



User Guide

AWS AppConfig



AWS AppConfig: User Guide

Copyright © 2024 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 Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist AWS AppConfig?	1
Anwendungsfälle für AWS AppConfig	2
Vorteile von AWS AppConfig	3
Funktionsweise von AWS AppConfig	4
Erste Schritte mit AWS AppConfig	6
SDKs	6
Preise für AWS AppConfig	6
AWS AppConfig-Kontingente	7
Einrichten AWS AppConfig	8
Melden Sie sich an für ein AWS-Konto	8
Erstellen Sie einen Benutzer mit Administratorzugriff	8
Erteilen programmgesteuerten Zugriffs	10
(Optional) Konfigurieren Sie die Berechtigungen für das Rollback auf der Grundlage von Alarmen CloudWatch	11
Schritt 1: Erstellen Sie die Berechtigungsrichtlinie für ein Rollback auf der Grundlage von Alarmen CloudWatch	12
Schritt 2: Erstellen Sie die IAM-Rolle für Rollback auf der Grundlage von Alarmen CloudWatch	13
Schritt 3: Hinzufügen einer Vertrauensstellung	14
Erstellen	16
Beispielkonfigurationen	17
Informationen zur IAM-Rolle des Konfigurationsprofils	20
Einen Namespace erstellen	22
Eine AWS AppConfig Anwendung (Konsole) erstellen	22
Eine AWS AppConfig Anwendung erstellen (Befehlszeile)	23
Erstellen von Umgebungen	25
Eine AWS AppConfig Umgebung (Konsole) erstellen	25
Eine AWS AppConfig Umgebung erstellen (Befehlszeile)	26
Erstellen eines Konfigurationsprofils in AWS AppConfig	28
Über Validatoren	29
Erstellen eines Feature-Flag-Konfigurationsprofils	32
Erstellen eines Freiform-Konfigurationsprofils	47
Andere Quellen für Konfigurationsdaten	61
AWS Secrets Manager	61

Bereitstellen	63
Mit Bereitstellungsstrategien arbeiten	64
Vordefinierte Bereitstellungsstrategien	66
Erstellen einer Bereitstellungsstrategie	68
Bereitstellen einer Konfiguration	73
Stellen Sie eine Konfiguration bereit (Konsole)	74
Stellen Sie eine Konfiguration bereit (Befehlszeile)	75
Bereitstellungsintegration mit CodePipeline	78
Wie funktioniert die Integration	79
Wird abgerufen	80
Über den AWS AppConfig Data Plane-Service	81
Vereinfachte Abrufmethoden	82
Abrufen von Konfigurationsdaten mithilfe der AWS AppConfig Agent Lambda-Erweiterung ...	83
Abrufen von Konfigurationsdaten von Amazon EC2 EC2-Instances	141
Abrufen von Konfigurationsdaten von Amazon ECS und Amazon EKS	157
Zusätzliche Abruffunktionen	173
AWS AppConfig Lokale Entwicklung des Agenten	184
Konfigurationen durch direktes Aufrufen von APIs abrufen	186
Ein Konfigurationsbeispiel wird abgerufen	187
Erweitern von Workflows	190
Informationen zu AWS AppConfig Erweiterungen	190
Schritt 1: Festlegen, was Sie mit Erweiterungen machen möchten	191
Schritt 2: Ermitteln, wann die Erweiterung ausgeführt werden soll	192
Schritt 3: Erstellen einer Erweiterungszuordnung	193
Schritt 4: Bereitstellen einer Konfiguration und Überprüfen, ob die Erweiterungsaktionen ausgeführt werden	194
Arbeiten mit AWS von erstellten Erweiterungen	194
Arbeiten mit der Amazon CloudWatch -Evidently-Erweiterung	195
Arbeiten mit der AWS AppConfig deployment events to Amazon EventBridge Erweiterung	195
Arbeiten mit der AWS AppConfig deployment events to Amazon SNS Erweiterung	198
Arbeiten mit der AWS AppConfig deployment events to Amazon SQS Erweiterung	201
Arbeiten mit der JCCP-Erweiterung	204
Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen	209

Erstellen einer Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung ..	211
Konfigurieren von Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung	216
Erstellen einer benutzerdefinierten AWS AppConfig Erweiterung	218
Erstellen einer Erweiterungszuordnung für eine benutzerdefinierte AWS AppConfig Erweiterung	222
Ausführen einer Aktion, die eine benutzerdefinierte AWS AppConfig Erweiterung aufruft	223
Erweiterungsintegration mit J Bol	223
Codebeispiele	224
Erstellen oder Aktualisieren einer im gehosteten Konfigurationsspeicher gespeicherten Freiformkonfiguration	224
Erstellen eines Konfigurationsprofils für ein in Secrets Manager gespeichertes Secret	227
Bereitstellen eines Konfigurationsprofils	228
Verwenden von -AWS AppConfigAgent zum Lesen eines Freiform-Konfigurationsprofils	233
Verwenden von AWS AppConfig Agent zum Lesen eines bestimmten Feature-Flags	235
Verwenden der GetLatestConfig API-Aktion zum Lesen eines Freiform-Konfigurationsprofils ...	236
Bereinigen Ihrer Umgebung	240
Sicherheit	247
Implementieren des Zugriffs mit geringsten Berechtigungen	247
Datenverschlüsselung im Ruhezustand für AWS AppConfig	248
AWS PrivateLink	253
Überlegungen	253
Erstellen eines Schnittstellenendpunkts	253
Erstellen einer Endpunktrichtlinie	254
Secrets-Manager-Schlüsselrotation	255
Einrichten der automatischen Drehung von Secrets-Manager-Secrets, die von bereitgestellt werden AWS AppConfig	255
Überwachen	258
CloudTrail Protokolle	258
AWS AppConfig Informationen in CloudTrail	259
AWS AppConfig -Datenereignisse in CloudTrail	260
AWS AppConfig -Verwaltungsereignisse in CloudTrail	261
Grundlagen zu AWS AppConfig-Protokolldateieinträgen	262
Protokollieren von Metriken für Aufrufe auf AWS AppConfig Datenebene	263
Erstellen eines Alarms für eine CloudWatch Metrik	266
Dokumentverlauf	267

AWS-Glossar	289
.....	CCXC

Was ist AWS AppConfig?

AWS AppConfig Feature-Flags und dynamische Konfigurationen helfen Softwareentwicklern dabei, das Anwendungsverhalten in Produktionsumgebungen ohne vollständige Codebereitstellungen schnell und sicher anzupassen. 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 eine Funktion von 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.

Anwendungsfälle für AWS AppConfig

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 von AWS AppConfig

AWS AppConfig bietet die folgenden Vorteile für Ihr Unternehmen:

- Reduzieren Sie unerwartete Ausfallzeiten für Ihre Kunden

AWS AppConfig reduziert Anwendungsausfallzeiten, da 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 Konfigurationsänderungen den Anwendungsanforderungen entsprechen.
 - Für die semantische 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 AWS AppConfig mit Anwendungen verwenden, die auf EC2-Instances, in AWS Lambda, Containern, mobilen Anwendungen oder auf IoT-Geräten gehostet werden.

Ziele müssen nicht mit dem Systems Manager SSM Agent oder dem IAM-Instanzprofil konfiguriert werden, das für andere Systems Manager Manager-Funktionen erforderlich ist. Dies bedeutet, dass AWS AppConfig mit nicht verwalteten Instances funktioniert.

