn eine gehostete Konfiguration erstellt oder eine Konfiguration bereitgestellt wird, wird AWS AppConfig die Erweiterung aufgerufen und die angegebenen Aktionen ausgeführt. Wenn beim Aufrufen einer Erweiterung während eines `PRE-*` Aktionspunkts im System ein Fehler auftritt, werden Informationen zu diesem Fehler AWS AppConfig zurückgegeben.

Mit erstellten Erweiterungen AWS arbeiten

AWS AppConfig umfasst die folgenden von AWS Autoren erstellten Erweiterungen. Diese Erweiterungen können Ihnen helfen, den AWS AppConfig Workflow in andere Dienste zu integrieren. Sie können diese Erweiterungen im AWS Management Console oder verwenden, indem Sie API-Aktionen für Erweiterungen direkt über das AWS CLI AWS -Tools für PowerShell, oder das SDK aufrufen.

Erweiterung	Beschreibung
Amazon: CloudWatch Offensichtlich A/B-Tests	Diese Erweiterung ermöglicht es Ihrer Anwendung, Benutzersitzungen lokal Varianten zuzuweisen, anstatt den EvaluateFeatureVorgang aufzurufen. Weitere Informationen finden Sie unter Verwenden der Amazon CloudWatch Evidenly-Erweiterung.
AWS AppConfig Bereitstellungsereignisse für EventBridge	Diese Erweiterung sendet Ereignisse an den EventBridge Standard-Event-Bus, wenn eine Konfiguration bereitgestellt wird.
AWS AppConfig Bereitstellungsereignisse für Amazon Simple Notification Service (Amazon SNS)	Diese Erweiterung sendet Nachrichten an ein Amazon SNS SNS-Thema, das Sie angeben, wenn eine Konfiguration bereitgestellt wird.
AWS AppConfig Bereitstellungsereignisse für Amazon Simple Queue Service (Amazon SQS)	Diese Erweiterung stellt Nachrichten in Ihre Amazon SQS SQS-Warteschlange, wenn eine Konfiguration bereitgestellt wird.
Integrationserweiterung — Atlassian Jira	Mit dieser Erweiterung kannst du Probleme erstellen und aktualisieren AWS AppConfig , wenn du Änderungen an einem Feature-Flag vornimmst.

Verwenden der Amazon CloudWatch Evidenly-Erweiterung

Sie können Amazon CloudWatch Evidently verwenden, um neue Funktionen sicher zu validieren, indem Sie sie während der Einführung der Funktion einem bestimmten Prozentsatz Ihrer Nutzer zur Verfügung stellen. Sie können die Leistung des neuen Feature überwachen, um zu entscheiden, wann Sie den Traffic für Ihre Benutzer erhöhen möchten. Dadurch senken Sie Risiken und erkennen unbeabsichtigtes Verhalten noch bevor Sie das Feature vollständig einführen. Sie können auch A/B-Experimente durchführen, um Features auf der Grundlage von Erkenntnissen und Daten zu gestalten.

Die AWS AppConfig Erweiterung für CloudWatch Evidently ermöglicht es Ihrer Anwendung, Benutzersitzungen lokal Varianten zuzuweisen, anstatt den EvaluateFeatureVorgang aufzurufen. Eine lokale Sitzung mindert die Latenz- und Verfügbarkeitsrisiken, die mit einem API-Aufruf einhergehen. Informationen zur Konfiguration und Verwendung der Erweiterung finden Sie unter Durchführen von Produkteinführungen und A/B-Experimenten mit CloudWatch Evidently im CloudWatch Amazon-Benutzerhandbuch.

EventBridge Erweiterung „ AWS AppConfig Deployment Events to Amazon“ verwenden

Bei der `AWS AppConfig deployment events to Amazon EventBridge` Erweiterung handelt es sich um eine AWS eigens erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung sendet bei jeder Bereitstellung einer Konfiguration Ereignisbenachrichtigungen an den EventBridge Standardereignisbus. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, AWS AppConfig sendet sie nach jedem Start, Ende und Rollback der Konfigurationsbereitstellung Ereignisbenachrichtigungen an den Event-Bus.

Wenn Sie mehr Kontrolle darüber haben möchten, welche Aktionspunkte EventBridge Benachrichtigungen senden, können Sie eine benutzerdefinierte Erweiterung erstellen und den EventBridge standardmäßigen Amazon-Ressourcennamen (ARN) für das URI-Feld für den Ereignisbus eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unterExemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.

> **Important**
>
> Diese Erweiterung unterstützt nur den EventBridge Standard-Event-Bus.

Verwenden Sie die Erweiterung

Um die `AWS AppConfig deployment events to Amazon EventBridge` Erweiterung zu verwenden, fügen Sie die Erweiterung zunächst einer Ihrer AWS AppConfig Ressourcen hinzu, indem Sie eine Erweiterungszuordnung erstellen. Sie erstellen die Zuordnung mithilfe der AWS AppConfig Konsole oder der CreateExtensionAssociationAPI-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN eines AWS AppConfig Anwendungs-, Umgebungs- oder Konfigurationsprofils an. Wenn Sie die Erweiterung einer Anwendung oder Umgebung zuordnen,

wird eine Ereignisbenachrichtigung für jedes Konfigurationsprofil gesendet, das in der angegebenen Anwendung oder Umgebung enthalten ist.

Wenn Sie die Zuordnung erstellt haben und eine Konfiguration für die angegebene AWS AppConfig Ressource bereitgestellt wird, wird die AWS AppConfig Erweiterung aufgerufen und Benachrichtigungen entsprechend den in der Erweiterung angegebenen Aktionspunkten gesendet.

> **Note**
>
> Diese Erweiterung wird von den folgenden Aktionspunkten aufgerufen:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Sie können die Aktionspunkte für diese Erweiterung nicht anpassen. Um verschiedene Aktionspunkte aufzurufen, können Sie Ihre eigene Erweiterung erstellen. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung mithilfe der AWS Systems Manager Konsole oder der AWS CLI zu erstellen.

So erstellen Sie eine Erweiterungszuordnung (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich AWS AppConfig aus.
3. Wählen Sie auf der Registerkarte Erweiterungen die Option Zur Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource unter Ressourcentyp einen AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource werden Sie AWS AppConfig aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispielereignis, an das gesendet wird EventBridge , wenn die Erweiterung aufgerufen wird.

```
{
   "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"
   }
}
```

Verwenden der AWS AppConfig Bereitstellungsereignisse für die Amazon SNS SNS-Erweiterung

Bei der `AWS AppConfig deployment events to Amazon SNS` Erweiterung handelt es sich um eine AWS eigens erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung veröffentlicht Nachrichten zu einem Amazon SNS SNS-Thema, wenn eine Konfiguration bereitgestellt

wird. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, AWS AppConfig veröffentlicht sie nach jedem Start, Ende und Rollback der Konfigurationsbereitstellung eine Nachricht zu diesem Thema.

Wenn Sie mehr Kontrolle darüber haben möchten, welche Aktionspunkte Amazon SNS SNS-Benachrichtigungen senden, können Sie eine benutzerdefinierte Erweiterung erstellen und ein Amazon SNS SNS-Thema Amazon Resource Name (ARN) für das URI-Feld eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unter. Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig

Verwenden der Erweiterung

In diesem Abschnitt wird beschrieben, wie Sie die `AWS AppConfig deployment events to Amazon SNS` Erweiterung verwenden.

Schritt 1: So konfigurieren AWS AppConfig , dass Nachrichten zu einem Thema veröffentlicht werden

Fügen Sie Ihrem Amazon SNS SNS-Thema Erteilung AWS AppConfig (`appconfig.amazonaws.com`) Veröffentlichungsberechtigungen () eine Zugriffskontrollrichtlinie hinzu (`sns:Publish`). Weitere Informationen finden Sie unter Beispielfälle für Amazon SNS SNS-Zugriffskontrolle.

Schritt 2: Erstellen Sie eine Erweiterungszuordnung

Hängen Sie die Erweiterung an eine Ihrer AWS AppConfig Ressourcen an, indem Sie eine Erweiterungszuordnung erstellen. Sie erstellen die Zuordnung mithilfe der AWS AppConfig Konsole oder der CreateExtensionAssociationAPI-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN eines AWS AppConfig Anwendungs-, Umgebungs- oder Konfigurationsprofils an. Wenn Sie die Erweiterung einer Anwendung oder Umgebung zuordnen, wird eine Benachrichtigung für jedes Konfigurationsprofil gesendet, das in der angegebenen Anwendung oder Umgebung enthalten ist. Wenn Sie die Zuordnung erstellen, müssen Sie einen Wert für den `topicArn` Parameter eingeben, der den ARN des Amazon SNS SNS-Themas enthält, das Sie verwenden möchten.

Wenn Sie die Zuordnung erstellt haben und eine Konfiguration für die angegebene AWS AppConfig Ressource bereitgestellt wird, wird die AWS AppConfig Erweiterung aufgerufen und Benachrichtigungen gemäß den in der Erweiterung angegebenen Aktionspunkten gesendet.

> **Note**
>
> Diese Erweiterung wird von den folgenden Aktionspunkten aufgerufen:

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

Sie können die Aktionspunkte für diese Erweiterung nicht anpassen. Um verschiedene Aktionspunkte aufzurufen, können Sie Ihre eigene Erweiterung erstellen. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung mithilfe der AWS Systems Manager Konsole oder der AWS CLI zu erstellen.

So erstellen Sie eine Erweiterungszuordnung (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich AWS AppConfig aus.
3. Wählen Sie auf der Registerkarte Erweiterungen die Option Zur Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource unter Ressourcentyp einen AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource werden Sie AWS AppConfig aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispiel für die Nachricht, die an das Amazon SNS SNS-Thema gesendet wird, wenn die Erweiterung aufgerufen wird.

```
{
    "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"
        }
    }
}
```

Verwenden der AWS AppConfig Bereitstellungsereignisse für die Amazon SQS SQS-Erweiterung

Bei der `AWS AppConfig deployment events to Amazon SQS` Erweiterung handelt es sich um eine AWS eigens erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung stellt Nachrichten immer dann in die Warteschlange Ihres Amazon Simple Queue Service (Amazon SQS), wenn eine Konfiguration bereitgestellt wird. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, stellt sie nach jedem Start, AWS AppConfig Ende und Rollback der Konfigurationsbereitstellung eine Nachricht in die Warteschlange.

Wenn Sie mehr Kontrolle darüber haben möchten, welche Aktionspunkte Amazon SQS SQS-Benachrichtigungen senden, können Sie eine benutzerdefinierte Erweiterung erstellen und eine Amazon SQS SQS-Warteschlange mit Amazon Resource Name (ARN) für das URI-Feld eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unter. Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig

Verwenden der Erweiterung

In diesem Abschnitt wird beschrieben, wie Sie die `AWS AppConfig deployment events to Amazon SQS` Erweiterung verwenden.

Schritt 1: Konfigurieren Sie die Konfiguration AWS AppConfig , um Nachrichten in die Warteschlange zu stellen

Fügen Sie Ihrer Amazon SQS SQS-Warteschlange eine Amazon SQS SQS-Richtlinie hinzu, die AWS AppConfig (`appconfig.amazonaws.com`) Sendeberechtigungen (`sqs:SendMessage`) erteilt. Weitere Informationen finden Sie unter Grundlegende Beispiele für Amazon SQS SQS-Richtlinien.

Schritt 2: Erstellen Sie eine Erweiterungszuordnung

Hängen Sie die Erweiterung an eine Ihrer AWS AppConfig Ressourcen an, indem Sie eine Erweiterungszuordnung erstellen. Sie erstellen die Zuordnung mithilfe der AWS AppConfig Konsole oder der CreateExtensionAssociationAPI-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN eines AWS AppConfig Anwendungs-, Umgebungs- oder Konfigurationsprofils an. Wenn Sie die Erweiterung einer Anwendung oder Umgebung zuordnen, wird eine Benachrichtigung für jedes Konfigurationsprofil gesendet, das in der angegebenen Anwendung oder Umgebung enthalten ist. Wenn Sie die Zuordnung erstellen, müssen Sie einen `Here` Parameter eingeben, der den ARN der Amazon SQS SQS-Warteschlange enthält, die Sie verwenden möchten.

Wenn Sie nach der Erstellung der Zuordnung eine Konfiguration für die angegebene AWS AppConfig Ressource erstellt oder bereitgestellt haben, wird die AWS AppConfig Erweiterung aufgerufen und Benachrichtigungen entsprechend den in der Erweiterung angegebenen Aktionspunkten gesendet.

> **Note**
>
> Diese Erweiterung wird von den folgenden Aktionspunkten aufgerufen:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`

Sie können die Aktionspunkte für diese Erweiterung nicht anpassen. Um verschiedene Aktionspunkte aufzurufen, können Sie Ihre eigene Erweiterung erstellen. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig.

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung mithilfe der AWS Systems Manager Konsole oder der AWS CLI zu erstellen.

So erstellen Sie eine Erweiterungszuordnung (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich AWS AppConfig aus.
3. Wählen Sie auf der Registerkarte Erweiterungen die Option Zur Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource unter Ressourcentyp einen AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource werden Sie AWS AppConfig aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispiel für die Nachricht, die an die Amazon SQS SQS-Warteschlange gesendet wird, wenn die Erweiterung aufgerufen wird.

```
{
   "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"
}
```

Verwendung der Atlassian Jira-Erweiterung für AWS AppConfig

Durch die Integration mit Atlassian Jira AWS AppConfig kannst du Probleme in der Atlassian-Konsole erstellen und aktualisieren, wenn du Änderungen an einem Feature-Flag in deinem für das angegebene Feature vornimmst. AWS-Konto AWS-Region Jedes Jira-Problem umfasst den Flag-Namen, die Anwendungs-ID, die Konfigurationsprofil-ID und die Flag-Werte. Nachdem Sie Ihre Flag-Änderungen aktualisiert, gespeichert und bereitgestellt haben, aktualisiert Jira die vorhandenen Probleme mit den Details der Änderung.

> **Note**
>
> Jira aktualisiert Probleme immer dann, wenn du ein Feature-Flag erstellst oder aktualisierst. Jira aktualisiert Probleme auch, wenn Sie ein Flag-Attribut auf untergeordneter Ebene aus einem Flag auf übergeordneter Ebene löschen. Jira zeichnet keine Informationen auf, wenn Sie ein Kennzeichen auf übergeordneter Ebene löschen.

Um die Integration zu konfigurieren, müssen Sie wie folgt vorgehen:

- Berechtigungen für die AWS AppConfig Jira-Integration konfigurieren
- Konfiguration der AWS AppConfig Jira-Integrationsanwendung

Berechtigungen für die AWS AppConfig Jira-Integration konfigurieren

Wenn du die AWS AppConfig Integration mit Jira konfigurierst, gibst du Anmeldeinformationen für einen Benutzer an. Insbesondere geben Sie die Zugriffsschlüssel-ID und den geheimen Schlüssel des Benutzers in der Anwendung AWS AppConfig for Jira ein. Dieser Benutzer erteilt Jira die Erlaubnis, mit ihm zu kommunizieren. AWS AppConfig AWS AppConfig verwendet diese Anmeldeinformationen einmal, um eine Verbindung zwischen AWS AppConfig und Jira herzustellen.