- Unterbrechungsfreie Aktualisierung von Anwendungen

AWS AppConfig stellt Konfigurationsänderungen zur Laufzeit auf Ihren Zielen bereit – ohne komplizierten Erstellungsprozess oder Offlineschalten der Ziele.

- Steuern der Bereitstellung von Änderungen in Ihrer Anwendung

Bei der Implementierung von Konfigurationsänderungen an Ihren Zielen AWS AppConfig können Sie so das Risiko minimieren, indem Sie eine Bereitstellungsstrategie verwenden. Mithilfe von Bereitstellungsstrategien können Sie Konfigurationsänderungen langsam in Ihrer Flotte

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

Funktionsweise von AWS AppConfig

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 eines Freiform-Konfigurationsprofils in AWS AppConfig](#)

Ein Konfigurationsprofil kann auch optionale Validierungen enthalten, um sicherzustellen, dass Ihre Konfigurationsdaten syntaktisch und semantisch korrekt sind. AWS AppConfig führt eine Überprüfung mit den Validierungen durch, wenn Sie eine Bereitstellung starten. Wenn Fehler erkannt 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 [StartDeployment](#) API-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 [StartConfigurationSession](#) und [GetLatestConfiguration](#) API-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 beschrieben [Vereinfachte Abrufmethoden](#).

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 [GetLatestConfiguration](#) API-Aktionen [StartConfigurationSession](#) und direkt aufrufen.

Erste Schritte mit AWS AppConfig

Die folgenden Ressourcen können bei der direkten Nutzung von AWS AppConfig unterstützen.

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 finden Sie in den folgenden Ressourcen:

- [AWS Command Line Interface](#)
- [AWS-SDK für .NET](#)
- [AWS-SDK für C++](#)
- [AWS-SDK für Go](#)
- [AWS-SDK für Java V2](#)
- [AWS SDK für JavaScript](#)
- [AWS SDK für PHP V3](#)
- [AWS SDK für Python](#)
- [AWS SDK für Ruby V3](#)

Preise 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

Weitere Informationen zu Kontingenten für Services, die AWS AppConfig-Konfigurationen speichern, finden Sie unter [Informationen zu Kontingenten und Einschränkungen des Konfigurationsspeichers](#).

Einrichten AWS AppConfig

Falls Sie dies noch nicht getan haben, registrieren Sie sich für ein 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 ein anzumelden AWS-Konto

1. Öffnen Sie <https://portal.aws.amazon.com/billing/signup>.
2. Folgen Sie den Online-Anweisungen.

Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für ein anmelden AWS-Konto, Root-Benutzer des AWS-Kontos wird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um [Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich](#) ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu <https://aws.amazon.com/> auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für ein 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 Console](#) als 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 Sie einen Benutzer 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 Benutzer in IAM Identity Center Administratorzugriff.

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

Melden Sie sich als Benutzer mit Administratorzugriff an

- 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 Zugriffsportale](#).

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

Anweisungen finden Sie im Benutzerhandbuch unter [Einen Berechtigungssatz erstellen](#).AWS IAM Identity Center

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

Anweisungen finden [Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen](#).

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 von der Art des Benutzers 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 die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	<p>Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten.</p> <ul style="list-style-type: none"> Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an	Folgen Sie den Anweisungen unter Verwenden temporäre Anmeldeinformationen mit

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. <ul style="list-style-type: none"> • Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldedaten im Benutzerhandbuch. AWS CLI AWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

(Optional) Konfigurieren Sie die Berechtigungen für das Rollback auf der Grundlage von Alarmen CloudWatch

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.

Note

Die IAM-Rolle muss zum aktuellen Konto gehören. Standardmäßig AWS AppConfig können nur Alarme überwacht werden, die dem aktuellen Konto gehören. Wenn Sie so konfigurieren AWS AppConfig möchten, dass Bereitstellungen als Reaktion auf Messwerte von einem anderen Konto rückgängig gemacht werden, müssen Sie kontenübergreifende Alarme konfigurieren. Weitere Informationen finden Sie unter [Kontenübergreifende regionsübergreifende CloudWatch Konsole](#) im CloudWatch Amazon-Benutzerhandbuch.

Verwenden Sie die folgenden Verfahren, um eine IAM-Rolle zu erstellen, die ein Rollback AWS AppConfig auf der Grundlage von Alarmen ermöglicht. 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 die AWS AppConfig Berechtigung zum Aufrufen der DescribeAlarms API-Aktion erteilt.

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

1. Öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich Policies (Richtlinien) und dann Create policy (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 [DescribeAlarms](#) API-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 die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
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 SSM. CloudWatchAlarmDiscoveryPolicy
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-Namespaces](#)

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](#)

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 einer anderen Systems Manager Manager-Funktion oder einem anderen AWS Dienst speichern, der sich in [Erstellen eines Freiform-Konfigurationsprofils in AWS AppConfig](#) integrieren lässt AWS AppConfig, wie unter beschrieben.

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

Sofern Sie keine speziellen Anforderungen an das Speichern von Geheimnissen AWS Secrets Manager oder das Verwalten von Daten in Amazon Simple Storage Service (Amazon S3) haben, empfehlen wir, Ihre Konfigurationsdaten im AWS AppConfig gehosteten Konfigurationsspeicher zu hosten, da dieser die meisten Funktionen und Verbesserungen bietet.

Themen

- [Beispielkonfigurationen](#)
- [Informationen 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](#)
- [Andere Quellen für Konfigurationsdaten](#)

Beispielkonfigurationen

Verwenden Sie [AWS AppConfig](#) eine Funktion von AWS Systems Manager, um Anwendungskonfigurationen zu erstellen, zu verwalten und schnell bereitzustellen. Eine Konfiguration ist eine Sammlung von Einstellungen, die das Verhalten Ihrer Anwendung beeinflussen. Hier sind einige Beispiele.

Konfiguration der Feature-Flags

Die folgende Feature-Flag-Konfiguration aktiviert oder deaktiviert 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
```

Betriebskonfiguration

Die folgende Freiformkonfiguration legt Beschränkungen für die Verarbeitung von Anfragen durch eine Anwendung fest.

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
```

Konfiguration der Zugriffskontrollliste

Die folgende Freiformkonfiguration für die Zugriffskontrollliste gibt 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
```

Informationen 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.

Note

Sie können AWS CloudFormation 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 [Informationen zu AWS AppConfig Erweiterungen](#).
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 for 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-APPApplication `
-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 WebMobile, 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 [\(Optional\) Konfigurieren Sie die Berechtigungen für das Rollback auf der Grundlage von Alarmen CloudWatch](#).

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 Anwendungen und dann 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 aus, die berechtigt ist, eine Konfiguration rückgängig zu machen, falls ein Alarm ausgelöst wird.
7. Wählen Sie in der CloudWatch Alarmliste einen oder mehrere Alarme aus, die überwacht werden sollen. AWS AppConfig macht Ihre Konfigurationsbereitstellung rückgängig, wenn einer dieser Alarme in einen Alarmzustand übergeht.
8. (Optional) Wählen Sie im Abschnitt „Erweiterungen zuordnen“ eine Erweiterung aus der Liste aus. Weitere Informationen finden Sie unter [Informationen zu AWS AppConfig Erweiterungen](#).
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 for 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-APPCEEnvironment `
  -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
HttpStatusCodes   : 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
ResponseMetadata  : Runtime Metadata
State             : State of the environment
```

Fahren Sie mit [Erstellen eines Konfigurationsprofils in AWS AppConfig](#) fort.

Erstellen eines Konfigurationsprofils in AWS AppConfig

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 zwei Arten von Konfigurationsprofilen: Feature-Flags und Freiform-Konfigurationen. Eine Feature-Flag-Konfiguration speichert Daten im AWS AppConfig gehosteten Konfigurationsspeicher, und der URI ist einfachhosted. Eine Freiformkonfiguration kann Daten im AWS AppConfig gehosteten Konfigurationsspeicher, in verschiedenen Systems Manager Manager-Funktionen oder in einem AWS Dienst speichern, der integriert ist. AWS AppConfig Weitere Informationen finden Sie unter [Erstellen eines Freiform-Konfigurationsprofils 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. 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.

Themen

- [Über Validatoren](#)
- [Erstellen Sie ein Feature-Flag-Konfigurationsprofil in AWS AppConfig](#)
- [Erstellen eines Freiform-Konfigurationsprofils in AWS AppConfig](#)

Über Validatoren

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 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.

Erstellen Sie ein Feature-Flag-Konfigurationsprofil 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. Weitere Informationen zum AWS AppConfig gehosteten Konfigurationsspeicher finden Sie im [Über den AWS AppConfig gehosteten Konfigurationsspeicher](#) Abschnitt.

Themen

- [Erstellen eines Feature-Flag-Konfigurationsprofils \(Konsole\)](#)
- [Erstellen eines Feature-Flags und eines Feature-Flag-Konfigurationsprofils \(Befehlszeile\)](#)
- [Geben Sie eine Referenz für ein AWS.AppConfig.FeatureFlags](#)

Bevor Sie beginnen

Im folgenden Verfahren können Sie im optionalen Abschnitt Verschlüsselung einen AWS Key Management Service (AWS KMS) -Schlüssel auswählen. 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 in Sicherheit in AWS AppConfig](#).

Das folgende Verfahren bietet Ihnen auch 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 [Informationen zu AWS AppConfig Erweiterungen](#).

Schließlich können Sie im Abschnitt Feature-Flag-Attribute Einschränkungen angeben, wenn Sie die Attributdetails einer neuen Feature-Flagge eingeben. Einschränkungen stellen sicher, dass keine unerwarteten Attributwerte in Ihrer Anwendung bereitgestellt werden. AWS AppConfig unterstützt die folgenden Typen von Flag-Attributen 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	None

Typ	Constraint	Beschreibung
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.

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 die Registerkarte Konfigurationsprofile und Feature-Flags und dann Konfiguration erstellen aus.
4. Wählen Sie im Abschnitt Konfigurationsoptionen die Option Feature-Flag aus.
5. Scrollen Sie nach unten. 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.
 - b. Wählen Sie im Abschnitt „Erweiterungen zuordnen“ eine Erweiterung aus der Liste aus.

- c. 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 Flaggschlüssel.
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. Beachten Sie, dass die Kennzeichnung dadurch AWS AppConfig nicht deaktiviert wird.
13. Wählen Sie im Abschnitt Flaggenattribute die Option Attribut definieren aus. Mithilfe von Attributen können Sie zusätzliche Werte in Ihrem Kennzeichen angeben.
14. Geben Sie für Schlüssel einen Flaggschlüssel an und wählen Sie seinen Typ aus der Liste Typ aus. Sie können optional Attributwerte anhand bestimmter Einschränkungen überprüfen. In der folgenden Abbildung sehen Sie ein Beispiel.

Key	Type	Value	Constraint	
currency	String	USD	CAD,USD,MXN	Remove

Required
 Regular expression
 Enum

Define attribute

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

Note

Notieren Sie die folgenden Informationen:

- 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.
- Flaggenattributsschlüssel für eine bestimmte Flagge müssen eindeutig sein.
- Wählen Sie Erforderlicher Wert aus, um anzugeben, ob ein Attributwert erforderlich ist.

15. 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.
16. Wählen Sie Weiter aus.
17. 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-Flags und eines Feature-Flag-Konfigurationsprofils (Befehlszeile)

Das folgende Verfahren beschreibt, wie Sie das AWS Command Line Interface (unter Linux oder Windows) oder Tools für Windows verwenden, PowerShell um ein AWS AppConfig Feature-Flag-Konfigurationsprofil zu 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.

So erstellen Sie Schritt für Schritt eine Feature-Flags-Konfiguration

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

Linux

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

Windows

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

PowerShell

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

- 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 unter [Geben Sie eine Referenz für ein `AWS.AppConfig.FeatureFlags`](#).
- Verwenden Sie die `CreateHostedConfigurationVersion` API, um Ihre Feature-Flag-Konfigurationsdaten zu speichern AWS AppConfig.

Linux

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

Windows

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

```
--content file://path/to/feature_flag_configuration_data ^  
file_name_for_system_to_store_configuration_data
```

PowerShell

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

Hier ist ein Linux-Beispielbefehl.

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

Der content Parameter verwendet die folgenden base64 codierten Daten.

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

Das System gibt unter anderem folgende Informationen zurück

Linux

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

Windows

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

PowerShell

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

Der `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.

Geben Sie eine Referenz für ein 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", "flags"],
      "additionalProperties": false
    },
    "flagDefinitions": {
      "type": "object",
      "patternProperties": {
        "^[a-z][a-zA-Z\\d-]{0,63}$": {
          "$ref": "#/definitions/flagDefinition"
        }
      },
      "maxProperties": 100,
      "additionalProperties": false
    },
    "flagDefinition": {
      "type": "object",
      "properties": {
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "description": {
          "$ref": "#/definitions/customerDefinedDescription"
        }
      }
    }
  }
}
```

```

    "_createdAt": {
      "type": "string"
    },
    "_updatedAt": {
      "type": "string"
    },
    "_deprecation": {
      "type": "object",
      "properties": {
        "status": {
          "type": "string",
          "enum": ["planned"]
        }
      },
      "additionalProperties": false
    },
    "attributes": {
      "$ref": "#/definitions/attributeDefinitions"
    }
  },
  "additionalProperties": false
},
"attributeDefinitions": {
  "type": "object",
  "patternProperties": {
    "^[a-z][a-zA-Z\\d-_{0,63}$": {
      "$ref": "#/definitions/attributeDefinition"
    }
  },
  "maxProperties": 25,
  "additionalProperties": false
},
"attributeDefinition": {
  "type": "object",
  "properties": {
    "description": {
      "$ref": "#/definitions/customerDefinedDescription"
    },
    "constraints": {
      "oneOf": [
        { "$ref": "#/definitions/numberConstraints" },
        { "$ref": "#/definitions/stringConstraints" },
        { "$ref": "#/definitions/arrayConstraints" },
        { "$ref": "#/definitions/boolConstraints" }
      ]
    }
  }
}

```

```

    ]
  }
},
"additionalProperties": false
},
"flagValues": {
  "type": "object",
  "patternProperties": {
    "^[a-z][a-zA-Z\\d-_{0,63}$": {
      "$ref": "#/definitions/flagValue"
    }
  },
  "maxProperties": 100,
  "additionalProperties": false
},
"flagValue": {
  "type": "object",
  "properties": {
    "enabled": {
      "type": "boolean"
    },
    "_createdAt": {
      "type": "string"
    },
    "_updatedAt": {
      "type": "string"
    }
  },
  "patternProperties": {
    "^[a-z][a-zA-Z\\d-_{0,63}$": {
      "$ref": "#/definitions/attributeValue",
      "maxProperties": 25
    }
  },
  "required": ["enabled"],
  "additionalProperties": false
},
"attributeValue": {
  "oneOf": [
    { "type": "string", "maxLength": 1024 },
    { "type": "number" },
    { "type": "boolean" },
    {
      "type": "array",

```