Die Anmeldeinformationen werden nicht gespeichert. Sie können die Zuordnung entfernen, indem Sie die Anwendung AWS AppConfig for Jira deinstallieren.

Für das Benutzerkonto ist eine Berechtigungsrichtlinie erforderlich, die die folgenden Aktionen umfasst:

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

Führen Sie die folgenden Aufgaben aus, um eine IAM-Berechtigungsrichtlinie und einen Benutzer für eine AWS AppConfig Jira-Integration zu erstellen:

Aufgaben

- Aufgabe 1: Erstellen Sie eine IAM-Berechtigungsrichtlinie für AWS AppConfig eine Jira-Integration
- Aufgabe 2: Erstellen Sie einen Benutzer für AWS AppConfig eine Jira-Integration

Aufgabe 1: Erstellen Sie eine IAM-Berechtigungsrichtlinie für AWS AppConfig eine Jira-Integration

Verwende das folgende Verfahren, um eine IAM-Berechtigungsrichtlinie zu erstellen, mit der Atlassian Jira kommunizieren kann. AWS AppConfig Wir empfehlen, dass du eine neue Richtlinie erstellst und diese Richtlinie an eine neue IAM-Rolle anfügst. Das Hinzufügen der erforderlichen Berechtigung zu einer bestehenden IAM-Richtlinie und -Rolle widerspricht dem Prinzip der geringsten Rechte und wird nicht empfohlen.

Um eine IAM-Richtlinie für eine Jira-Integration AWS AppConfig zu erstellen

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Richtlinien und dann Richtlinie erstellen.
3. Wählen Sie auf der Seite Richtlinie erstellen die Registerkarte JSON aus und ersetzen Sie den Standardinhalt durch die folgende Richtlinie. Ersetzen Sie in der folgenden Richtlinie, *`Region account_IDapplication_ID`*, und *`configuration_profile_ID`* durch Informationen aus Ihrer AWS AppConfig Feature-Flag-Umgebung.

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

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

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

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

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

```
    ]
}
```

4. Wählen Sie Weiter: Tags aus.
5. (Optional) Fügen Sie ein oder mehrere Tag (Markierung)-Schlüssel-Wert-Paare hinzu, um den Zugriff für diese Richtlinie zu organisieren, zu verfolgen oder zu steuern, und wählen Sie dann Next: Review (Nächster Schritt: Prüfen) aus.
6. Geben Sie auf der Seite Review policy (Richtlinie überprüfen) im Feld Name einen Namen ein, wie z. B **`AppConfigJiraPolicy`**, und geben Sie anschließend eine optionale Beschreibung ein.
7. Wählen Sie Richtlinie erstellen aus.

Aufgabe 2: Erstellen Sie einen Benutzer für AWS AppConfig eine Jira-Integration

Gehen Sie wie folgt vor, um einen Benutzer für eine Atlassian AWS AppConfig Jira-Integration zu erstellen. Nachdem du den Benutzer erstellt hast, kannst du die Zugriffsschlüssel-ID und den geheimen Schlüssel kopieren, die du nach Abschluss der Integration angeben wirst.

Um einen Benutzer für AWS AppConfig eine Jira-Integration zu erstellen

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Users (Benutzer) und dann Add User (Benutzer hinzufügen).
3. Geben Sie im Feld Benutzername einen Namen ein, z. B. **`AppConfigJiraUser`**
4. Wählen Sie unter AWS Anmeldeinformationstyp auswählen die Option Zugriffsschlüssel — Programmgesteuerter Zugriff aus.
5. Wählen Sie Weiter: Berechtigungen aus.
6. Wählen Sie auf der Seite „Berechtigungen festlegen“ die Option Bestehende Richtlinien direkt anhängen aus. Suchen und aktivieren Sie das Kontrollkästchen für die Richtlinie, in der Sie sie erstellt habenAufgabe 1: Erstellen Sie eine IAM-Berechtigungsrichtlinie für AWS AppConfig eine Jira-Integration, und wählen Sie dann Weiter: Tags aus.
7. Fügen Sie auf der Seite Tags hinzufügen (optional) ein oder mehrere Tag-Schlüssel-Wertepaare hinzu, um den Zugriff für diesen Benutzer zu organisieren, nachzuverfolgen oder zu kontrollieren. Wählen Sie Weiter: Prüfen aus.
8. Überprüfen Sie auf der Seite „Überprüfen“ die Benutzerdetails.
9. Wählen Sie Create user (Benutzer erstellen) aus. Das System zeigt die Zugriffsschlüssel-ID und den geheimen Schlüssel des Benutzers an. Laden Sie entweder die CSV-Datei herunter

oder kopieren Sie diese Anmeldeinformationen an einen anderen Speicherort. Sie geben diese Anmeldeinformationen an, wenn Sie die Integration konfigurieren.

Konfiguration der AWS AppConfig Jira-Integrationsanwendung

Gehen Sie wie folgt vor, um die erforderlichen Optionen in der Anwendung AWS AppConfig for Jira zu konfigurieren. Nachdem Sie dieses Verfahren abgeschlossen haben, erstellt Jira für jedes Feature-Flag in Ihrem AWS-Konto für das angegebene Feature ein neues Problem. AWS-Region Wenn Sie Änderungen an einem Feature-Flag in vornehmen AWS AppConfig, zeichnet Jira die Details in den vorhandenen Issues auf.

> **Note**
>
> Ein AWS AppConfig Feature-Flag kann mehrere Flaggenattribute auf untergeordneter Ebene enthalten. Jira erstellt für jedes Feature-Flag auf übergeordneter Ebene ein Problem. Wenn Sie ein Flag-Attribut auf untergeordneter Ebene ändern, können Sie die Details dieser Änderung im Jira-Problem für das Kennzeichen der übergeordneten Ebene einsehen.

Um die Integration zu konfigurieren

1. Melde dich im Atlassian Marketplace an.
2. Geben Sie **`AWS AppConfig`** das Suchfeld ein und drücken Sie die Eingabetaste.
3. Installieren Sie die Anwendung auf Ihrer Jira-Instanz.
4. Wähle in der Atlassian-Konsole Apps verwalten und dann Jira aus AWS AppConfig .
5. Wählen Sie Konfigurieren aus.
6. Wähle unter Konfigurationsdetails die Option Jira-Projekt und dann das Projekt aus, das du mit deinem Feature-Flag verknüpfen möchtest. AWS AppConfig
7. Wählen Sie und wählen Sie dann die Region aus AWS-Region, in der sich Ihr AWS AppConfig Feature-Flag befindet.
8. Geben Sie im Feld Anwendungs-ID den Namen der AWS AppConfig Anwendung ein, die Ihr Feature-Flag enthält.
9. Geben Sie im Feld Konfigurationsprofil-ID den Namen des AWS AppConfig Konfigurationsprofils für Ihr Feature-Flag ein.

10. Geben Sie in den Feldern Access Key ID und Secret Key die Anmeldeinformationen ein, die Sie kopiert habenAufgabe 2: Erstellen Sie einen Benutzer für AWS AppConfig eine Jira-Integration. Optional können Sie auch ein Sitzungstoken angeben.
11. Wählen Sie Absenden aus.
12. Wähle in der Atlassian-Konsole Projekte und dann das Projekt aus, das du für AWS AppConfig die Integration ausgewählt hast. Auf der Seite „Probleme“ wird für jedes Feature-Flag im angegebenen AWS-Konto und ein Problem angezeigt. AWS-Region

Löschen der Anwendung und der Daten AWS AppConfig für Jira

Wenn du die Jira-Integration mit AWS AppConfig Feature-Flags nicht mehr verwenden möchtest, kannst du die Anwendung AWS AppConfig für Jira in der Atlassian-Konsole löschen. Das Löschen der Integrationsanwendung hat folgende Auswirkungen:

- Löscht die Zuordnung zwischen Ihrer Jira-Instanz und AWS AppConfig
- Löscht Ihre Jira-Instanzdetails von AWS AppConfig

Um die Anwendung AWS AppConfig für Jira zu löschen

1. Wähle in der Atlassian-Konsole Apps verwalten aus.
2. Wähle AWS AppConfig für Jira.
3. Wählen Sie Deinstallieren.

Exemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig

Führen Sie die folgenden Aufgaben aus, um eine benutzerdefinierte AWS AppConfig Erweiterung zu erstellen. Jede Aufgabe wird in späteren Themen ausführlicher beschrieben.

> **Note**
>
> Beispiele für benutzerdefinierte AWS AppConfig Erweiterungen finden Sie unter GitHub:
>
> - Beispielerweiterung, die Bereitstellungen mit einem `blocked_day` Moratoriumskalender mithilfe von Systems Manager Change Calendar verhindert

- Beispielerweiterung, die verhindert, dass Geheimnisse mithilfe von Git-Secrets in Konfigurationsdaten gelangen
- Beispielerweiterung, die verhindert, dass personenbezogene Daten (PII) mit Amazon Comprehend in Konfigurationsdaten gelangen

1. Erstellen Sie eine Funktion AWS Lambda

   In den meisten Anwendungsfällen müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine AWS Lambda Funktion erstellen, die alle in der Erweiterung definierten Berechnungen und Verarbeitungen durchführt. Eine Ausnahme von dieser Regel ist, wenn Sie benutzerdefinierte Versionen der erstellten AWS Benachrichtigungserweiterungen erstellen, um Aktionspunkte hinzuzufügen oder zu entfernen. Weitere Informationen zu dieser Ausnahme finden Sie unter. Schritt 3: Erstellen Sie eine benutzerdefinierte AWS AppConfig Erweiterung

2. Konfigurieren Sie die Berechtigungen für Ihre benutzerdefinierte Erweiterung

   Um die Berechtigungen für Ihre benutzerdefinierte Erweiterung zu konfigurieren, können Sie einen der folgenden Schritte ausführen:

   - Erstellen Sie eine AWS Identity and Access Management (IAM) -Servicerolle, die `InvokeFunction` Berechtigungen beinhaltet.
   - Erstellen Sie mithilfe der AddPermissionLambda-API-Aktion eine Ressourcenrichtlinie.

   In dieser exemplarischen Vorgehensweise wird beschrieben, wie Sie die IAM-Servicerolle erstellen.

3. Erstellen Sie eine Erweiterung

   Sie können eine Erweiterung mithilfe der AWS AppConfig Konsole oder durch Aufrufen der CreateExtensionAPI-Aktion über das AWS CLI AWS -Tools für PowerShell, oder das SDK erstellen. Die exemplarische Vorgehensweise verwendet die Konsole.

4. Erstellen Sie eine Erweiterungszuordnung

   Sie können eine Erweiterungszuordnung mithilfe der AWS AppConfig Konsole oder durch Aufrufen der CreateExtensionAssociationAPI-Aktion über das AWS CLI AWS -Tools für PowerShell, oder das SDK erstellen. Die exemplarische Vorgehensweise verwendet die Konsole.

5. Führen Sie eine Aktion aus, die die Erweiterung aufruft

   Ruft nach dem Erstellen der Zuordnung AWS AppConfig die Erweiterung auf, wenn die durch die Erweiterung definierten Aktionspunkte für diese Ressource eintreten. Wenn Sie beispielsweise

eine Erweiterung zuordnen, die eine `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` Aktion enthält, wird die Erweiterung jedes Mal aufgerufen, wenn Sie eine neue Version der gehosteten Konfiguration erstellen.

In den Themen in diesem Abschnitt werden alle Aufgaben beschrieben, die beim Erstellen einer benutzerdefinierten AWS AppConfig Erweiterung anfallen. Jede Aufgabe wird im Kontext eines Anwendungsfalls beschrieben, in dem ein Kunde eine Erweiterung erstellen möchte, die eine Konfiguration automatisch in einem Amazon Simple Storage Service (Amazon S3) -Bucket sichert. Die Erweiterung wird immer dann ausgeführt, wenn eine gehostete Konfiguration erstellt (`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`) oder bereitgestellt (`PRE_START_DEPLOYMENT`) wird.

Themen

- Schritt 1: Erstellen Sie eine Lambda-Funktion für eine benutzerdefinierte Erweiterung AWS AppConfig
- Schritt 2: Konfigurieren Sie die Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung
- Schritt 3: Erstellen Sie eine benutzerdefinierte AWS AppConfig Erweiterung
- Schritt 4: Erstellen Sie eine Erweiterungszuordnung für eine benutzerdefinierte Erweiterung AWS AppConfig

Schritt 1: Erstellen Sie eine Lambda-Funktion für eine benutzerdefinierte Erweiterung AWS AppConfig

In den meisten Anwendungsfällen müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine AWS Lambda Funktion erstellen, die alle in der Erweiterung definierten Berechnungen und Verarbeitungen durchführt. Dieser Abschnitt enthält Beispielcode für eine Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung. Dieser Abschnitt enthält auch Referenzdetails für Payload-Anfragen und Antworten. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Getting started with Lambda im AWS Lambda Developer Guide.

Beispiel-Code

Der folgende Beispielcode für eine Lambda-Funktion sichert, wenn er aufgerufen wird, automatisch eine AWS AppConfig Konfiguration in einem Amazon S3 S3-Bucket. Die Konfiguration wird jedes Mal gesichert, wenn eine neue Konfiguration erstellt oder bereitgestellt wird. Das Beispiel verwendet

Erweiterungsparameter, sodass der Bucket-Name in der Lambda-Funktion nicht fest codiert werden muss. Mithilfe von Erweiterungsparametern kann der Benutzer die Erweiterung an mehrere Anwendungen anhängen und Konfigurationen in verschiedenen Buckets sichern. Das Codebeispiel enthält Kommentare zur näheren Erläuterung der Funktion.

Beispiel für eine Lambda-Funktion für eine Erweiterung 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)
```

Wenn Sie dieses Beispiel in dieser exemplarischen Vorgehensweise verwenden möchten, speichern Sie es mit dem Namen **`MyS3ConfigurationBackUpExtension`** und kopieren Sie den Amazon-Ressourcennamen (ARN) für die Funktion. Sie geben den ARN an, wenn Sie die AWS Identity and Access Management (IAM) -Annahme-Rolle im nächsten Abschnitt erstellen. Sie geben den ARN und den Namen an, wenn Sie die Erweiterung erstellen.

Payload-Referenz

Dieser Abschnitt enthält Referenzinformationen zu Payload-Anfragen und Antworten für die Arbeit mit benutzerdefinierten AWS AppConfig Erweiterungen.

Struktur der Anfrage

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 anfragen

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'
}
```

Asynchrone Ereignisse

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 der Antwort

Die folgenden Beispiele zeigen, was Ihre Lambda-Funktion als Antwort auf die Anfrage einer benutzerdefinierten AWS AppConfig Erweiterung zurückgibt.

PRE_* Synchrone Ereignisse — erfolgreiche Antwort

Wenn Sie den Inhalt transformieren möchten, verwenden Sie Folgendes:

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

AT_* Synchrone Ereignisse — erfolgreiche Antwort

Wenn Sie die nächsten Schritte einer Bereitstellung steuern möchten (eine Bereitstellung fortsetzen oder rückgängig machen), legen Sie die `Description` Attribute in der `Directive` Antwort fest.