```
    "oneOf": [
      {
        "items": {
          "type": "string",
          "maxLength": 1024
        }
      },
      {
        "items": {
          "type": "number"
        }
      }
    ]
  },
  "additionalProperties": false
},
"stringConstraints": {
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": ["string"]
    }
  }
},
"required": {
  "type": "boolean"
},
"pattern": {
  "type": "string",
  "maxLength": 1024
},
"enum": {
  "type": "array",
  "maxLength": 100,
  "items": {
    "oneOf": [
      {
        "type": "string",
        "maxLength": 1024
      },
      {
        "type": "integer"
      }
    ]
  }
}
```

```
    }
  }
},
"required": ["type"],
"not": {
  "required": ["pattern", "enum"]
},
"additionalProperties": false
},
"numberConstraints": {
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": ["number"]
    }
  },
  "required": {
    "type": "boolean"
  },
  "minimum": {
    "type": "integer"
  },
  "maximum": {
    "type": "integer"
  }
},
"required": ["type"],
"additionalProperties": false
},
"arrayConstraints": {
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "enum": ["array"]
    }
  },
  "required": {
    "type": "boolean"
  },
  "elements": {
    "$ref": "#/definitions/elementConstraints"
  }
}
},
"required": ["type"],
```

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

⚠ Important

Um Feature-Flag-Konfigurationsdaten abzurufen, muss Ihre Anwendung die `GetLatestConfiguration` API aufrufen. Sie können die Konfigurationsdaten für Feature-Flags nicht durch einen Aufruf `GetConfiguration`, was veraltet ist. Weitere Informationen finden Sie unter [GetLatestKonfiguration](#) in der AWS AppConfig API-Referenz.

Wenn Ihre Anwendung [GetLatestConfiguration](#) aufruft und eine neu bereitgestellte Konfiguration 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 `true` gesetzt ist, 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
  }
}

```

Erstellen eines Freiform-Konfigurationsprofils in AWS AppConfig

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 einfachhosted. Bei Freiform-Konfigurationsprofilen können Sie Ihre Daten im AWS AppConfig gehosteten Konfigurationsspeicher oder in einem der folgenden AWS Dienste und Systems Manager Manager-Funktionen speichern:

Ort	Unterstützte Dateitypen
AWS AppConfig gehosteter Konfigurationsspeicher	YAML, JSON und Text, falls sie mit dem AWS Management Console hinzugefügt wurden. Beliebiger Dateityp, wenn er mithilfe der AWS AppConfig CreateHostedConfigurationVersion API-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

- [Informationen zu Kontingenten und Einschränkungen des Konfigurationsspeichers](#)
- [Über den AWS AppConfig gehosteten Konfigurationsspeicher](#)
- [Über in Amazon S3 gespeicherte Konfigurationen](#)
- [Erstellen einer Freiformkonfiguration und eines Konfigurationsprofils](#)

Informationen 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 Dokumente	AWS CodePipeline
Begrenzung der Konfigurationsgröße	Standard: 2 MB, maximal 4 MB	2 MB Erzwungen durch AWS AppConfig, nicht durch S3	4 KB (kostenlos 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
Ressourcen Speicherlimit	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

Über den AWS AppConfig gehosteten Konfigurationsspeicher

AWS AppConfig umfasst einen internen oder gehosteten Konfigurationsspeicher. Die 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.

Über in Amazon S3 gespeicherte 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: GetObject Ausführung

Berechtigungen zum Auflisten von S3-Buckets

s3: ListAll MyBuckets

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

- s3: GetBucket Standort
- s3: GetBucket Versionierung
- s3: ListBucket
- s3: ListBucket Versionen

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 die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich Policies (Richtlinien) und dann Create policy (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 die *Platzhalterwerte* durch Ihre eigenen Informationen.

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

5. 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 Roles (Rollen) zurück.

Die IAM-Rolle für den Zugriff auf ein S3-Objekt erstellen

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 die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und dann Create role (Rolle erstellen).
3. Wählen Sie im Abschnitt Typ der vertrauenswürdigen Entität auswählen die Option AWS Service aus.
4. Wählen Sie im Abschnitt Choose a use case (Wählen Sie einen Anwendungsfall) unter Common use cases (Häufige Anwendungsfälle) die Option EC2, aus, und wählen Sie dann Next: Permissions (Weiter: Berechtigungen).
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 Roles (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 einer Freiformkonfiguration und eines Konfigurationsprofils

In diesem Abschnitt wird beschrieben, wie Sie eine Freiformkonfiguration und ein Konfigurationsprofil erstellen. Bevor Sie beginnen, notieren Sie sich die folgenden Informationen.

- Für das folgende Verfahren müssen Sie eine IAM-Dienstrolle angeben, damit Sie auf Ihre Konfigurationsdaten in dem von Ihnen ausgewählten Konfigurationsspeicher zugreifen AWS AppConfig können. Diese Rolle ist nicht erforderlich, wenn Sie den AWS AppConfig gehosteten Konfigurationsspeicher verwenden. Wenn Sie S3, Parameter Store oder den Systems Manager Manager-Dokumentenspeicher wählen, müssen Sie entweder eine bestehende IAM-Rolle auswählen oder die Option wählen, dass das System die Rolle automatisch für Sie erstellt. Weitere Informationen über diese Rolle finden Sie unter [Informationen zur IAM-Rolle des Konfigurationsprofils](#).

- Das folgende Verfahren bietet Ihnen auch 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 [Informationen zu AWS AppConfig Erweiterungen](#).
- Wenn Sie ein Konfigurationsprofil für Konfigurationen erstellen möchten, die in S3 gespeichert sind, müssen Sie Berechtigungen konfigurieren. Weitere Informationen zu Berechtigungen und anderen Anforderungen für die Verwendung von S3 als Konfigurationsspeicher finden Sie unter [Über in Amazon S3 gespeicherte Konfigurationen](#).
- Wenn Sie Validatoren verwenden möchten, überprüfen Sie die Details und Anforderungen für deren Verwendung. Weitere Informationen finden Sie unter [Über Validatoren](#).

Themen

- [Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils \(Konsole\)](#)
- [Erstellen eines AWS AppConfig Freiform-Konfigurationsprofils \(Befehlszeile\)](#)

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	<ol style="list-style-type: none"> 1. Wählen Sie ein Dokument aus der Liste aus oder wählen Sie Neues Dokument erstellen. 2. Wenn Sie Neues Dokument erstellen wählen, geben Sie unter Dokumentname einen Namen ein. Erweitern Sie optional

Option ausgewählt	Details
	<p>den Versionsnamen und geben Sie einen Namen für die Dokumentversion ein.</p> <p>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.</p> <p>4. Wählen Sie im Abschnitt Content (Inhalt) entweder YAML oder JSON aus und geben Sie dann die Konfigurationsdaten in das Feld ein.</p>

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, dass Sie eine Validierungsprüfung für neue oder aktualisierte SSM-Parameterkonfigurationen mithilfe von erstellen. 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 erbringen [Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig](#). Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschrieben [Konfigurationen durch direktes Aufrufen von APIs abrufen](#). 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 for 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 The_application_ID \
  --name A_name_for_the_configuration_profile \
  --description A_description_of_the_configuration_profile \
  --location-uri A_URI_to_locate_the_configuration or hosted \
  --retrieval-role-
arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified_location \
  --tags User_defined_key_value_pair_metadata_of_the_configuration_profile \
  --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS_Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Windows