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

`Directive`unterstützt zwei Werte: `CONTINUE` oder`ROLL_BACK`. Verwenden Sie diese Aufzählungen in Ihrer Payload-Antwort, um die nächsten Schritte einer Bereitstellung zu steuern.

Synchrone Ereignisse — erfolgreiche Reaktion

Wenn Sie den Inhalt transformieren möchten, verwenden Sie Folgendes:

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

Wenn Sie den Inhalt nicht transformieren möchten, geben Sie nichts zurück.

Asynchrone Ereignisse — erfolgreiche Antwort

Gib nichts zurück.

Alle Fehlerereignisse

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

Schritt 2: Konfigurieren Sie die Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung

Gehen Sie wie folgt vor, um eine AWS Identity and Access Management (IAM-) Dienstrolle zu erstellen und zu konfigurieren (oder eine Rolle anzunehmen). AWS AppConfig verwendet diese Rolle, um die Lambda-Funktion aufzurufen.

Um eine IAM-Servicerolle zu erstellen und zu erlauben AWS AppConfig , sie zu übernehmen

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option Benutzerdefinierte Vertrauensrichtlinie aus.
4. Fügen Sie die folgende JSON-Richtlinie in das Feld Benutzerdefinierte Vertrauensrichtlinie ein.

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

Wählen Sie Weiter aus.

5. Wählen Sie auf der Seite „Berechtigungen hinzufügen“ die Option Richtlinie erstellen aus. Die Seite Create policy (Richtlinie erstellen) wird in einer neuen Registerkarte geöffnet.
6. Wählen Sie die Registerkarte JSON und fügen Sie dann die folgende Berechtigungsrichtlinie in den Editor ein. Die `lambda:InvokeFunction` Aktion wird für `PRE_*` Aktionspunkte verwendet. Die `lambda:InvokeAsync` Aktion wird für `ON_*` Aktionspunkte verwendet. *`Your Lambda ARN`*Ersetzen Sie es durch den Amazon Resource Name (ARN) Ihres Lambda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "Your Lambda ARN"
    }
  ]
}
```

7. Wählen Sie Weiter: Tags aus.
8. Fügen Sie auf der Seite Tags hinzufügen (optional) ein oder mehrere Schlüssel-Wert-Paare hinzu und wählen Sie dann Weiter: Überprüfen aus.

9. Geben Sie auf der Seite Richtlinie überprüfen einen Namen und eine Beschreibung ein und wählen Sie dann Richtlinie erstellen aus.
10. Wählen Sie auf der Browser-Registerkarte für Ihre benutzerdefinierte Vertrauensrichtlinie das Aktualisierungssymbol aus und suchen Sie dann nach der Berechtigungsrichtlinie, die Sie gerade erstellt haben.
11. Aktivieren Sie das Kontrollkästchen für Ihre Berechtigungsrichtlinie und wählen Sie dann Weiter.
12. Geben Sie auf der Seite Name, Überprüfung und Erstellung einen Namen in das Feld Rollenname und anschließend eine Beschreibung ein.
13. Wählen Sie Create role (Rolle erstellen) aus. Das System leitet Sie zur Seite Rollen zurück. Wählen Sie im Banner die Option Rolle anzeigen aus.
14. Kopieren Sie den ARN. Sie geben diesen ARN an, wenn Sie die Erweiterung erstellen.

Schritt 3: Erstellen Sie eine benutzerdefinierte AWS AppConfig Erweiterung

Eine Erweiterung definiert eine oder mehrere Aktionen, die sie während eines AWS AppConfig Workflows ausführt. Die AWS verfasste `AWS AppConfig deployment events to Amazon SNS` Erweiterung umfasst beispielsweise eine Aktion zum Senden einer Benachrichtigung an ein Amazon SNS SNS-Thema. Jede Aktion wird entweder aufgerufen, wenn Sie mit einem Prozess interagieren AWS AppConfig oder wenn ein Prozess in AWS AppConfig Ihrem Namen ausgeführt wird. Diese Punkte werden Aktionspunkte genannt. AWS AppConfig Erweiterungen unterstützen die folgenden Aktionspunkte:

PRE_*-Aktionspunkte: Für `PRE_*` Aktionspunkte konfigurierte Erweiterungsaktionen werden nach der Überprüfung der Anfrage angewendet, jedoch bevor die Aktivität AWS AppConfig ausgeführt wird, die dem Namen des Aktionspunkts entspricht. Diese Aktionsaufrufe werden gleichzeitig mit einer Anfrage verarbeitet. Wenn mehr als eine Anforderung gestellt wird, werden Aktionsaufrufe nacheinander ausgeführt. Beachten Sie auch, dass `PRE_*` Aktionspunkte den Inhalt einer Konfiguration empfangen und ändern können. `PRE_*`Aktionspunkte können auch auf einen Fehler reagieren und verhindern, dass eine Aktion ausgeführt wird.

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`

ON_*-Aktionspunkte: Eine Erweiterung kann auch parallel zu einem AWS AppConfig Workflow ausgeführt werden, indem ein `ON_*` Aktionspunkt verwendet wird. `ON_*`Aktionspunkte werden

asynchron aufgerufen. `ON_*`Aktionspunkte erhalten nicht den Inhalt einer Konfiguration. Wenn bei einer Erweiterung während eines `ON_*` Aktionspunkts ein Fehler auftritt, ignoriert der Dienst den Fehler und setzt den Workflow fort.

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

AT_*-Aktionspunkte: Für `AT_*` Aktionspunkte konfigurierte Erweiterungsaktionen werden synchron und parallel zu einem Workflow aufgerufen. AWS AppConfig Wenn bei einer Erweiterung während eines `AT_*` Aktionspunkts ein Fehler auftritt, stoppt der Dienst den Workflow und setzt die Bereitstellung zurück.

- `AT_DEPLOYMENT_TICK`

> **Note**
>
> Der `AT_DEPLOYMENT_TICK` Aktionspunkt unterstützt die Integration der Überwachung durch Drittanbieter. `AT_DEPLOYMENT_TICK`wird während der Konfiguration, Bereitstellung, Verarbeitung und Orchestrierung aufgerufen. Wenn Sie eine Überwachungslösung eines Drittanbieters verwenden (z. B. Datadog), können Sie eine AWS AppConfig Erweiterung erstellen, die am `AT_DEPLOYMENT_TICK` Aktionspunkt nach Alarmen sucht und als Sicherheitsmaßnahme die Bereitstellung rückgängig macht, falls sie einen Alarm ausgelöst hat. Ein Codebeispiel für eine AWS AppConfig Erweiterung, die den `AT_DEPLOYMENT_TICK` Aktionspunkt zur Integration in Datadog verwendet, finden Sie unter aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub

Beispiel für eine Erweiterung

Die folgende Beispielerweiterung definiert eine Aktion, die den `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` Aktionspunkt aufruft. In dem `Uri` Feld gibt die Aktion den Amazon-Ressourcennamen (ARN) der `MyS3ConfigurationBackUpExtension` Lambda-Funktion an, die zuvor in dieser exemplarischen Vorgehensweise erstellt wurde. Die Aktion

gibt auch den ARN für die Übernahme der Rolle AWS Identity and Access Management (IAM) an, der weiter oben in dieser exemplarischen Vorgehensweise erstellt wurde.

Beispiel für eine Erweiterung AWS AppConfig

```
{
    "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**
>
> Informationen zur Anforderungssyntax und zu den Feldbeschreibungen beim Erstellen einer Erweiterung finden Sie unter dem CreateExtensionThema in der AWS AppConfig API-Referenz.

So erstellen Sie eine Erweiterung (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich AWS AppConfig aus.
3. Wählen Sie auf der Registerkarte Erweiterungen die Option Erweiterung erstellen aus.

4. Geben Sie unter Erweiterungsname einen eindeutigen Namen ein. Geben **`MyS3ConfigurationBackUpExtension`** Sie für die Zwecke dieser exemplarischen Vorgehensweise ein. Geben Sie optional eine Beschreibung ein.
5. Wählen Sie im Abschnitt Aktionen die Option Neue Aktion hinzufügen aus.
6. Geben Sie unter Aktionsname einen eindeutigen Namen ein. Geben **`PreCreateHostedConfigVersionActionForS3Backup`** Sie für die Zwecke dieser exemplarischen Vorgehensweise ein. Dieser Name beschreibt den Aktionspunkt, der von der Aktion verwendet wurde, und den Zweck der Erweiterung.
7. Wählen Sie in der Liste der Aktionspunkte PRE_CREATE_HOSTED_CONFIGURATION_VERSION aus.
8. Wählen Sie für Uri die Lambda-Funktion und dann die Funktion in der Lambda-Funktionsliste aus. Wenn Sie Ihre Funktion nicht sehen, vergewissern Sie sich, dass Sie sich in derselben AWS-Region Datei befinden, in der Sie die Funktion erstellt haben.
9. Wählen Sie für IAM-Rolle die Rolle aus, die Sie zuvor in dieser exemplarischen Vorgehensweise erstellt haben.
10. Wählen Sie im Abschnitt Erweiterungsparameter (optional) die Option Neuen Parameter hinzufügen aus.
11. Geben Sie unter Parametername einen Namen ein. Geben **`S3_BUCKET`** Sie für die Zwecke dieser exemplarischen Vorgehensweise ein.
12. Wiederholen Sie die Schritte 5—11, um eine zweite Aktion für den `PRE_START_DEPLOYMENT` Aktionspunkt zu erstellen.
13. Wählen Sie Erweiterung erstellen.

Anpassen der erstellten AWS Benachrichtigungserweiterungen

Sie müssen kein Lambda oder eine Erweiterung erstellen, um AWS verfasste Benachrichtigungserweiterungen zu verwenden. Sie können einfach eine Erweiterungszuordnung erstellen und dann einen Vorgang ausführen, der einen der unterstützten Aktionspunkte aufruft. Standardmäßig unterstützen die erstellten AWS Benachrichtigungserweiterungen die folgenden Aktionspunkte:

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

Wenn Sie benutzerdefinierte Versionen der `AWS AppConfig deployment events to Amazon SNS` Erweiterung und der `AWS AppConfig deployment events to Amazon SQS` Erweiterungen erstellen, können Sie die Aktionspunkte angeben, für die Sie Benachrichtigungen erhalten möchten.

> **Note**
>
> Die `AWS AppConfig deployment events to EventBridge` Erweiterung unterstützt die `PRE_*` Aktionspunkte nicht. Sie können eine benutzerdefinierte Version erstellen, wenn Sie einige der Standard-Aktionspunkte entfernen möchten, die der erstellten Version zugewiesen AWS sind.

Sie müssen keine Lambda-Funktion erstellen, wenn Sie benutzerdefinierte Versionen der erstellten AWS Benachrichtigungserweiterungen erstellen. Sie müssen nur einen Amazon-Ressourcennamen (ARN) in das `Uri` Feld für die neue Erweiterungsversion eingeben.

- Geben Sie für eine benutzerdefinierte EventBridge Benachrichtigungserweiterung den ARN der EventBridge Standardereignisse in das `Uri` Feld ein.
- Für eine benutzerdefinierte Amazon SNS SNS-Benachrichtigungserweiterung geben Sie den ARN eines Amazon SNS SNS-Themas in das `Uri` Feld ein.
- Für eine benutzerdefinierte Amazon SQS SQS-Benachrichtigungserweiterung geben Sie den ARN einer Amazon SQS SQS-Nachrichtenwarteschlange in das `Uri` Feld ein.

Schritt 4: Erstellen Sie eine Erweiterungszuordnung für eine benutzerdefinierte Erweiterung AWS AppConfig

Um eine Erweiterung zu erstellen oder eine AWS erstellte Erweiterung zu konfigurieren, definieren Sie die Aktionspunkte, die eine Erweiterung aufrufen, wenn eine bestimmte AWS AppConfig Ressource verwendet wird. Sie können sich beispielsweise dafür entscheiden, die `AWS AppConfig deployment events to Amazon SNS` Erweiterung auszuführen und Benachrichtigungen zu einem Amazon SNS SNS-Thema zu erhalten, wenn eine Konfigurationsbereitstellung für eine bestimmte Anwendung gestartet wird. Die Definition, welche Aktionspunkte eine Erweiterung für eine bestimmte AWS AppConfig Ressource aufrufen, wird als Erweiterungszuordnung bezeichnet. Eine Erweiterungszuordnung ist eine bestimmte Beziehung zwischen einer Erweiterung und einer AWS AppConfig Ressource, z. B. einer Anwendung oder einem Konfigurationsprofil.

Eine einzelne AWS AppConfig Anwendung kann mehrere Umgebungen und Konfigurationsprofile enthalten. Wenn Sie einer Anwendung oder einer Umgebung eine Erweiterung zuordnen, AWS AppConfig ruft sie die Erweiterung für alle Workflows auf, die sich auf die Anwendungs- oder Umgebungsressourcen beziehen, sofern zutreffend.

Nehmen wir beispielsweise an, Sie haben eine AWS AppConfig Anwendung aufgerufen MobileApps , die ein Konfigurationsprofil namens AccessList enthält. Nehmen wir an, die MobileApps Anwendung umfasst Beta-, Integrations- und Produktionsumgebungen. Sie erstellen eine Erweiterungszuordnung für die AWS verfasste Amazon SNS SNS-Benachrichtigungserweiterung und ordnen die Erweiterung der Anwendung zu. MobileApps Die Amazon SNS SNS-Benachrichtigungserweiterung wird immer dann aufgerufen, wenn die Konfiguration für die Anwendung in einer der drei Umgebungen bereitgestellt wird.

Gehen Sie wie folgt vor, um mithilfe der Konsole eine AWS AppConfig Erweiterungszuordnung zu erstellen. AWS AppConfig

So erstellen Sie eine Erweiterungszuordnung (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich AWS AppConfig aus.
3. Wählen Sie auf der Registerkarte Erweiterungen ein Optionsfeld für eine Erweiterung aus und klicken Sie dann auf Zur Ressource hinzufügen. Wählen Sie für die Zwecke dieser exemplarischen Vorgehensweise ConfigurationBackUpExtensionMyS3 aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource unter Ressourcentyp einen AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource werden Sie AWS AppConfig aufgefordert, andere Ressourcen auszuwählen. Wählen Sie für die Zwecke dieser exemplarischen Vorgehensweise die Option Anwendung aus.
5. Wählen Sie eine Anwendung in der Liste aus.
6. Vergewissern Sie sich im Abschnitt Parameter, dass S3_BUCKET im Feld Schlüssel aufgeführt ist. Fügen Sie in das Feld Wert den ARN der Lambda-Erweiterungen ein. Beispiel: `arn:aws:lambda:`*`aws-region`*`:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Wählen Sie Zuordnung zur Ressource erstellen aus.

Nachdem Sie die Zuordnung erstellt haben, können Sie die `MyS3ConfigurationBackUpExtension` Erweiterung aufrufen, indem Sie ein

neues Konfigurationsprofil erstellen, das `hosted` für ihre `SourceUri` Im Rahmen des Workflows zur Erstellung der neuen Konfiguration wird AWS AppConfig auf den `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` Aktionspunkt gestoßen. Wenn dieser Aktionspunkt erkannt wird, wird die `MyS3ConfigurationBackUpExtension` Erweiterung aufgerufen, die automatisch eine Sicherungskopie der neu erstellten Konfiguration in dem S3-Bucket erstellt, der im `Parameter` Abschnitt der Erweiterungszuordnung angegeben ist.

Verwenden von Codebeispielen zur Ausführung allgemeiner AWS AppConfig Aufgaben

Dieser Abschnitt enthält Codebeispiele für die programmgesteuerte Ausführung gängiger AWS AppConfig Aktionen. Wir empfehlen Ihnen, diese Beispiele mit Java und Python JavaScript SDKs zu verwenden und die Aktionen in einer Testumgebung durchzuführen. Dieser Abschnitt enthält ein Codebeispiel für die Bereinigung Ihrer Testumgebung, nachdem Sie fertig sind.

Themen

- Erstellen oder Aktualisieren einer Freiformkonfiguration, die im gehosteten Konfigurationsspeicher gespeichert ist
- Erstellen eines Konfigurationsprofils für ein in Secrets Manager gespeichertes Geheimnis
- Bereitstellen eines Konfigurationsprofils
- Verwenden von AWS AppConfig Agent zum Lesen eines Freiform-Konfigurationsprofils
- Verwenden AWS AppConfig des Agenten zum Lesen eines bestimmten Feature-Flags
- Verwenden von AWS AppConfig Agent zum Abrufen eines Feature-Flags mit Varianten
- Verwenden der GetLatestConfiguration API-Aktion zum Lesen eines Freiform-Konfigurationsprofils
- Säubere deine Umgebung

Erstellen oder Aktualisieren einer Freiformkonfiguration, die im gehosteten Konfigurationsspeicher gespeichert ist

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die durch den Code ausgeführt werden. Die Beispiele in diesem Abschnitt lauten wie folgt 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",
  })
);
```

Erstellen eines Konfigurationsprofils für ein in Secrets Manager gespeichertes Geheimnis

Jedes der folgenden Beispiele enthält Kommentare zu den vom Code ausgeführten Aktionen. Die Beispiele in diesem Abschnitt lauten wie folgt 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",
  })
);
```

Bereitstellen eines Konfigurationsprofils

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die durch den Code ausgeführt werden. Die Beispiele in diesem Abschnitt lauten wie folgt 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",
  })
);
```

Verwenden von AWS AppConfig Agent zum Lesen eines Freiform-Konfigurationsprofils

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die durch den Code ausgeführt werden.

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)
```

Verwenden AWS AppConfig des Agenten zum Lesen eines bestimmten Feature-Flags

Jedes der folgenden Beispiele enthält Kommentare zu den vom Code ausgeführten Aktionen.

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 }
```

Verwenden von AWS AppConfig Agent zum Abrufen eines Feature-Flags mit Varianten

Jedes der folgenden Beispiele enthält Kommentare zu den vom Code ausgeführten Aktionen.

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'
config_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 config_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);
```

Verwenden der GetLatestConfiguration API-Aktion zum Lesen eines Freiform-Konfigurationsprofils

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die durch den Code ausgeführt werden. Die Beispiele in diesem Abschnitt lauten wie folgt 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.
*/
class AppConfigApiRetriever {
    /* AWS AppConfig Data SDK client used to interact with the AWS AppConfig Data
 service.
     */
    private 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) {
           StartConfigurationSessionResponse session =
appConfigData.startConfigurationSession(req -> req
                   .applicationIdentifier("MyDemoApp")
                   .configurationProfileIdentifier("MyConfig")
                   .environmentIdentifier("Beta"));
           this.configurationToken = session.initialConfigurationToken();
       }

       /*
       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.
        */
       GetLatestConfigurationResponse response =
appConfigData.getLatestConfiguration(

GetLatestConfigurationRequest.builder().configurationToken(this.configurationToken).build()

       /*
       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.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;
    }
}
```

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:
    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:
           session = self.appconfigdata.start_configuration_session(
               ApplicationIdentifier='MyDemoApp',
               ConfigurationProfileIdentifier='MyConfig',
               EnvironmentIdentifier='Beta'
           )
           self.configuration_token = session['InitialConfigurationToken']

       # 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)

       # 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_from_api = response['Configuration'].read()
       if config_from_api:
           self.configuration = config_from_api
           print('Configuration contents have changed since the last
GetLatestConfiguration call, new contents = ' + str(self.configuration))
       else:
           print('GetLatestConfiguration returned an empty response because we
already have the latest configuration')
```

```
        return self.configuration
```

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;
    }

    /*
    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) {
            const session = await this.appconfigdata.send(
                new StartConfigurationSessionCommand({
                    ApplicationIdentifier: "MyDemoApp",
                    ConfigurationProfileIdentifier: "MyConfig",
                    EnvironmentIdentifier: "Beta"
                })
            );
            this.configurationToken = session.InitialConfigurationToken;
        }

        /*
        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.
        */
        const response = await this.appconfigdata.send(
            new GetLatestConfigurationCommand({
                ConfigurationToken: this.configurationToken
            })
        );

        /*
        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.transformToString();
        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;
    }
}
```

Säubere deine Umgebung

Wenn Sie eines oder mehrere der Codebeispiele in diesem Abschnitt ausgeführt haben, empfehlen wir Ihnen, eines der folgenden Beispiele zu verwenden, um die durch diese Codebeispiele erstellten AWS AppConfig Ressourcen zu finden und zu löschen. Die Beispiele in diesem Abschnitt lauten wie folgt 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})`)
  }
}
```

AWS AppConfig Löschschutz konfigurieren

AWS AppConfig bietet eine Kontoeinstellung, um zu verhindern, dass Benutzer versehentlich aktiv genutzte Umgebungen und Konfigurationsprofile löschen. AWS AppConfig überwacht Anrufe an GetLatestConfigurationGetConfigurationund verfolgt innerhalb eines 60-Minuten-Intervalls (Standardeinstellung), welche Konfigurationsprofile und Umgebungen in diesen Aufrufen enthalten waren. Jedes Konfigurationsprofil oder jede Umgebung, auf die innerhalb dieses Intervalls zugegriffen wurde, wird als aktiv betrachtet. Wenn Sie versuchen, ein aktives Konfigurationsprofil oder eine aktive Konfigurationsumgebung zu löschen, wird ein Fehler AWS AppConfig zurückgegeben. Bei Bedarf können Sie diesen Fehler umgehen, indem Sie den `DeletionProtectionCheck` Parameter verwenden. Weitere Informationen finden Sie unter Umgehen oder Erzwingen einer Löschschutzprüfung.

Konfigurieren Sie den Löschschutz über die Konsole

Gehen Sie wie folgt vor, um den Löschschutz mithilfe der AWS Systems Manager Konsole zu konfigurieren.

So konfigurieren Sie den Löschschutz (Konsole)

1. Öffnen Sie die AWS Systems Manager Konsole unter https://console.aws.amazon.com/systems-manager/appconfig/.
2. Wählen Sie im Navigationsbereich Settings (Einstellungen).
3. Verwenden Sie den Schalter, um den Löschschutz zu aktivieren oder zu deaktivieren.
4. Legen Sie für den Schutzzeitraum die Definition einer aktiven Ressource auf einen Wert zwischen 15 und 1440 Minuten fest.
5. Klicken Sie auf Apply (Anwenden).

Konfigurieren Sie den Löschschutz mit dem AWS CLI

Gehen Sie wie folgt vor, um den Löschschutz mithilfe von zu konfigurieren AWS CLI. Ersetzen Sie *`value`* die folgenden Befehle durch den Wert, den Sie in Ihrer Umgebung verwenden möchten.

> **Note**
>
> Bevor Sie beginnen, empfehlen wir Ihnen, auf die neueste Version von zu aktualisieren AWS CLI. Weitere Informationen finden Sie im AWS Command Line Interface Benutzerhandbuch unter Installation oder Aktualisierung AWS CLI auf die neueste Version von.

So konfigurieren Sie den Löschschutz (CLI)

1. Führen Sie den folgenden Befehl aus, um die aktuellen Löschschutzeinstellungen anzuzeigen.

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

2. Führen Sie den folgenden Befehl aus, um den Löschschutz zu aktivieren oder zu deaktivieren. Geben Sie `false` an, ob der Löschschutz deaktiviert oder aktiviert werden `true` soll.

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

3. Sie können das Standardintervall auf maximal 24 Stunden erhöhen. Führen Sie den folgenden Befehl aus, um ein neues Intervall anzugeben.

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

Umgehen oder Erzwingen einer Löschschutzprüfung

Um Ihnen bei der Verwaltung des Löschschutzes zu helfen, DeleteConfigurationProfile APIs enthalten die DeleteEnvironmentund einen Parameter namens`DeletionProtectionCheck`. Dieser Parameter unterstützt die folgenden Werte:

- `BYPASS`: Weist an, die Löschschutzprüfung AWS AppConfig zu umgehen und ein Konfigurationsprofil zu löschen, auch wenn der Löschschutz dies andernfalls verhindert hätte.
- `APPLY`: Weist an, dass die Löschschutzprüfung auch dann ausgeführt wird, wenn der Löschschutz auf Kontoebene deaktiviert ist. `APPLY`erzwingt außerdem, dass die Löschschutzprüfung für Ressourcen ausgeführt wird, die in der letzten Stunde erstellt wurden und die normalerweise von den Löschschutzprüfungen ausgeschlossen sind.

- `ACCOUNT_DEFAULT`: Die Standardeinstellung, die anweist, den in der `UpdateAccountSettings` API angegebenen Wert für den Löschschutz AWS AppConfig zu implementieren.

> **Note**
>
> Standardmäßig werden Konfigurationsprofile und Umgebungen, die in der letzten Stunde erstellt wurden, `DeletionProtectionCheck` übersprungen. Die Standardkonfiguration soll verhindern, dass der Löschschutz Tests und Demos beeinträchtigt, die kurzlebige Ressourcen erzeugen. Sie können dieses Verhalten überschreiben, indem Sie `DeletionProtectionCheck=APPLY` beim Aufrufen von oder übergeben. `DeleteEnvironment` `DeleteConfigurationProfile`

In der folgenden CLI-Komplettlösung wird anhand von Beispielbefehlen veranschaulicht, wie der `DeletionProtectionCheck` Parameter verwendet wird. Ersetzen Sie *`ID`* die folgenden Befehle durch die ID für Ihre AWS AppConfig Artefakte.

1. Rufen Sie eine bereitgestellte Konfiguration GetLatestConfigurationauf.

```
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. Warten Sie 60 Sekunden AWS AppConfig , bis registriert ist, dass die Konfiguration aktiv ist.
3. Führen Sie den folgenden Befehl aus, um die Umgebung aufzurufen DeleteEnvironmentund den Löschschutz anzuwenden.

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

   Der Befehl sollte den folgenden Fehler zurückgeben.

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

4. Führen Sie den folgenden Befehl aus, um den Löschschutz zu umgehen und die Umgebung zu löschen.

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

Sicherheit in AWS AppConfig

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einem Rechenzentrum und einer Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der AWS Dienste in der ausgeführt AWS Cloud werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Systems Manager, finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

AWS AppConfig ist ein Tool in AWS Systems Manager. Informationen zur Anwendung des Modells der gemeinsamen Verantwortung bei der Verwendung AWS AppConfig finden Sie unter Sicherheit in AWS Systems Manager. In diesem Abschnitt wird beschrieben, wie Systems Manager konfiguriert wird, um die Sicherheits- und Compliance-Ziele für zu erreichen AWS AppConfig.

Implementieren des Zugriffs mit geringsten Berechtigungen

Aus Sicherheitsgründen empfiehlt es sich, Identitäten die mindestens erforderlichen Berechtigungen zu gewähren, um bestimmte Aktionen mit bestimmten Ressourcen unter bestimmten Bedingungen durchzuführen. AWS AppConfig Der Agent bietet zwei Funktionen, mit denen der Agent auf das Dateisystem einer Instanz oder eines Containers zugreifen kann: Backup und Schreiben auf Festplatte. Wenn Sie diese Funktionen aktivieren, stellen Sie sicher, dass nur der AWS AppConfig Agent über Schreibberechtigungen für die angegebenen Konfigurationsdateien im Dateisystem verfügt. Stellen Sie außerdem sicher, dass nur die Prozesse, die zum Lesen aus diesen Konfigurationsdateien erforderlich sind, dazu in der Lage sind. Die Implementierung der geringstmöglichen Zugriffsrechte ist eine grundlegende Voraussetzung zum Reduzieren des

Sicherheitsrisikos und der Auswirkungen, die aufgrund von Fehlern oder böswilligen Absichten entstehen könnten.

Weitere Informationen zur Implementierung des Zugriffs mit den geringsten Rechten finden Sie unter SEC03-BP02 Grant Least Privilege Access im AWS Well-Architected Tool Benutzerhandbuch. Weitere Informationen zu den in diesem Abschnitt genannten AWS AppConfig Agent-Funktionen finden Sie unter. Verwenden eines Manifests, um zusätzliche Abruffunktionen zu aktivieren

Verschlüsselung der Daten im Ruhezustand für AWS AppConfig

AWS AppConfig bietet standardmäßig Verschlüsselung zum Schutz von Kundendaten im Speicher mithilfe von AWS-eigene Schlüssel.

AWS-eigene Schlüssel— AWS AppConfig verwendet diese Schlüssel standardmäßig, um Daten, die vom Service bereitgestellt und im AWS AppConfig Datenspeicher gehostet werden, automatisch zu verschlüsseln. Sie können ihre Verwendung nicht einsehen, verwalten AWS-eigene Schlüssel, verwenden oder überwachen. Sie müssen jedoch keine Maßnahmen ergreifen oder Programme zum Schutz der Schlüssel ändern, die zur Verschlüsselung Ihrer Daten verwendet werden. Weitere Informationen finden Sie unter AWS-eigene Schlüssel im AWS Key Management Service - Entwicklerhandbuch.

Sie können diese Verschlüsselungsebene zwar nicht deaktivieren oder einen anderen Verschlüsselungstyp auswählen, aber Sie können einen vom Kunden verwalteten Schlüssel angeben, der verwendet werden soll, wenn Sie im Datenspeicher gehostete AWS AppConfig Konfigurationsdaten speichern und wenn Sie Ihre Konfigurationsdaten bereitstellen.

Vom Kunden verwaltete Schlüssel — AWS AppConfig unterstützt die Verwendung eines symmetrischen, vom Kunden verwalteten Schlüssels, den Sie selbst erstellen, besitzen und verwalten, um der vorhandenen AWS-eigener Schlüssel eine zweite Verschlüsselungsebene hinzuzufügen. Da Sie die volle Kontrolle über diese Verschlüsselungsebene haben, können Sie beispielsweise folgende Aufgaben ausführen:

- Festlegung und Aufrechterhaltung wichtiger Richtlinien und Zuschüsse
- Festlegung und Aufrechterhaltung von IAM-Richtlinien
- Aktivieren und Deaktivieren wichtiger Richtlinien
- Kryptographisches Material mit rotierendem Schlüssel
- Hinzufügen von Tags

- Erstellen von Schlüsselaliasen
- Schlüssel für das Löschen von Schlüsseln planen

Weitere Informationen finden Sie unter Vom Kunden verwalteter Schlüssel im AWS Key Management Service Entwicklerhandbuch.

AWS AppConfig unterstützt vom Kunden verwaltete Schlüssel

AWS AppConfig bietet Unterstützung für die vom Kunden verwaltete Schlüsselverschlüsselung für Konfigurationsdaten. Für Konfigurationsversionen, die im AWS AppConfig gehosteten Datenspeicher gespeichert sind, können Kunden ein `KmsKeyIdentifier` entsprechendes Konfigurationsprofil einrichten. Jedes Mal, wenn mithilfe der `CreateHostedConfigurationVersion` API-Operation eine neue Version von Konfigurationsdaten erstellt wird, wird ein AWS KMS Datenschlüssel aus dem AWS AppConfig generiert, `KmsKeyIdentifier` um die Daten vor dem Speichern zu verschlüsseln. Wenn später auf die Daten zugegriffen wird, entweder während des `GetHostedConfigurationVersion` oder des `StartDeployment` API-Vorgangs, werden die Konfigurationsdaten anhand von Informationen über den generierten Datenschlüssel AWS AppConfig entschlüsselt.

AWS AppConfig bietet auch Unterstützung für die vom Kunden verwaltete Schlüsselverschlüsselung für bereitgestellte Konfigurationsdaten. Um Konfigurationsdaten zu verschlüsseln, können Kunden ihrer Bereitstellung eine `KmsKeyIdentifier` beifügen. AWS AppConfig generiert damit den AWS KMS Datenschlüssel`KmsKeyIdentifier`, um Daten bei der `StartDeployment` API-Operation zu verschlüsseln.

AWS AppConfig Verschlüsselter Zugriff

Verwenden Sie bei der Erstellung eines vom Kunden verwalteten Schlüssels die folgende Schlüsselrichtlinie, um sicherzustellen, dass der Schlüssel verwendet werden kann.

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