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

PowerShell

```
New-APPConfigurationProfile `
  -Name A_name_for_the_configuration_profile `
  -ApplicationId The_application_ID `
  -Description Description_of_the_configuration_profile `
  -LocationUri A_URI_to_locate_the_configuration or hosted `
  -
  RetrievalRoleArn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified_location `
  -
  Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_configuration_profile `
  -
```

```
-Validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS_Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Important

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 erbringen [Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig](#). Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschrieben [Konfigurationen durch direktes Aufrufen von APIs abrufen](#). 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 [CreateHostedConfigurationVersion](#) API-Operationen neue Versionen der Konfiguration erstellen. AWS CLI Einzelheiten und Beispielbefehle für diesen API-Vorgang finden Sie unter [create-hosted-configuration-version](#) in der Befehlsreferenz.AWS CLI

Fahren Sie mit [Bereitstellung von Feature-Flags und Konfigurationsdaten in AWS AppConfig](#) fort.

Andere Quellen für Konfigurationsdaten

Dieses Thema enthält Informationen zu anderen AWS Diensten, die in integriert AWS AppConfig werden können.

AWS AppConfig Integration 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 Secret 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, das Secrets Manager verwendet, finden Sie unter [Erstellen von Feature-Flags und Freiform-Konfigurationsdaten in AWS AppConfig](#).

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 mithilfe eines vom Kunden verwalteten Schlüssels.

AWS AppConfig führt beim Aufrufen der [StartDeployment](#) API-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.

AWS AppConfig lässt sich in Amazon integrieren CloudWatch , um Bereitstellungen zu überwachen. Wenn eine Bereitstellung einen Alarm auslöst CloudWatch, AWS AppConfig wird die Bereitstellung automatisch zurückgesetzt, um die Auswirkungen auf Ihre Anwendungsbutzer zu minimieren.


Themen

- [Mit Bereitstellungsstrategien arbeiten](#)
- [Bereitstellen einer Konfiguration](#)
- [AWS AppConfig Bereitstellung, Integration mit CodePipeline](#)

Mit Bereitstellungsstrategien arbeiten

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.

Einstellung	Beschreibung														
Deployment type (Bereitstellungstyp)	<p>Der Bereitstellungstyp definiert, wie die Konfiguration bereitgestellt oder eingeführt wird. AWS AppConfig unterstützt lineare und exponentielle Bereitstellungstypen.</p> <ul style="list-style-type: none"> • 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 border="1" data-bbox="862 1121 1507 1732"> <thead> <tr> <th>Verstrichene Zeit</th> <th>Fortschritt bei der Bereitstellung</th> </tr> </thead> <tbody> <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> <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> </tbody> </table> <ul style="list-style-type: none"> • Exponentiell: Für diesen Typ verarbeitet AWS AppConfig die Bereitstellung exponentiell mit der folgenden Formel: $G * (2^N)$. 	Verstrichene Zeit	Fortschritt bei der Bereitstellung	0 Stunde	0%	2 Stunden	20 %	4 Stunden	40%	6 Stunden	60%	8 Stunden	80%	10 Stunden	100 %
Verstrichene Zeit	Fortschritt bei der Bereitstellung														
0 Stunde	0%														
2 Stunden	20 %														
4 Stunden	40%														
6 Stunden	60%														
8 Stunden	80%														
10 Stunden	100 %														

Einstellung	Beschreibung
	<p>In dieser Formel ist G der vom Benutzer angegebene Schrittprozentsatz und N 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:</p> <div data-bbox="862 569 1507 730" style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; margin: 10px 0;"> $2 * (2^0)$ $2 * (2^1)$ $2 * (2^2)$ </div> <p>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.</p>
Schrittprozentsatz (Wachstumsfaktor)	<p>Diese Einstellung gibt den Prozentsatz der Aufrufer an, für den die Bereitstellung bei den einzelnen Schritten ausgeführt werden soll.</p> <div data-bbox="829 1213 1507 1480" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p> Note</p> <p>Im SDK und in der AWS AppConfig - API-Referenz wird <code>step_percentage</code> als <code>growth factor</code> bezeichnet.</p> </div>
Deployment time (Bereitstellungszeit)	<p>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.</p>

Einstellung	Beschreibung
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 (Optional) Konfigurieren Sie die Berechtigungen für das Rollback auf der Grundlage von Alarmen CloudWatch .

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

Themen

- [Vordefinierte Bereitstellungsstrategien](#)
- [Erstellen einer Bereitstellungsstrategie](#)

Vordefinierte 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 bereitges

Bereitstellungsstrategie	Beschreibung
	<p>teilt, 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.</p> <p>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.</p>
AppConfig. Kanarische 10 Prozent 20 Minuten	<p>AWS empfohlen:</p> <p>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 empfangen werden, ist die Bereitstellung abgeschlossen. Wenn während dieser Zeit ein Alarm ausgelöst wird, AWS AppConfig wird die Bereitstellung rückgängig gemacht.</p> <p>Wir empfehlen, diese Strategie für Produktionsbereitstellungen zu verwenden, da sie den AWS bewährten Methoden für Konfigurationsbereitstellungen entspricht.</p>

Bereitstellungsstrategie	Beschreibung
AppConfig.AllAtOnce	<p>Schnell:</p> <p>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.</p>
AppConfig.Linear 50 30 Sekunden PercentEvery	<p>Testen/Vorführung:</p> <p>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.</p> <p>Wir empfehlen, diese Strategie aufgrund der kurzen Dauer und Bake-Zeit nur für Test- oder Demonstrationszwecke zu verwenden.</p>

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 erbringen [Bereitstellen einer Konfiguration](#). Sie müssen jedoch einen Client so konfigurieren, dass er Updates für die Anwendungskonfiguration erhält, wie unter beschrieben [Konfigurationen durch direktes Aufrufen von APIs abrufen](#). 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 for 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_interval \
  --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_interval
^
--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-APPCCDeploymentStrategy `
--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_interval
`
--
GrowthType The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_time
`
--
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

```

Bereitstellen einer Konfiguration

Nachdem Sie die [erforderlichen Artefakte](#) für die Arbeit mit Feature-Flags und Freiform-Konfigurationsdaten erstellt haben, können Sie mithilfe des SDK AWS Management Console, des oder des AWS CLI SDK eine neue Bereitstellung erstellen. Beim Starten einer Bereitstellung in wird der AWS AppConfig [StartDeployment](#)API-Vorgang aufgerufen. Der Aufruf enthält die IDs der AWS AppConfig -Anwendung, der Umgebung und des Konfigurationsprofils sowie (optional) der Konfigurationsdatenversion, die bereitgestellt werden soll. 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 `KmsKeyId` Parameter angeben. Wenn Ihre Konfiguration nicht verschlüsselt oder mit einem verschlüsselt ist Von AWS verwalteter Schlüssel, ist die Angabe eines Werts für den `KmsKeyId` Parameter nicht erforderlich.