```
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
            ],
        "Resource": "*"
        }
 ]
```

Um gehostete Konfigurationsdaten mit einem vom Kunden verwalteten Schlüssel zu verschlüsseln, `CreateHostedConfigurationVersion` benötigt der Identity Calling die folgende Richtlinienerklärung, die einem Benutzer, einer Gruppe oder einer Rolle zugewiesen werden kann:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Wenn Sie ein Secrets Manager Manager-Geheimnis oder andere Konfigurationsdaten verwenden, die mit einem vom Kunden verwalteten Schlüssel verschlüsselt wurden, `kms:Decrypt` müssen Sie die Daten entschlüsseln und abrufen. `retrievalRoleArn`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:account_ID:configuration source/object"
    }
  ]
}
```

Beim Aufrufen der AWS AppConfig StartDeploymentAPI-Operation `StartDeployment` benötigt der Identitätsaufruf die folgende IAM-Richtlinie, die einem Benutzer, einer Gruppe oder einer Rolle zugewiesen werden kann:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:Region:account_ID:key_ID"
    }
  ]
}
```

Beim Aufrufen des AWS AppConfig GetLatestConfigurationAPI-Vorgangs `GetLatestConfiguration` benötigt der Identitätsaufruf die folgende Richtlinie, die einem Benutzer, einer Gruppe oder einer Rolle zugewiesen werden kann:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Verschlüsselungskontext

Ein Verschlüsselungskontext ist ein optionaler Satz von Schlüssel-Wert-Paaren, die zusätzliche kontextbezogene Informationen zu den Daten enthalten.

AWS KMS verwendet den Verschlüsselungskontext als zusätzliche authentifizierte Daten, um die authentifizierte Verschlüsselung zu unterstützen. Wenn Sie einen Verschlüsselungskontext in eine Anforderung zum Verschlüsseln von Daten einbeziehen, wird der Verschlüsselungskontext AWS KMS an die verschlüsselten Daten gebunden. Zur Entschlüsselung von Daten müssen Sie denselben Verschlüsselungskontext in der Anfrage übergeben.

AWS AppConfig Verschlüsselungskontext: AWS AppConfig verwendet bei allen AWS KMS kryptografischen Vorgängen für verschlüsselte gehostete Konfigurationsdaten und Bereitstellungen

einen Verschlüsselungskontext. Der Kontext enthält einen Schlüssel, der dem Datentyp entspricht, und einen Wert, der das spezifische Datenelement identifiziert.

Überwachen Sie Ihre Verschlüsselungsschlüssel für AWS

Wenn Sie einen vom AWS KMS Kunden verwalteten Schlüssel mit verwenden AWS AppConfig, können Sie Amazon CloudWatch Logs verwenden AWS CloudTrail , um Anfragen zu verfolgen, die AWS AppConfig an gesendet AWS KMS werden.

Das folgende Beispiel ist ein CloudTrail Ereignis zur Überwachung von AWS KMS VorgängenDecrypt, die aufgerufen werden AWS AppConfig , um auf Daten zuzugreifen, die mit Ihrem vom Kunden verwalteten Schlüssel verschlüsselt wurden:

```
{
    "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"
}
```

Zugriff AWS AppConfig über einen Schnittstellenendpunkt (AWS PrivateLink)

Sie können verwenden AWS PrivateLink , um eine private Verbindung zwischen Ihrer VPC und AWS AppConfig herzustellen. Sie können darauf zugreifen, AWS AppConfig als ob es in Ihrer VPC wäre, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder AWS Direct Connect eine Verbindung zu verwenden. Instances in Ihrer VPC benötigen für den Zugriff AWS AppConfig keine öffentlichen IP-Adressen.

Sie stellen diese private Verbindung her, indem Sie einen Schnittstellen-Endpunkt erstellen, der von AWS PrivateLink unterstützt wird. Wir erstellen eine Endpunkt-Netzwerkschnittstelle in jedem Subnetz, das Sie für den Schnittstellen-Endpunkt aktivieren. Hierbei handelt es sich um vom Anforderer verwaltete Netzwerkschnittstellen, die als Eingangspunkt für den Datenverkehr dienen, der für AWS AppConfig bestimmt ist.

Weitere Informationen finden Sie unter Zugriff auf AWS-Services über AWS PrivateLink im AWS PrivateLink -Leitfaden.

Überlegungen zu AWS AppConfig

Bevor Sie einen Schnittstellen-Endpunkt für einrichten AWS AppConfig, lesen Sie die Überlegungen im AWS PrivateLink Handbuch.

AWS AppConfig unterstützt das Aufrufen der `appconfig`AND-Dienste `appconfigdata` über den Schnittstellenendpunkt.

Einen Schnittstellen-Endpunkt für AWS AppConfig erstellen

Sie können einen Schnittstellenendpunkt für die AWS AppConfig Verwendung entweder der Amazon VPC-Konsole oder der AWS Command Line Interface (AWS CLI) erstellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

Erstellen Sie einen Schnittstellenendpunkt für die AWS AppConfig Verwendung der folgenden Servicenamen:

```
com.amazonaws.region.appconfig
```

```
com.amazonaws.region.appconfigdata
```

Wenn Sie privates DNS für den Schnittstellenendpunkt aktivieren, können Sie API-Anfragen an die AWS AppConfig Verwendung des standardmäßigen regionalen DNS-Namens stellen. Beispiel: `appconfig.us-east-1.amazonaws.com` und `appconfigdata.us-east-1.amazonaws.com`.

Erstellen einer Endpunktrichtlinie für Ihren Schnittstellen-Endpunkt

Eine Endpunktrichtlinie ist eine IAM-Ressource, die Sie an einen Schnittstellen-Endpunkt anfügen können. Die standardmäßige Endpunktrichtlinie ermöglicht den vollen Zugriff AWS AppConfig über den Schnittstellenendpunkt. Um den Zugriff zu kontrollieren, der AWS AppConfig von Ihrer VPC aus gewährt wird, fügen Sie dem Schnittstellenendpunkt eine benutzerdefinierte Endpunktrichtlinie hinzu.

Eine Endpunktrichtlinie gibt die folgenden Informationen an:

- Die Prinzipale, die Aktionen ausführen können (AWS-Konten, IAM-Benutzer und IAM-Rollen).
- Aktionen, die ausgeführt werden können
- Die Ressourcen, auf denen die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit Endpunktrichtlinien im AWS PrivateLink -Leitfaden.

Beispiel: VPC-Endpunktrichtlinie für Aktionen AWS AppConfig

Im Folgenden finden Sie ein Beispiel für eine benutzerdefinierte Endpunktrichtlinie. Wenn Sie diese Richtlinie an Ihren Schnittstellen-Endpunkt anhängen, gewährt sie allen Prinzipalen auf allen Ressourcen den Zugriff auf die aufgeführten AWS AppConfig -Aktionen.

```
{
   "Statement": [
      {
         "Principal": "*",
```

```
         "Effect": "Allow",
         "Action": [
            "appconfig:CreateApplication",
            "appconfig:CreateEnvironment",
            "appconfig:CreateConfigurationProfile",
            "appconfig:StartDeployment",
            "appconfig:GetLatestConfiguration"
            "appconfig:StartConfigurationSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Secrets Manager Schlüsselrotation

In diesem Abschnitt werden wichtige Sicherheitsinformationen zur AWS AppConfig Integration mit Secrets Manager beschrieben. Informationen zu Secrets Manager finden Sie unter Was ist AWS Secrets Manager? im AWS Secrets Manager Benutzerhandbuch.

Einrichtung der automatischen Rotation von Secrets Manager Manager-Geheimnissen, bereitgestellt von AWS AppConfig

Rotation ist der Vorgang, bei dem ein im Secrets Manager gespeichertes Geheimnis regelmäßig aktualisiert wird. Wenn Sie ein Secret rotieren, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank oder im Service aktualisiert. Sie können die automatische Rotation von Geheimnissen in Secrets Manager konfigurieren, indem Sie eine AWS Lambda Funktion verwenden, um das Geheimnis und die Datenbank zu aktualisieren. Weitere Informationen finden Sie im AWS Secrets Manager Benutzerhandbuch unter Rotation von AWS Secrets Manager Geheimnissen.

Um die Schlüsselrotation von Secrets Manager Manager-Geheimnissen zu aktivieren, die von bereitgestellt wurden AWS AppConfig, aktualisieren Sie Ihre Rotations-Lambda-Funktion und stellen Sie das rotierte Geheimnis bereit.

> **Note**
>
> Stellen Sie Ihr AWS AppConfig Konfigurationsprofil bereit, nachdem Ihr Secret rotiert und vollständig auf die neue Version aktualisiert wurde. Sie können feststellen, ob das Geheimnis rotiert wurde, weil `VersionStage` sich der Status von `AWSPENDING` zu ändert`AWSCURRENT`.

Der Abschluss der geheimen Rotation erfolgt in der Secrets Manager `finish_secret` Manager-Funktion „Rotationsvorlagen“.

Hier ist eine Beispielfunktion, die eine AWS AppConfig Bereitstellung startet, nachdem ein Geheimnis rotiert wurde.

```
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))
```

Überwachung AWS AppConfig

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS AppConfig anderen AWS Lösungen. AWS bietet die folgenden Überwachungstools, mit denen Sie beobachten AWS AppConfig, melden können, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen ergreifen können:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.
- Amazon EventBridge kann verwendet werden, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse aus AWS Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.

Themen

- Protokollieren von AWS AppConfig API-Aufrufen mit AWS CloudTrail

- Protokollierung von Metriken für AWS AppConfig Datenebenenanrufe
- Bereitstellungen im Hinblick auf automatisches Rollback überwachen

Protokollieren von AWS AppConfig API-Aufrufen mit AWS CloudTrail

AWS AppConfig ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle API-Aufrufe AWS AppConfig als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS AppConfig Konsole und Codeaufrufen für die AWS AppConfig API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, an die die Anfrage gestellt wurde AWS AppConfig, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS Management Console sind regionsübergreifend.

Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter Erstellen eines Trails für Ihr AWS-Konto und Erstellen eines Trails für eine Organisation im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3 – Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur Preisgestaltung finden Sie unter CloudTrail AWS CloudTrail Preisgestaltung.

AWS AppConfig Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden (z. B. das Abrufen der zuletzt bereitgestellten Konfiguration durch Aufrufen GetLatestConfiguration). Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Protokolliert standardmäßig CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für die AWS AppConfig Ressourcentypen mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Die Tabelle in diesem Abschnitt zeigt die verfügbaren Ressourcentypen für AWS AppConfig.

- Um Datenereignisse mithilfe der CloudTrail Konsole zu protokollieren, erstellen Sie einen Trail - oder Ereignisdatenspeicher, um Datenereignisse zu protokollieren, oder aktualisieren Sie einen vorhandenen Trail- oder Ereignisdatenspeicher, um Datenereignisse zu protokollieren.

  1. Wählen Sie Datenereignisse aus, um Datenereignisse zu protokollieren.
  2. Wählen Sie aus der Liste Datenereignistyp die Option AWS AppConfig.
  3. Wählen Sie die Protokollauswahlvorlage aus, die Sie verwenden möchten. Sie können alle Datenereignisse für den Ressourcentyp protokollieren, alle `readOnly` Ereignisse protokollieren, alle `writeOnly` Ereignisse protokollieren oder eine benutzerdefinierte Protokollauswahlvorlage erstellen, um nach den Feldern `readOnly`eventName, und `resources.`ARN zu filtern.
  4. Geben Sie als Namen des Selektors den Wert ein. AppConfigDataEvents Informationen zur Aktivierung von Amazon CloudWatch Logs für Ihren Datenereignis-Trail finden Sie unterProtokollierung von Metriken für AWS AppConfig Datenebenenanrufe.

- Um Datenereignisse mit dem zu protokollieren AWS CLI, konfigurieren Sie den `--advanced-event-selectors` Parameter so, dass das `eventCategory` Feld dem Ressourcentypwert entspricht `Data` und das `resources.type` Feld dem Ressourcentypwert entspricht (siehe Tabelle). Sie können Bedingungen hinzufügen, um nach den Werten der `resources.`ARN Felder `readOnly`eventName, und zu filtern.

  - Um einen Trail zum Protokollieren von Datenereignissen zu konfigurieren, führen Sie den put-event-selectorsBefehl. Weitere Informationen finden Sie unter Protokollieren von Datenereignissen für Pfade mit dem AWS CLI.
  - Um einen Ereignisdatenspeicher für die Protokollierung von Datenereignissen zu konfigurieren, führen Sie den create-event-data-storeBefehl zum Erstellen eines neuen Ereignisdatenspeichers zum Protokollieren von Datenereignissen, oder führen Sie den update-event-data-storeBefehl zum Aktualisieren eines vorhandenen Ereignisdatenspeichers. Weitere Informationen finden Sie unter Protokollieren von Datenereignissen für Ereignisdatenspeicher mit dem AWS CLI.