Note

Bei dem Wert, den Sie angeben, `KmsKeyId` 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 `startKmsKeyId`, 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 für die Beschreibung der Bereitstellung 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 for 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-APPDeployment \  
  -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 \  
  -Tags User_defined_key_value_pair_metadata_of_the_deployment
```

-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

AWS AppConfig Bereitstellung, Integration 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.

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. Wenn Sie eine der in diesem Abschnitt beschriebenen vereinfachten Abrufmethoden verwenden, verwaltet entweder die AWS AppConfig Agent-Lambda-Erweiterung oder der AWS AppConfig Agent eine Reihe von API-Aufrufen und Sitzungstoken in Ihrem Namen. Sie konfigurieren den AWS AppConfig Agenten als lokalen Host und lassen den Agenten nach Konfigurationsupdates AWS AppConfig fragen. Der Agent ruft die [StartConfigurationSessions](#) - und [GetLatestKonfigurationen-API-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 beschrieben [Vereinfachte Abrufmethoden](#).

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

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.

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.

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.

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.

Dieser Abschnitt enthält folgende Informationen.

Inhalt

- [Über den AWS AppConfig Data Plane-Service](#)
- [Vereinfachte Abrufmethoden](#)
- [Konfigurationen durch direktes Aufrufen von APIs abrufen](#)

Über den AWS AppConfig Data Plane-Service

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: [StartConfigurationSitzung](#) und [GetLatestKonfiguration](#). Der Datenebenendienst verwendet auch [neue Endpunkte](#).

Wenn Sie AWS AppConfig vor dem 28. Januar 2022 mit der Nutzung begonnen haben, ruft der Dienst die `GetConfiguration` API-Aktion möglicherweise direkt auf oder verwendet einen von bereitgestellten Client AWS, z. B. die AWS AppConfig Agent Lambda-Erweiterung, um diese API-Aktion aufzurufen. Wenn Sie die `GetConfiguration` API-Aktion direkt aufrufen, ergreifen Sie die erforderlichen Schritte, um die `StartConfigurationSession` und `GetLatestConfiguration` API-Aktionen zu verwenden. Wenn Sie die AWS AppConfig Agent-Lambda-Erweiterung verwenden, finden Sie weitere Informationen im Abschnitt „Wie sich diese Änderung auf die AWS AppConfig Agent-Lambda-Erweiterung auswirkt“ weiter unten in diesem Thema.

Die neuen API-Aktionen auf Datenebene bieten die folgenden Vorteile gegenüber der `GetConfiguration` API-Aktion, die jetzt nicht mehr unterstützt wird.

1. Sie müssen keinen Parameter verwalten. `ClientID` wird beim Datenebenendienst intern durch `ClientID` das Sitzungstoken verwaltet, das von `StartConfigurationSession` erstellt wurde.
2. Sie müssen die zwischengespeicherte Version Ihrer Konfigurationsdaten nicht mehr angeben. `ClientConfigurationVersion` wird beim Datenebenendienst intern durch `ClientConfigurationVersion` das Sitzungstoken verwaltet, das von `StartConfigurationSession` erstellt wurde.
3. Der neue dedizierte Endpunkt für API-Aufrufe auf Datenebene verbessert die Codestruktur, indem Aufrufe auf Steuerungsebene und Datenebene getrennt werden.

4. Der neue Datenebenenendienst verbessert die future Erweiterbarkeit des Datenebenenbetriebs. Durch die Verwendung einer Konfigurationssitzung, die den Abruf von Konfigurationsdaten verwaltet, kann das AWS AppConfig Team in future leistungsstärkere Verbesserungen vornehmen.

Migration von zu **GetConfigurationGetLatestConfiguration**

Um mit der Nutzung des neuen Datenebenendienstes zu beginnen, müssen Sie Ihren Code aktualisieren, der die `GetConfiguration` API-Aktion aufruft. Starten Sie eine Konfigurationssitzung mithilfe der `StartConfigurationSession` API-Aktion und rufen Sie dann die `GetLatestConfiguration` API-Aktion auf, um die Konfigurationsdaten abzurufen. Um die Leistung zu verbessern, empfehlen wir Ihnen, Ihre Konfigurationsdaten lokal zwischenspeichern. Weitere Informationen finden Sie unter [Konfigurationen durch direktes Aufrufen von APIs abrufen](#).

Wie sich diese Änderung auf die AWS AppConfig Agent Lambda-Erweiterung auswirkt

Diese Änderung hat keine direkten Auswirkungen auf die Funktionsweise der AWS AppConfig Agent Lambda-Erweiterung. Ältere Versionen der AWS AppConfig Agent Lambda-Erweiterung haben die `GetConfiguration` API-Aktion in Ihrem Namen aufgerufen. Neuere Versionen rufen die API-Aktionen der Datenebene auf. Wenn Sie die AWS AppConfig Lambda-Erweiterung verwenden, empfehlen wir Ihnen, Ihre Erweiterung auf den neuesten Amazon Resource Name (ARN) zu aktualisieren und die Berechtigungen für die neuen API-Aufrufe zu aktualisieren. Weitere Informationen finden Sie unter [Abrufen von Konfigurationsdaten mithilfe der AWS AppConfig Agent Lambda-Erweiterung](#).

Vereinfachte Abrufmethoden

AWS AppConfig bietet mehrere vereinfachte Methoden zum Abrufen von Konfigurationsdaten. Wenn Sie AWS AppConfig Feature-Flags oder Freiform-Konfigurationsdaten in einer AWS Lambda Funktion verwenden, können Sie die AWS AppConfig Agent Lambda-Erweiterung verwenden, um Konfigurationen abzurufen. Wenn Sie Anwendungen auf Amazon EC2 EC2-Instances ausführen, können Sie AWS AppConfig Agent verwenden, um Konfigurationen abzurufen. AWS AppConfig Agent unterstützt auch Anwendungen, die auf Container-Images von Amazon Elastic Kubernetes Service (Amazon EKS) oder Amazon Elastic Container Service (Amazon ECS) ausgeführt werden.

Nach Abschluss der Ersteinrichtung sind diese Methoden zum Abrufen von Konfigurationsdaten einfacher als das direkte Aufrufen von APIs. AWS AppConfig Sie implementieren automatisch

bewährte Methoden und können Ihre Nutzungskosten senken, AWS AppConfig da weniger API-Aufrufe zum Abrufen von Konfigurationen erforderlich sind.

Themen