In der folgenden Tabelle sind die AWS AppConfig Ressourcentypen aufgeführt. In der Spalte Datenereignistyp (Konsole) wird der Wert angezeigt, der aus der Liste Datenereignistyp auf der CloudTrail Konsole ausgewählt werden kann. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie bei der Konfiguration erweiterter Event-Selektoren mithilfe von oder angeben würden. AWS CLI CloudTrail APIs In der CloudTrail Spalte APIs Protokollierte Daten werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail auf*
AWS AppConfig	`AWS::AppConfig::Configuration`	• GetLatestConfiguration • StartConfigurationSession

*Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den `resources.ARN` Felder`eventName`, und filtern`readOnly`, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Weitere Informationen zu diesen Feldern finden Sie unter AdvancedFieldSelector.

AWS AppConfig Managementereignisse in CloudTrail

AWS AppConfig protokolliert alle Operationen auf der AWS AppConfig Steuerungsebene als Verwaltungsereignisse. Eine Liste der Vorgänge auf der AWS AppConfig Steuerungsebene, bei denen eine AWS AppConfig Anmeldung erfolgt CloudTrail, finden Sie in der AWS AppConfig API-Referenz.

AWS AppConfig Beispiele für Ereignisse

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das den StartConfigurationSessionVorgang demonstriert.

```
{
      "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"
      }
    }
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalt.

Protokollierung von Metriken für AWS AppConfig Datenebenenanrufe

Wenn Sie für AWS CloudTrail die Protokollierung von AWS AppConfig Datenereignissen konfiguriert haben, können Sie Amazon CloudWatch Logs aktivieren, um Metriken für Aufrufe auf der AWS AppConfig Datenebene zu protokollieren. Anschließend können Sie Protokolldaten in CloudWatch Logs suchen und filtern, indem Sie einen oder mehrere Metrikfilter erstellen. Metrikfilter definieren die Begriffe und Muster, nach denen in Protokolldaten gesucht werden muss, wenn diese an CloudWatch Logs gesendet werden. CloudWatch Logs verwendet metrische Filter, um Protokolldaten in numerische CloudWatch Metriken umzuwandeln. Sie können Metriken grafisch darstellen oder sie mit einem Alarm konfigurieren.

Bevor Sie beginnen

Aktivieren Sie die Protokollierung von AWS AppConfig Datenereignissen in AWS CloudTrail. Das folgende Verfahren beschreibt, wie Sie die Metrikprotokollierung für einen vorhandenen AWS AppConfig Trail-In aktivieren CloudTrail. Hinweise zum Aktivieren der CloudTrail Protokollierung von AWS AppConfig Datenplanaufrufen finden Sie unterAWS AppConfig Datenereignisse in CloudTrail.

Gehen Sie wie folgt vor, um CloudWatch Logs zu aktivieren, um Metriken für Aufrufe auf der AWS AppConfig Datenebene zu protokollieren.

So aktivieren Sie CloudWatch Logs, um Metriken für Aufrufe auf der AWS AppConfig Datenebene zu protokollieren

1. Öffnen Sie die CloudTrail Konsole unter https://console.aws.amazon.com/cloudtrail/.
2. Wähle auf dem Armaturenbrett deinen AWS AppConfig Trail aus.
3. Wähle im Abschnitt CloudWatch Logs die Option Bearbeiten aus.
4. Wählen Sie Aktiviert.

5. Behalten Sie für den Namen der Protokollgruppe entweder den Standardnamen bei oder geben Sie einen Namen ein. Notieren Sie den Namen. Sie werden die Protokollgruppe später in der CloudWatch Logs-Konsole auswählen.
6. Geben Sie in Role name (Name der Rolle) einen Namen ein.
7. Wählen Sie Änderungen speichern.

Gehen Sie wie folgt vor, um eine Metrik und einen Metrikfilter für AWS AppConfig in CloudWatch Logs zu erstellen. Das Verfahren beschreibt, wie Sie einen Metrikfilter für Aufrufe von `operation` und (optional) Aufrufe von `operation` und erstellen`Amazon Resource Name (ARN)`.

Um eine Metrik und einen Metrikfilter für AWS AppConfig in CloudWatch Logs zu erstellen

1. Öffnen Sie die CloudWatch Konsole unterhttps://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Log groups (Protokollgruppen) aus.
3. Wählen Sie das Kontrollkästchen neben der AWS AppConfig Protokollgruppe aus.
4. Wählen Sie Aktionen und dann Metrikfilter erstellen.
5. Geben Sie unter Filtername einen Namen ein.
6. Geben Sie unter Filtermuster Folgendes ein:

```
{ $.eventSource = "appconfig.amazonaws.com" }
```

7. (Optional) Wählen Sie im Abschnitt Testmuster Ihre Protokollgruppe aus der Liste Zu testende Protokolldaten auswählen aus. Wenn CloudTrail keine Anrufe protokolliert wurden, können Sie diesen Schritt überspringen.
8. Wählen Sie Weiter.
9. Geben **`AWS AppConfig`** Sie für Metric Namespace den Wert ein.
10. Geben Sie bei Metric name den Metriknamen **`Calls`** ein.
11. Geben Sie für Metric value (Metrikwert) **`1`** ein.
12. Überspringen Sie den Standardwert und die Einheit.
13. Geben Sie als Namen der Dimension ein**`operation`**.
14. Geben Sie als Wert für Dimension den Wert ein**`$.eventName`**.

    (Optional) Sie können eine zweite Dimension eingeben, die den Amazon-Ressourcennamen (ARN) enthält, der den Anruf tätigt. Um eine zweite Dimension hinzuzufügen, geben Sie als

Dimensionsname Folgendes ein**`resource`**. Geben Sie als Wert für Dimension den Wert ein**`$.resources[0].ARN`**.

Wählen Sie Weiter.

15. Überprüfen Sie die Details des Filters und Erstellen Sie einen metrischen Filter.

(Optional) Sie können dieses Verfahren wiederholen, um einen neuen Metrikfilter für einen bestimmten Fehlercode wie zu erstellen AccessDenied. Geben Sie in diesem Fall die folgenden Details ein:

1. Geben Sie unter Filtername einen Namen ein.
2. Geben Sie unter Filtermuster Folgendes ein:

```
{ $.errorCode = "codename" }
```

Beispiel

```
{ $.errorCode = "AccessDenied" }
```

3. Geben **`AWS AppConfig`** Sie für Metric Namespace den Wert ein.
4. Geben Sie bei Metric name den Metriknamen **`Errors`** ein.
5. Geben Sie für Metric value (Metrikwert) **`1`** ein.
6. Geben Sie als Standardwert eine Null (0) ein.
7. Einheit, Abmessungen und Alarme überspringen.

Nachdem Sie API-Aufrufe CloudTrail protokolliert haben, können Sie die Metriken unter anzeigen CloudWatch. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch unter Ihre Messwerte und Logs in der Konsole anzeigen. Informationen darüber, wie Sie eine von Ihnen erstellte Metrik finden, finden Sie unter Nach verfügbaren Metriken suchen.

> **Note**
>
> Wenn Sie die Fehlermetrik ohne Dimension einrichten, wie hier beschrieben, können Sie diese Metriken auf der Seite Metriken ohne Dimension anzeigen.

Einen Alarm für eine CloudWatch Metrik erstellen

Nachdem Sie Metriken erstellt haben, können Sie Metrikalarme in erstellen CloudWatch. Sie können beispielsweise einen Alarm für die AWS AppConfig Anrufmetrik erstellen, die Sie im vorherigen Verfahren erstellt haben. Insbesondere können Sie einen Alarm für Aufrufe der AWS AppConfig `StartConfigurationSession` API-Aktion erstellen, die einen Schwellenwert überschreiten. Informationen zum Erstellen eines Alarms für eine Metrik finden Sie unter Erstellen eines CloudWatch Alarms auf der Grundlage eines statischen Schwellenwerts im CloudWatch Amazon-Benutzerhandbuch. Informationen zu Standardgrenzwerten für Aufrufe der AWS AppConfig Datenebene finden Sie unter Standardgrenzwerte für Datenebene in der Allgemeine Amazon Web Services-Referenz.

Bereitstellungen im Hinblick auf automatisches Rollback überwachen

Während einer Bereitstellung können Sie Situationen vermeiden, in denen fehlerhafte oder falsche Konfigurationsdaten Fehler in Ihrer Anwendung verursachen, indem Sie eine Kombination aus AWS AppConfig Bereitstellungsstrategien und automatischen Rollbacks auf der Grundlage von Amazon-Alarmen verwenden. CloudWatch Wenn nach der Konfiguration ein oder mehrere CloudWatch Alarme während einer Bereitstellung in den `INSUFFICIENT_DATA` Status `ALARM` oder wechseln, AWS AppConfig werden Ihre Konfigurationsdaten automatisch auf die vorherige Version zurückgesetzt, wodurch Anwendungsausfälle oder -fehler vermieden werden. Sie können eine Konfiguration auch rückgängig machen, indem Sie den StopDeploymentAPI-Vorgang aufrufen, während eine Bereitstellung noch läuft.

> **Important**
>
> Unterstützt bei erfolgreich abgeschlossenen Bereitstellungen AWS AppConfig auch das Zurücksetzen von Konfigurationsdaten auf eine frühere Version, indem der `AllowRevert` Parameter zusammen mit dem StopDeploymentAPI-Vorgang verwendet wird. Für einige Kunden garantiert das Zurücksetzen auf eine vorherige Konfiguration nach einer erfolgreichen Bereitstellung, dass die Daten dieselben sind wie vor der Bereitstellung. Beim Zurücksetzen werden auch Alarmanzeigen ignoriert, wodurch verhindert werden kann, dass ein Rollforward während eines Anwendungsnotfalls fortgesetzt wird. Weitere Informationen finden Sie unter Konfiguration rückgängig machen.

Um automatische Rollbacks zu konfigurieren, geben Sie den Amazon-Ressourcennamen (ARN) einer oder mehrerer CloudWatch Metriken im Feld CloudWatch Alarme an, wenn Sie eine AWS AppConfig Umgebung erstellen (oder bearbeiten). Weitere Informationen finden Sie unter Umgebungen für Ihre Anwendung erstellen in AWS AppConfig.

> **Note**
>
> Wenn Sie eine Überwachungslösung eines Drittanbieters (z. B. Datadog) verwenden, können Sie eine AWS AppConfig Erweiterung erstellen, die am `AT_DEPLOYMENT_TICK` Aktionspunkt nach Alarmen sucht und als Sicherheitsmaßnahme die Bereitstellung rückgängig macht, falls sie einen Alarm ausgelöst hat. Weitere Informationen AWS AppConfig zu Erweiterungen finden Sie unter. Erweiterung von AWS AppConfig Workflows mithilfe von Erweiterungen Weitere Informationen zu benutzerdefinierten Erweiterungen finden Sie unterExemplarische Vorgehensweise: Benutzerdefinierte Erweiterungen erstellen AWS AppConfig. Ein Codebeispiel für eine AWS AppConfig Erweiterung, die den `AT_DEPLOYMENT_TICK` Aktionspunkt zur Integration in Datadog verwendet, finden Sie unter aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub

Empfohlene Metriken zur Überwachung des automatischen Rollbacks

Welche Metriken Sie überwachen möchten, hängt von der Hardware und Software ab, die von Ihren Anwendungen verwendet wird. AWS AppConfig Kunden überwachen häufig die folgenden Messwerte. Eine vollständige Liste der empfohlenen Messwerte, gruppiert nach AWS-Service, finden Sie unter Empfohlene Alarme im CloudWatch Amazon-Benutzerhandbuch.

Nachdem Sie die Messwerte festgelegt haben, die Sie überwachen möchten, können Sie CloudWatch sie zur Konfiguration von Alarmen verwenden. Weitere Informationen finden Sie unter CloudWatchAmazon-Alarme verwenden.

Service	Metrik	Details
Amazon API Gateway	4 XXError	Dieser Alarm erkennt eine hohe Rate von clientsei tigen Fehlern. Dies kann auf ein Problem mit den Autorisierungs- oder Client-Anfrageparametern hinweisen

Service	Metrik	Details
		. Es könnte auch bedeuten, dass eine Ressource entfernt wurde oder dass ein Client eine Ressource anfordert, die nicht existiert. Erwägen Sie, Amazon CloudWatch Logs zu aktivieren und nach Fehlern zu suchen, die die 4XX-Fehler verursachen könnten. Erwägen Sie außerdem, detaillierte CloudWatch Metriken zu aktivieren, um diese Metrik pro Ressource und Methode anzuzeigen und die Fehlerquelle einzugren zen. Fehler können auch durch eine Überschreitung des konfigurierten Drosselun gslimits verursacht werden.
Amazon API Gateway	5XXError	Dieser Alarm hilft dabei, eine hohe Rate serverseitiger Fehler zu erkennen. Dies kann darauf hindeuten, dass im API-Backend, im Netzwerk oder bei der Integration zwischen dem API-Gateway und der Backend-API etwas nicht stimmt.

Service	Metrik	Details
Amazon API Gateway	Latency	Dieser Alarm erkennt eine hohe Latenz in einer Phase. Suchen Sie den `IntegrationLatency` -Metrikwert, um die Latenz des API-Backends zu überprüfen. Wenn die beiden Metriken größtenteils übereinstimmen, ist das API-Backend die Ursache für die höhere Latenz, und Sie sollten dort nach Problemen suchen. Erwägen Sie auch, CloudWatch Protokolle zu aktivieren und nach Fehlern zu suchen, die die hohe Latenz verursachen könnten.
Amazon EC2 Auto Scaling	GroupInServiceCapacity	Dieser Alarm hilft zu erkennen, wenn die Kapazität in der Gruppe unter der gewünschten Kapazität liegt, die für Ihre Arbeitslast erforderlich ist. Um Fehler zu beheben, überprüfen Sie Ihre Skalierungsaktivitäten auf Startfehler und stellen Sie sicher, dass Ihre gewünschte Kapazitätskonfiguration korrekt ist.

Service	Metrik	Details
Amazon EC2	CPUUtilization	Dieser Alarm hilft bei der Überwachung der CPU-Auslastung einer EC2 Instanz. Je nach Anwendung kann eine gleichbleibend hohe Auslastung normal sein. Wenn jedoch die Leistung beeinträchtigt ist und die Anwendung nicht durch Festplatten-I/O, Arbeitsspeicher oder Netzwerkressourcen eingeschränkt ist, kann eine ausgelastete CPU auf einen Ressourcengpass oder auf Probleme mit der Anwendungsleistung hinweisen.
Amazon ECS	CPUReservation	Dieser Alarm hilft Ihnen, eine hohe CPU-Reservierung des ECS-Clusters zu erkennen. Eine hohe CPU-Reservierung kann darauf hindeuten, dass dem Cluster nicht mehr genügend CPUs für die Aufgabe registrierte Kapazität zur Verfügung steht.

Service	Metrik	Details
Amazon ECS	HTTPCode_Target_5xx_Count	Dieser Alarm hilft Ihnen, eine hohe serverseitige Fehleranz ahl für den ECS-Service zu erkennen. Dies kann darauf hindeuten, dass Fehler vorliegen, die dazu führen, dass der Server Anfragen nicht bearbeiten kann.
Amazon EKS mit Container-Einblicken	node_cpu_utilization	Dieser Alarm hilft dabei, eine hohe CPU-Auslastung in Worker-Knoten des Amazon EKS-Clusters zu erkennen. Wenn die Auslastun g konstant hoch ist, kann dies darauf hindeuten, dass Sie Ihre Worker-Knoten durch Instances mit mehr CPU ersetzen müssen oder dass das System horizontal skaliert werden muss.
Amazon EKS mit Container-Einblicken	node_memory_utilization	Dieser Alarm hilft bei der Erkennung einer hohen Speicherauslastung in Worker-Knoten des Amazon EKS-Clusters. Wenn die Auslastun g konstant hoch ist, deutet dies möglicherweise darauf hin, dass Sie die Anzahl der Pod-Replikate skalieren oder Ihre Anwendung optimieren müssen.

Service	Metrik	Details
Amazon EKS mit Container-Einblicken	pod_cpu_utilization_over_pod_limit	Dieser Alarm hilft bei der Erkennung einer hohen CPU-Auslastung in Pods des Amazon EKS-Clusters. Wenn die Auslastung konstant hoch ist, deutet dies möglicherweise darauf hin, dass das CPU-Limit für den betroffenen Pod erhöht werden muss.
Amazon EKS mit Container-Einblicken	pod_memory_utilization_over_pod_limit	Dieser Alarm hilft bei der Erkennung einer hohen CPU-Auslastung in Pods des Amazon EKS-Clusters. Wenn die Auslastung konstant hoch ist, deutet dies möglicherweise darauf hin, dass das CPU-Limit für den betroffenen Pod erhöht werden muss.
AWS Lambda	Fehler	Dieser Alarm erkennt hohe Fehlerzahlen. Zu den Fehlern gehören die vom Code ausgelösten Ausnahmen sowie die von der Lambda-Laufzeit ausgelösten Ausnahmen.
AWS Lambda	Drosselungen	Dieser Alarm erkennt eine hohe Anzahl gedrosselter Aufrufanforderungen. Drosselung findet statt, wenn keine Gleichzeitigkeit für das Hochskalieren verfügbar ist.

Service	Metrik	Details
Lambda-Einblicke	memory_utilization	Dieser Alarm wird verwendet, um zu erkennen, ob sich die Speicherauslastung einer Lambda-Funktion dem konfigurierten Grenzwert nähert.
Amazon S3	4xxErrors	Dieser Alarm hilft uns dabei, die Gesamtzahl der 4xx-Fehlerstatuscodes zu melden, die als Antwort auf Kundenanfragen ausgegeben wurden. 403-Fehlercodes können auf eine falsche IAM-Richtlinie hinweisen, und 404-Fehlercodes können beispielsweise auf ein fehlerhaftes Verhalten der Client-Anwendung hinweisen.
Amazon S3	5xxErrors	Dieser Alarm hilft Ihnen, eine große Anzahl von serverseitigen Fehlern zu erkennen. Diese Fehler deuten darauf hin, dass ein Client eine Anfrage gestellt hat, die der Server nicht abschließen konnte. So können Sie das Problem, mit dem Ihre Anwendung aufgrund von S3 konfrontiert ist, besser zuordnen.

AWS AppConfig Dokumentverlauf des Benutzerhandbuchs

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation seit der letzten Version von beschrieben AWS AppConfig.

Aktuelle API-Version: 2019-10-09

Änderung	Beschreibung	Datum
IPv6 Unterstützung	Alle unterstützen IPv4 und IPv6 rufen AWS AppConfig APIs jetzt voll an. Weitere Informationen finden Sie unter Grundlegendes zum IPv6 Support.	23. April 2025
Neues Thema: Speichern einer früheren Feature-Flag-Version in einer neuen Version	Wenn Sie ein Feature-Flag aktualisieren, AWS AppConfig werden Ihre Änderungen automatisch in einer neuen Version gespeichert. Wenn Sie eine frühere Feature-Flag-Version verwenden möchten, müssen Sie sie in eine Entwurfsversion kopieren und dann speichern. Sie können Änderungen an einer früheren Flag-Version nicht bearbeiten und speichern , ohne sie in einer neuen Version zu speichern. Weitere Informationen finden Sie unter Speichern einer früheren Feature-Flag-Version in einer neuen Version.	15. April 2025

Neues Thema: Beispiele für Feature-Flags für den lokalen Entwicklungsmodus für AWS AppConfig Agenten	AWS AppConfig Der Agent unterstützt einen lokalen Entwicklungsmodus. Wenn Sie den lokalen Entwicklungsmodus aktivieren, liest der Agent Konfigurationsdaten aus einem angegebenen Verzeichnis auf der Festplatte. Er ruft keine Konfigurationsdaten von ab AWS AppConfig. Damit Sie besser verstehen, wie Sie den lokalen Entwicklungsmodus verwenden, enthält dieses Handbuch jetzt ein Thema mit Beispielen für Feature-Flags. Weitere Informationen finden Sie unter Beispiele für Feature-Flags für den lokalen Entwicklungsmodus für AWS AppConfig Agenten.	18. Februar 2025

Neues Thema: Ein Konfigurationsprofil für nicht-native Datenquellen erstellen	Das Thema beschreibt den allgemeinen Prozess für die Verwendung einer AWS AppConfig Erweiterung zum Abrufen von Konfigurationsdaten aus Quellen, die nicht nativ unterstützt werden, einschließlich anderer AWS Dienste wie Amazon RDS und Amazon DynamoDB sowie Quellen von Drittanbietern wie GitHub, GitLab oder einem lokalen Repository. Weitere Informationen finden Sie unter Erstellen eines Konfigurationsprofils für nicht-native Datenquellen	19. Dezember 2024
Aktualisiertes Thema: Regex in der Typreferenz für Feature-Flags wurde korrigiert	Das JSON-Schema in der Feature-Flag-Typreferenz zeigte zuvor an verschiedenen Stellen das folgende Regex-Muster:. `"^[a-z][a-zA-Z\\d-_]{0,63}$"` Das richtige Regex-Muster ist. `"^[a-z][a-zA-Z\\d_-]{0,63}$"` Der Bindestrich wird nach dem Unterstrich aufgeführt. Weitere Informationen finden Sie unter Grundlegendes zur Typenreferenz für AWS. AppConfig. FeatureFlags	18. Dezember 2024

Aktualisierte Themen: Beispiele für Umgebungsvariablen hinzugefügt	Die Tabellen, in denen Umgebungsvariablen in den folgenden Themen beschrieben werden, wurden aktualisiert und enthalten nun auch Beispiele: • (Optional) Verwendung von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent für Amazon ECS und Amazon EKS • (Optional) Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon EC2 • Konfiguration der AWS AppConfig Agent-Lambda-Erweiterung	12. Dezember 2024
Neuer Abschnitt: Den Split-Operator verstehen	In einem neuen Abschnitt wird anhand von Beispielen erklärt, wie der `split` Operator für eine Feature-Flag-Regel mit mehreren Varianten funktioniert. Weitere Informationen finden Sie unter Grundlegendes zu Feature-Flag-Regeln mit mehreren Varianten.	22. November 2024

Neuer Aktionspunkt für Erweiterungen: AT_DEPLOYMENT_TICK	AWS AppConfig hat einen neuen Aktionspunkt für Benutzer gestartet, die benutzerdefinierte Erweiterungen erstellen. Der `AT_DEPLOYMENT_TICK` Aktionspunkt unterstützt die Integration von Überwachungslösungen durch Drittanbieter. `AT_DEPLOYMENT_TICK` wird während der Konfiguration, Bereitstellung, Verarbeitung und Orchestrierung aufgerufen. Wenn Sie eine Überwachungslösung eines Drittanbieters verwenden (z. B. Datadog), können Sie eine AWS AppConfig Erweiterung erstellen, die am `AT_DEPLOYMENT_TICK` Aktionspunkt nach Alarmen sucht und als Sicherheitsmaßnahme die Bereitstellung rückgängig macht, falls sie einen Alarm ausgelöst hat. Weitere Informationen zu AWS AppConfig Erweiterungen finden Sie unter Workflows mithilfe von Erweiterungen erweitern AWS AppConfig. Weitere Informationen zu benutzerdefinierten Erweiterungen finden Sie unter Exemplarische Vorgehensweise: Erstellen benutzerdefinierter AWS	22. November 2024

	AppConfig Erweiterungen. Ein Codebeispiel für eine AWS AppConfig Erweiterung, die den `AT_DEPLOYMENT_TICK` Aktionspunkt zur Integration in Datadog verwendet, finden Sie unter aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub	
AWS AppConfig Neues Thema: Überlegungen zur mobilen Nutzung	Ein neues Thema in diesem Handbuch beschreibt wichtige Überlegungen zur Verwendung von AWS AppConfig Feature-Flags auf Mobilgeräten. Weitere Informationen finden Sie unter Überlegungen zur Nutzung von AWS AppConfig Mobilgeräten.	21. November 2024
Neue Funktion: AWS AppConfig Löschschutz	AWS AppConfig bietet jetzt eine Kontoeinstellung, um zu verhindern, dass Benutzer versehentlich aktiv genutzte Umgebungen und Konfigurationsprofile löschen. Weitere Informationen finden Sie unter Löschschutz konfigurieren. AWS AppConfig	28. August 2024

Neue Version der AWS AppConfig Agent Lambda-Erweiterung	Der Agent wurde mit kleineren Verbesserungen und Fehlerkorrekturen aktualisiert. Die neuen Amazon-Ressourcennamen (ARNs) für die Erweiterung finden Sie unter Verfügbare Versionen der AWS AppConfig Agent Lambda-Erweiterung.	9. August 2024
Neue Codebeispiele zum Abrufen von Flaggenvarianten	Weitere Informationen finden Sie unter Verwenden AWS AppConfig des Agenten zum Abrufen eines Feature-Flags mit Varianten.	9. August 2024
Neue Version der AWS AppConfig Agent Lambda-Erweiterung	Der Agent wurde aktualisiert und unterstützt nun Feature-Flag-Ziele, -Varianten und -Splits. Die neuen Amazon-Ressourcennamen (ARNs) für die Erweiterung finden Sie unter Verfügbare Versionen der AWS AppConfig Agent Lambda-Erweiterung.	23. Juli 2024

Neue Funktion: Feature-Flags mit mehreren Varianten	Feature-Flags mit mehreren Varianten ermöglichen es Ihnen, eine Reihe möglicher Flag-Werte zu definieren, die für eine Anfrage zurückgeg eben werden sollen. Sie können auch verschiedene Status (aktiviert oder deaktivie rt) für Flags mit mehreren Varianten konfigurieren. Wenn Sie ein mit Varianten konfiguri ertes Kennzeichen anfordern , stellt Ihre Anwendung einen Kontext bereit, der anhand einer Reihe von benutzerd efinierten Regeln AWS AppConfig ausgewertet wird. Abhängig vom in der Anfrage angegebenen Kontext und den für die Variante definiert en Regeln werden unterschi edliche Flagwerte an die Anwendung AWS AppConfig zurückgegeben. Weitere Informationen finden Sie unter Feature-Flags mit mehreren Varianten erstellen.	23. Juli 2024

Neue Version der AWS AppConfig Agent Lambda-Erweiterung	Der Agent wurde mit kleineren Verbesserungen und Fehlerkorrekturen aktualisiert. Die neuen Amazon-Ressourcennamen (ARNs) für die Erweiterung finden Sie unter Verfügbare Versionen der AWS AppConfig Agent Lambda-Erweiterung.	28. Februar 2024
AWS AppConfig Beispiele für benutzerdefinierte Erweiterungen	Das Thema Exemplarische Vorgehensweise: Benutzerdefinierte AWS AppConfig Erweiterungen erstellen enthält jetzt Links zu den folgenden Beispielerweiterungen für: GitHub • Beispielerweiterung, die Bereitstellungen mit einem `blocked day` Moratoriumskalender mithilfe von Systems Manager Change Calendar verhindert • Beispielerweiterung, die verhindert, dass Geheimnisse mithilfe von Git-Secrets in Konfigurationsdaten gelangen • Beispielerweiterung, die verhindert, dass personenbezogene Daten (PII) mit Amazon Comprehend in Konfigurationsdaten gelangen	28. Februar 2024

Neues Thema: Protokollieren von AWS AppConfig API-Aufrufen mit AWS CloudTrail	AWS AppConfig ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden AWS AppConfig. CloudTrail erfasst alle API-Aufrufe AWS AppConfig als Ereignisse. Dieses neue Thema enthält AWS AppConfig spezifische Inhalte, anstatt auf die entsprechenden Inhalte im AWS Systems Manager Benutzerhandbuch zu verweisen. Weitere Informationen finden Sie unter Protokollieren von AWS AppConfig API-Aufrufen mithilfe von AWS CloudTrail.	18. Januar 2024

AWS AppConfig unterstützt jetzt AWS PrivateLink	Sie können AWS PrivateLink damit eine private Verbindung zwischen Ihrer VPC und AWS AppConfig herstellen. Sie können darauf zugreifen, AWS AppConfig als ob es in Ihrer VPC wäre, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder AWS Direct Connect eine Verbindung zu verwenden. Instances in Ihrer VPC benötigen für den Zugriff AWS AppConfig keine öffentlichen IP-Adressen. Weitere Informationen finden Sie unter Zugriff AWS AppConfig über einen Schnittstellenendpunkt (AWS PrivateLink).	6. Dezember 2023

Zusätzliche Funktionen zum Abrufen von AWS AppConfig Agenten und ein neuer lokaler Entwicklungsmodus	AWS AppConfig Agent bietet die folgenden zusätzlichen Funktionen, mit denen Sie Konfigurationen für Ihre Anwendungen abrufen können. Zusätzliche Abruffunktionen • Abruf mehrerer Konten: Verwenden Sie den AWS AppConfig Agenten von einem Primärkonto oder vom Abruf aus AWS-Konto , um Konfigurationsdaten von Konten mehrerer Anbieter abzurufen. • Konfigurationskopie auf Festplatte schreiben: Verwenden Sie den AWS AppConfig Agenten, um Konfigurationsdaten auf die Festplatte zu schreiben. Diese Funktion ermöglich t Kunden mit Anwendung en, die Konfigurationsdaten von der Festplatte lesen, die Integration in Anwendungen AWS AppConfig. **Note** Das Schreiben der Konfiguration auf die Festplatte ist nicht als Funktion	1. Dezember 2023

zur Sicherung der Konfiguration konzipiert. AWS AppConfig Der Agent liest nicht aus den auf die Festplatte kopierten Konfigurationsdateien. Informationen zum Sichern von Konfigurationen auf der Festplatte finden Sie in den `BACKUP_DIRECTORY` `PRELOAD_BACKUP` Umgebungsvariablen Using AWS AppConfig Agent with Amazon EC2 oder Using AWS AppConfig Agent with Amazon ECS and Amazon EKS.

Lokaler Entwicklungsmodus

AWS AppConfig Der Agent unterstützt einen lokalen Entwicklungsmodus. Wenn Sie den lokalen Entwicklungsmodus aktivieren, liest der Agent Konfigurationsdaten aus einem angegebenen Verzeichnis auf der Festplatte. Er ruft keine Konfigurationsdaten von ab AWS AppConfig. Sie können Konfigurationsbereitstellungen simulieren, indem Sie

	Dateien im angegebenen Verzeichnis aktualisieren. Wir empfehlen den lokalen Entwicklungsmodus für die folgenden Anwendungsfälle: • Testen Sie verschied ene Konfigurationsvers ionen, bevor Sie sie mithilfe von bereitstellen AWS AppConfig. • Testen Sie verschiedene Konfigurationsoptionen für eine neue Funktion, bevor Sie Änderungen in Ihr Code-Repository übernehmen. • Testen Sie verschiedene Konfigurationsszenarien, um sicherzustellen, dass sie erwartungsgemäß funktioni eren.	
Neues Thema mit Codebeisp ielen	Diesem Handbuch wurde ein neues Thema mit Codebeisp ielen hinzugefügt. Das Thema umfasst Beispiele in Java, Python und JavaScript für die programmgesteuerte Ausführung von sechs gängigen AWS AppConfig Aktionen.	17. November 2023

Das Inhaltsverzeichnis wurde überarbeitet, um den Arbeitsablauf besser widerzuspiegeln AWS AppConfig	Der Inhalt dieses Benutzerhandbuchs ist jetzt unter den Überschriften Workflows erstellen, bereitstellen, abrufen und erweitern gruppiert. Diese Organisation spiegelt den Arbeitsablauf bei der Verwendung besser wider AWS AppConfig und soll dazu beitragen, dass Inhalte leichter auffindbar sind.	7. November 2023
Payload-Referenz hinzugefügt	Das Thema „Eine Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung erstellen“ enthält jetzt eine Payload-Referenz für Anfragen und Antworten.	7. November 2023
Neue AWS vordefinierte Bereitstellungsstrategie	AWS AppConfig bietet jetzt die `AppConfig.Linear20PercentEvery6Minutes` vordefinierte Bereitstellungsstrategie an und empfiehlt diese. Weitere Informationen finden Sie unter Vordefinierte Bereitstellungsstrategien.	11. August 2023

AWS AppConfig Integration mit Amazon EC2	AWS AppConfig Mithilfe von AWS AppConfig Agent können Sie Anwendungen integrieren, die auf Ihren Amazon Elastic Compute Cloud (Amazon EC2) Linux-Instances ausgeführt werden. Der Agent unterstützt x86_64 und ARM64 Architekturen für Amazon. EC2 Weitere Informationen finden Sie unter AWS AppConfig Integration mit Amazon EC2.	20. Juli 2023
AWS CloudFormation Unterstützung für neue AWS AppConfig Ressourcen und ein Beispiel für eine Feature-Flagge	AWS CloudFormation unterstützt jetzt die AWS::AppConfig::ExtensionAssociationRessourcen AWS::AppConfig::Extensionund, um Ihnen den Einstieg in AWS AppConfig Erweiterungen zu erleichtern. Die Ressourcen AWS::AppConfig::ConfigurationProfile und AWS::AppConfig::HostedConfigurationVersion enthalten jetzt ein Beispiel für die Erstellung eines Feature-Flag-Konfigurationsprofils im AWS AppConfig gehosteten Konfigurationsspeicher.	12. April 2023

AWS AppConfig Integration mit AWS Secrets Manager	AWS AppConfig integriert mit AWS Secrets Manager. Secrets Manager hilft Ihnen dabei, Anmeldeinformationen für Ihre Datenbanken und andere Dienste sicher zu verschlüsseln, zu speichern und abzurufen. Anstatt Anmeldeinformationen in Ihren Apps fest zu codieren, können Sie Secrets Manager aufrufen, um Ihre Anmeldeinformationen bei Bedarf abzurufen. Secrets Manager hilft Ihnen dabei, den Zugriff auf Ihre IT-Ressourcen und Daten zu schützen, indem Sie den Zugriff auf Ihre Secrets rotieren und verwalten können. Wenn Sie ein Freiform-Konfigurationsprofil erstellen, können Sie Secrets Manager als Quelle Ihrer Konfigurationsdaten wählen. Sie müssen Secrets Manager nutzen und ein Geheimnis erstellen, bevor Sie das Konfigurationsprofil erstellen. Weitere Informationen zu Secrets Manager finden Sie unter Was ist AWS Secrets Manager? im AWS Secrets Manager Benutzerhandbuch. Informationen zum Erstellen eines Konfigurationsprofils	2. Februar 2023

finden Sie unter Erstellen eines Freiform-Konfigurationsprofils.

AWS AppConfig Integration mit Amazon ECS und Amazon EKS	Mithilfe des Agenten können Sie Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) integrieren AWS AppConfig . AWS AppConfig Der Agent fungiert als Sidecar-Container, der neben Ihren Amazon ECS- und Amazon EKS-Containeranwendungen ausgeführt wird. Der Agent verbessert die Verarbeitung und Verwaltung containerisierter Anwendungen auf folgende Weise: • Der Agent ruft in Ihrem Namen AWS AppConfig an, indem er eine AWS Identity and Access Management (IAM-) Rolle verwendet und einen lokalen Cache mit Konfigurationsdaten verwaltet. Durch das Abrufen von Konfigurationsdaten aus dem lokalen Cache benötigt Ihre Anwendung weniger Codeaktualisierungen zur Verwaltung von Konfigurationsdaten, ruft Konfigurationsdaten in Millisekunden ab und ist nicht von Netzwerkproblemen betroffen, die Aufrufe	2. Dezember 2022

solcher Daten unterbrechen können.

- Der Agent bietet eine native Oberfläche zum Abrufen und Auflösen von Feature-F lags. AWS AppConfig
- Der sofort einsatzbereite Agent bietet bewährte Methoden für Caching-S trategien, Abfrageintervalle und die Verfügbarkeit lokaler Konfigurationsdaten und verfolgt gleichzeitig die für nachfolgende Servicean fragen benötigten Konfigura tionstoken.
- Während der Ausführun g im Hintergrund fragt der Agent die AWS AppConfig Datenebene regelmäßig nach Aktualisierungen der Konfigurationsdaten ab. Ihre containerisierte Anwendung kann die Daten abrufen, indem sie über Port 2772 (ein anpassbarer Standard-Portwert) eine Verbindun g zu localhost herstellt und HTTP GET aufruft, um die Daten abzurufen.
- Der AWS AppConfig Agent aktualisiert die Konfigurationsdaten in Ihren Containern, ohne diese Container neu starten oder recyceln zu müssen.

Weitere Informationen finden Sie unter AWS AppConfig Integration mit Amazon ECS und Amazon EKS.

Neue Erweiterung: AWS AppConfig Erweiterung für CloudWatch Evidently	Sie können Amazon CloudWatch Evidently verwenden, um neue Funktionen sicher zu validiere n, indem Sie sie während der Einführung der Funktion einem bestimmten Prozentsatz Ihrer Nutzer zur Verfügung stellen. Sie können die Leistung des neuen Feature überwachen, um zu entscheiden, wann Sie den Traffic für Ihre Benutzer erhöhen möchten. Dadurch senken Sie Risiken und erkennen unbeabsichtigtes Verhalten noch bevor Sie das Feature vollständig einführen . Sie können auch A/B-Exper imente durchführen, um Features auf der Grundlage von Erkenntnissen und Daten zu gestalten. Die AWS AppConfig Erweiterung für CloudWatc h Evidently ermöglicht es Ihrer Anwendung, Benutzers itzungen lokal Varianten zuzuweisen, anstatt den EvaluateFeatureVorgang aufzurufen. Eine lokale Sitzung mindert die Latenz- und Verfügbarkeitsrisiken, die mit einem API-Aufru f einhergehen. Informati onen zur Konfiguration und	13. September 2022

	Verwendung der Erweiterung finden Sie unter Durchführen von Produkteinführungen und A/B-Experimenten mit CloudWatch Evidently im CloudWatch Amazon-Benutzerhandbuch.	
Die API-Aktion ist veraltet `GetConfiguration`	Am 18. November 2021 AWS AppConfig wurde ein neuer Datenebenendienst veröffentlicht. Dieser Dienst ersetzt den vorherigen Prozess des Abrufs von Konfigurationsdaten mithilfe der `GetConfiguration` API-Aktion. Der Datenebenendienst verwendet zwei neue API-Aktionen, StartConfigurationSessionund GetLatestConfiguration. Der Datenebenendienst verwendet auch neue Endpunkte. Weitere Informationen finden Sie unter Über den AWS AppConfig Datenebenendienst.	13. September 2022

Neue Version der AWS AppConfig Agent Lambda-Erweiterung	Version 2.0.122 der AWS AppConfig Agent Lambda-Erweiterung ist jetzt verfügbar. Die neue Erweiterung verwendet unterschiedliche Amazon-Ressourcennamen (ARNs). Weitere Informationen finden Sie in den Versionshinweisen zur AWS AppConfig Agent Lambda-Erweiterung.	23. August 2022
Einführung von Erweiterungen AWS AppConfig	Eine Erweiterung erweitert Ihre Fähigkeit, Logik oder Verhalten an verschiedenen Stellen während des AWS AppConfig Workflows zur Erstellung oder Bereitstellung einer Konfiguration einzufügen. Sie können Erweiterungen AWS verwenden oder eigene Erweiterungen erstellen. Weitere Informationen finden Sie unter Mit Erweiterungen arbeiten. AWS AppConfig	12. Juli 2022
Neue Version der AWS AppConfig Agent Lambda-Erweiterung	Version 2.0.58 der AWS AppConfig Agent Lambda-Erweiterung ist jetzt verfügbar. Die neue Erweiterung verwendet unterschiedliche Amazon-Ressourcennamen (ARNs). Weitere Informationen finden Sie unter Verfügbare Versionen der AWS AppConfig Lambda-Erweiterung.	3. Mai 2022

AWS AppConfig Integration mit Atlassian Jira	Die Integration mit Atlassian Jira ermöglicht es, Probleme in der Atlassian-Konsole AWS AppConfig zu erstellen und zu aktualisieren, wenn du Änderungen an einem Feature-Flag in deinem für das angegebene Feature vornimmst. AWS-Konto AWS-Region Jedes Jira-Problem umfasst den Flag-Namen, die Anwendungs-ID, die Konfigurationsprofil-ID und die Flag-Werte. Nachdem Sie Ihre Flag-Änderungen aktualisiert, gespeichert und bereitgestellt haben, aktualisiert Jira die vorhandenen Probleme mit den Details der Änderung. Weitere Informationen findest du unter AWS AppConfig Integration mit Atlassian Jira.	7. April 2022

Allgemeine Verfügbarkeit von Feature-Flags und Lambda-Erweiterungsunterstützung für ARM64 (Graviton2) -Prozessoren	Mit AWS AppConfig Feature-Flags können Sie eine neue Funktion entwickeln und sie in der Produktion einsetzen, während Sie die Funktion vor Benutzern verbergen. Sie beginnen damit, das Flag AWS AppConfig als Konfigurationsdaten hinzuzufügen. Sobald die Funktion zur Veröffentlichung bereit ist, können Sie die Flag-Konfigurationsdaten aktualisieren, ohne Code bereitzustellen. Diese Funktion verbessert die Sicherheit Ihrer Dev-Ops-Umgebung, da Sie keinen neuen Code bereitstellen müssen, um die Funktion zu veröffentlichen. Weitere Informationen finden Sie unter Erstellen eines Feature-Flag-Konfigurationsprofils. Die allgemeine Verfügbarkeit von Feature-Flags in AWS AppConfig umfasst die folgenden Verbesserungen: • Die Konsole enthält eine Option, mit der ein Flag als kurzfristiges Flag festgelegt werden kann. Sie können die Liste der Flaggen nach kurzfristigen Kennzeichnungen filtern und sortieren.	15. März 2022

	• Kunden, die Feature-Flags in verwenden AWS Lambda, können mit der neuen Lambda-Erweiterung einzelne Feature-Flags über einen HTTP-Endpunkt aufrufen. Weitere Informationen finden Sie unter Abrufen eines oder mehrerer Flags aus einer Feature-Flag-Konfiguration. Dieses Update bietet auch Unterstützung für AWS Lambda Erweiterungen, die für ARM64 (Graviton2) -Prozessoren entwickelt wurden. Weitere Informationen finden Sie unter Verfügbare Versionen der AWS AppConfig Lambda-Erweiterung.	
Die GetConfiguration API-Aktion ist veraltet	Die `GetConfiguration` API-Aktion ist veraltet. Aufrufe zum Empfangen von Konfigurationsdaten sollten stattdessen `StartConfigurationSession` und `GetLatestConfiguration` APIs verwenden. Weitere Informationen zu diesen APIs und ihrer Verwendung finden Sie unter Konfiguration abrufen.	28. Januar 2022

Neue Region ARN für AWS AppConfig Lambda-Erweiterung	AWS AppConfig Die Lambda-Erweiterung ist in der neuen Region Asien-Pazifik (Osaka) verfügbar. Der Amazon-Ressourcenname (ARN) ist erforderlich, um ein Lambda in der Region zu erstellen. Weitere Informationen zum ARN für die Region Asien-Pazifik (Osaka) finden Sie unter Hinzufügen der AWS AppConfig Lambda-Erweiterung.	4. März 2021
AWS AppConfig Lambda-Erweiterung	Wenn Sie AWS AppConfig Konfigurationen für eine Lambda-Funktion verwalten, empfehlen wir Ihnen, die AWS AppConfig Lambda-Erweiterung hinzuzufügen. Diese Erweiterung enthält bewährte Methoden, die die Verwendung vereinfachen AWS AppConfig und gleichzeitig die Kosten senken. Niedrigere Kosten resultieren aus weniger API-Aufrufen an den AWS AppConfig Service und, getrennt davon, geringere Kosten aus kürzeren Verarbeitungszeiten für Lambda-Funktionen. Weitere Informationen finden Sie unter AWS AppConfig Integration mit Lambda-Erweiterungen.	8. Oktober 2020

Neuer Abschnitt	Es wurde ein neuer Abschnitt hinzugefügt, der Anweisungen zur Einrichtung AWS AppConfig enthält. Weitere Informationen finden Sie unter Einrichtung AWS AppConfig.	30. September 2020
Befehlszeilenprozeduren hinzugefügt	Die Verfahren in diesem Benutzerhandbuch beinhalten jetzt Befehlszeilenschritte für AWS Command Line Interface (AWS CLI) und Tools für Windows. PowerShell Weitere Informationen finden Sie unter Arbeiten mit AWS AppConfig.	30. September 2020
Veröffentlichung des AWS AppConfig Benutzerhandbuchs	Verwenden Sie AWS AppConfig ein Tool in AWS Systems Manager, um Anwendungskonfigurationen zu erstellen, zu verwalten und schnell bereitzustellen. AWS AppConfig unterstützt kontrollierte Bereitstellungen für Anwendungen jeder Größe und umfasst integrierte Validierungsprüfungen und Überwachung. Sie können es AWS AppConfig mit Anwendungen verwenden, die auf EC2 Instances AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten gehostet werden.	31. Juli 2020

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.