- [Abrufen von Konfigurationsdaten mithilfe der AWS AppConfig Agent Lambda-Erweiterung](#)
- [Abrufen von Konfigurationsdaten von Amazon EC2 EC2-Instances](#)
- [Abrufen von Konfigurationsdaten von Amazon ECS und Amazon EKS](#)
- [Zusätzliche Abruffunktionen](#)
- [AWS AppConfig Lokale Entwicklung des Agenten](#)

Abrufen von Konfigurationsdaten mithilfe der AWS AppConfig Agent Lambda-Erweiterung

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

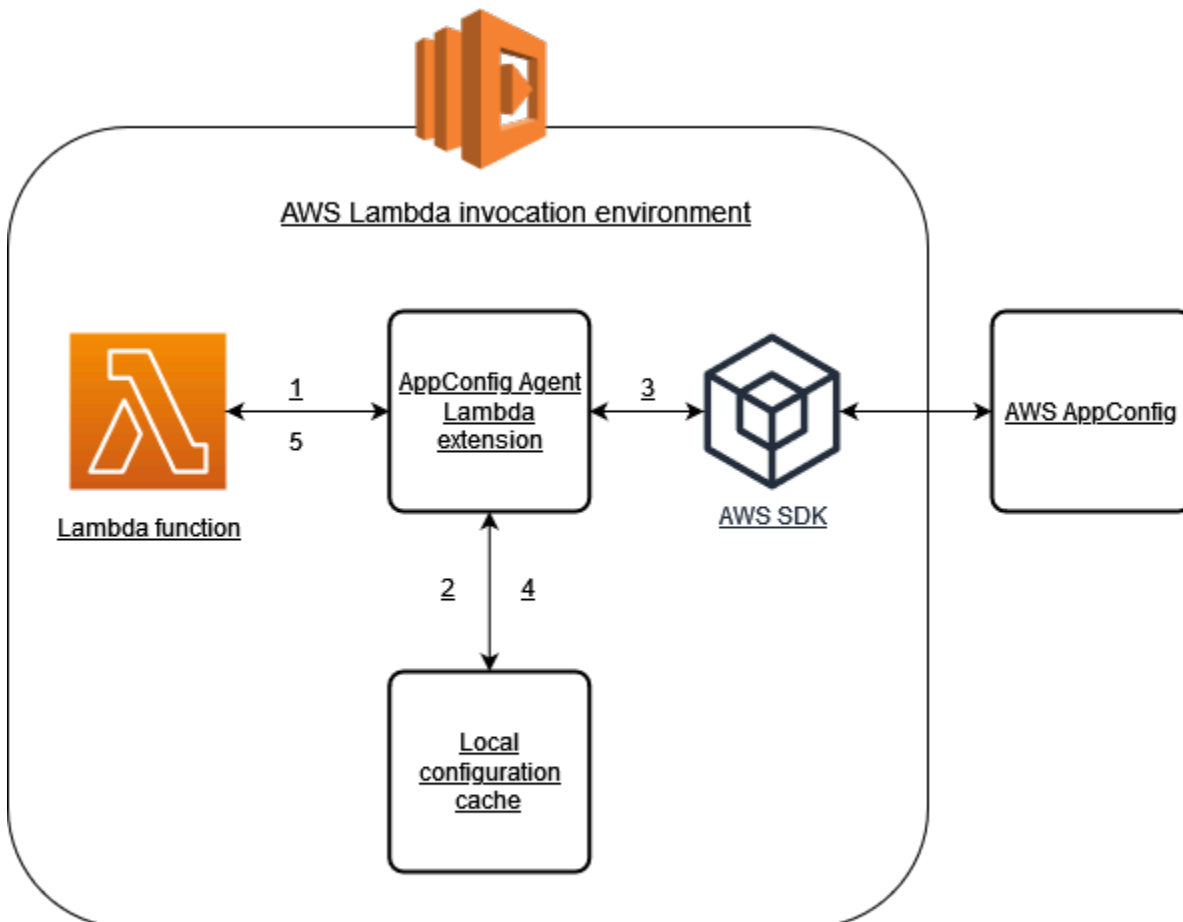
AWS AppConfig [Die Preisgestaltung](#) basiert auf der Häufigkeit, mit der eine Konfiguration aufgerufen und empfangen wird. Ihre Kosten steigen, wenn Ihr Lambda mehrere Kaltstarts durchführt und häufig neue Konfigurationsdaten abrufen.

Dieses Thema enthält Informationen zur AWS AppConfig Agent-Lambda-Erweiterung und das Verfahren zur Konfiguration der Erweiterung für die Verwendung mit Ihrer Lambda-Funktion.

Funktionsweise

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 [StartConfigurationSessions](#) - und [GetLatestConfiguration-API-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.

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

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.

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](#)
- [ARM64-Plattform](#)

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.

Informationen zum Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung als Container-Image finden Sie unter [Verwenden eines Container-Images zum Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung](#).

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 kommasetrennte 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 aus

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 teilweisen oder vollständigen ARNs, die den AWS AppConfig Konfigurationsressourcen 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.

Umgebungsvariable	Details	Standardwert
AWS_APPCONFIG_EXTENSION_HTTP_PORT	Diese Umgebungsvariable gibt den Port an, auf dem der lokale HTTP-Server läuft, der die Erweiterung hostet.	2772
AWS_APPCONFIG_EXTENSION_LOG_LEVEL	Diese Umgebungsvariable gibt an, welche AWS AppConfig erweiterungsspezifischen Protokolle für eine Funktion an Amazon CloudWatch Logs gesendet werden. Gültige Werte ohne Berücksichtigung der Groß- und Kleinschreibung sind: <code>debug</code> , <code>info</code> , <code>warn</code> und <code>error</code> . <code>none</code> Debug enthält detaillierte Informationen, einschließlich Zeitinformationen, über die Erweiterung.	info
AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS	Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, die die Erweiterung zum Abrufen von Konfigurationen verwendet. AWS AppConfig	3
AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS	Diese Umgebungsvariable steuert, wie oft die Erweiterung innerhalb von Sekunden AWS AppConfig nach einer aktualisierten Konfiguration fragt.	45

Umgebungsvariable	Details	Standardwert
<code>AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS</code>	Diese Umgebungsvariable steuert, wie lange (in Millisekunden) die Erweiterung maximal auf eine Antwort wartet, AWS AppConfig wenn sie Daten im Cache aktualisiert. Wenn innerhalb der angegebenen Zeit AWS AppConfig nicht reagiert wird, überspringt die Erweiterung dieses Abfrageintervall und gibt die zuvor aktualisierten zwischengespeicherten Daten zurück.	3000
<code>AWS_APPCONFIG_EXTENSION_PREFETCH_LIST</code>	Diese Umgebungsvariable gibt die Konfigurationsdaten an, die die Erweiterung abrufen, bevor die Funktion initialisiert und der Handler ausgeführt wird. Sie kann die Kaltstartzeit der Funktion erheblich reduzieren.	None

Umgebungsvariable	Details	Standardwert
AWS_APPCONFIG_EXTENSION_PROXY_HEADERS	<p>Diese Umgebungsvariable spezifiziert Header, die der Proxy benötigt, auf den in der <code>AWS_APPCONFIG_EXTENSION_PROXY_URL</code> Umgebungsvariablen verwiesen wird. Der Wert ist eine durch Kommas getrennte Liste von Headern. Jeder Header verwendet die folgende Form:</p> <pre>"header: value"</pre>	None
AWS_APPCONFIG_EXTENSION_PROXY_URL	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen von der AWS AppConfig Erweiterung zu verwendet AWS-Services werden soll. HTTPS und HTTP URLs werden unterstützt.	None
AWS_APPCONFIG_EXTENSION_ROLE_ARN	Diese Umgebungsvariable gibt den ARN der IAM-Rolle an, die einer Rolle entspricht, die von der AWS AppConfig Erweiterung zum Abrufen der Konfiguration übernommen werden sollte.	None
AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID	Diese Umgebungsvariable gibt die externe ID an, die in Verbindung mit der angenommenen Rolle ARN verwendet werden soll.	None

Umgebungsvariable	Details	Standardwert
AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	None
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.	None

Umgebungsvariable	Details	Standardwert
AWS_APPCONFIG_EXTENSION_MANIFEST	<p>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. Sie können einen der folgenden Werte eingeben:</p> <ul style="list-style-type: none"> "app:env:manifest-config" "file:/fully/qualified/path/to/manifest.json" <p>Weitere Informationen zu diesen Funktionen finden Sie unter Zusätzliche Abruffunktionen.</p>	true
AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST	<p>Diese Umgebungsvariable konfiguriert den AWS AppConfig Agenten so, dass er wartet, bis das Manifest verarbeitet ist, bevor der Start abgeschlossen wird.</p>	true

Ein oder mehrere Flags aus einer Feature-Flag-Konfiguration abrufen

Für Feature-Flag-Konfigurationen (Typkonfigurationen `AWS.AppConfig.FeatureFlags`) können Sie mit der Lambda-Erweiterung ein einzelnes Flag oder eine Teilmenge von Flags in einer

Konfiguration abrufen. Das Abrufen von ein oder zwei Flags ist nützlich, wenn Ihr Lambda nur einige Flags aus dem Konfigurationsprofil verwenden muss. Die folgenden Beispiele verwenden Python.

Note

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

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

Um auf ein einzelnes Flag und seine Attribute zuzugreifen

```
import urllib.request

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

Um auf mehrere Flags und ihre Attribute zuzugreifen

```
import urllib.request

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

AWS AppConfig Agent-Lambda-Erweiterungsprotokolle anzeigen

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)
```

Verfügbare 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 die `GetLatestConfiguration` APIs `StartConfigurationSession` und 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 neuen APIs konfigurieren. Weitere Informationen finden Sie unter [Über den AWS AppConfig Data Plane-Service](#).
- 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](#)

- [ARM64-Plattform](#)
- [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
2.0.358	01.12.2023	<p>Unterstützung für die folgenden Abruffunktionen wurde hinzugefügt:</p> <ul style="list-style-type: none"> • 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 ermöglicht Kunden mit Anwendungen, die Konfigurationsdaten von der Festplatte lesen, die Integration AWS AppConfig.
2.0.181	14.08.2023	<p>Unterstützung für Israel (Tel Aviv) AWS-Region il-central-1 hinzugefügt.</p>

Version	Startdatum	Hinweise
2.0.165	21.02.2023	<p>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:</p> <ul style="list-style-type: none">• 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. <code>AWS_APPCONFIG_EXTENSION_PROXY_URL</code> <code>AWS_APPCONFIG_EXTENSION_PROXY_HEADERS</code> .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-Prozessoren (ARM64) in Lambda.

Version	Startdatum	Hinweise
2.0.45	15.03.2022	Unterstützung für das Aufrufen eines einzelnen Feature-Flags wurde hinzugefügt. Bisher riefen Kunden Feature-Flags auf, die in einem Konfigurationsprofil gruppiert waren, und mussten die Antwort clientseitig analysieren. Mit dieser Version können Kunden beim Aufrufen des HTTP-Localhost-Endpunkts einen <code>flag=<flag-name></code> Parameter verwenden, um den Wert eines einzelnen Flags abzurufen. Außerdem wurde anfängliche Unterstützung für Graviton2-Prozessoren (ARM64) 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 die Option AWS- AppConfig -Extension aus der AWS Ebenenliste 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 den erforderlichen URLs für diese 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 2.0.358

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93</code>
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161</code>
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93</code>

Region	ARN
Europa (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106</code>
Europa (Zürich)	<code>arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93</code>
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83</code>
Europa (Spain)	<code>arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:76</code>

Region	ARN
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:76</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79</code>
Asien-Pazifik (Melbourne)	<code>arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20</code>

Region	ARN
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107</code>
Asien-Pazifik (Hyderabad)	<code>arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:83</code>
Israel (Tel Aviv)	<code>arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:22</code>
Naher Osten (VAE)	<code>arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54</code>

ARM64-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 ARM64-Plattform entwickelt wurden.

Version 2.0.358

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45</code>
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63</code>
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13</code>
Europa (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49</code>
Europa (Zürich)	<code>arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5</code>

Region	ARN
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45</code>
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11</code>
Europa (Spain)	<code>arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5</code>
Asien-Pazifik (Hongkong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16</code>

Region	ARN
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16</code>
Asien-Pazifik (Melbourne)	<code>arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49</code>
Asien-Pazifik (Hyderabad)	<code>arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11</code>

Region	ARN
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

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.

Ältere Erweiterungsversionen (x86-64-Plattform)

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

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

Region	ARN
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146</code>
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81</code>
Europa (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93</code>
Europa (Zürich)	<code>arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81</code>
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142</code>

Region	ARN
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73</code>
Europa (Spain)	<code>arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:68</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:68</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91</code>

Region	ARN
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64</code>
Asien-Pazifik (Melbourne)	<code>arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94</code>
Asien-Pazifik (Hyderabad)	<code>arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:73</code>
Israel (Tel Aviv)	<code>arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:7</code>
Naher Osten (VAE)	<code>arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34</code>

Region	ARN
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 14.08.2023

Version 2.0.165

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79</code>
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143</code>

Region	ARN
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79</code>
Europa (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91</code>
Europa (Zürich)	<code>arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79</code>
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71</code>
Europa (Spain)	<code>arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26</code>

Region	ARN
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:66</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:66</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60</code>

Region	ARN
Asien-Pazifik (Melbourne)	<code>arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92</code>
Asien-Pazifik (Hyderabad)	<code>arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71</code>
Naher Osten (VAE)	<code>arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.02.2023

Version 2.0.122

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59</code>
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114</code>
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59</code>
Europe (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59</code>

Region	ARN
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:52</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:52</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59</code>

Region	ARN
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 23.08.2022

Version 2.0.58

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50</code>
USA West (Nordkalifornien)	<code>arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101</code>
Kanada (Zentral)	<code>arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50</code>
Europe (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50</code>

Region	ARN
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46</code>

Region	ARN
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.04.2022

Version 2.0.45

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

Region	ARN
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45</code>

Region	ARN
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 15.03.2022

Version 2.0.30

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

Region	ARN
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44</code>
China (Peking)	<code>arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43</code>
China (Ningxia)	<code>arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43</code>
Asia Pacific (Hong Kong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42</code>
Asia Pacific (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54</code>

Region	ARN
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44</code>
AWS GovCloud (US-Ost)	<code>arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20</code>
AWS GovCloud (US-West)	<code>arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20</code>

Ältere Erweiterungsversionen (ARM64-Plattform)

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

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

Version 2.0.181

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

Region	ARN
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33</code>
Europe (Paris)	<code>arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1</code>
Europe (Stockholm)	<code>arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1</code>
Europa (Milan)	<code>arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Hongkong)	<code>arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37</code>
Asien-Pazifik (Seoul)	<code>arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Osaka)	<code>arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43</code>

Region	ARN
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36</code>
Asien-Pazifik (Jakarta)	<code>arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36</code>
Südamerika (São Paulo)	<code>arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1</code>
Afrika (Kapstadt)	<code>arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1</code>
Naher Osten (Bahrain)	<code>arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 30.03.2023

Version 2.0.165

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43</code>

Region	ARN
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31</code>
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45</code>
Europe (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34</code>

Datum ersetzt durch eine neuere Erweiterung: 21.02.2023

Version 2.0.122

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

Region	ARN
Asien-Pazifik (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13
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

Region	ARN
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2</code>

Datum, das durch eine neuere Erweiterung ersetzt wurde: 21.04.2022

Version 2.0.45

Region	ARN
USA Ost (Nord-Virginia)	<code>arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1</code>
USA Ost (Ohio)	<code>arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1</code>

Region	ARN
USA West (Oregon)	<code>arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2</code>
Europe (Frankfurt)	<code>arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1</code>
Europa (Irland)	<code>arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6</code>
Europe (London)	<code>arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Tokio)	<code>arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Singapur)	<code>arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2</code>
Asien-Pazifik (Sydney)	<code>arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1</code>
Asien-Pazifik (Mumbai)	<code>arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1</code>

Verwenden eines Container-Images zum Hinzufügen der AWS AppConfig Agent Lambda-Erweiterung

Sie können Ihre AWS AppConfig Agent Lambda-Erweiterung als Container-Image verpacken, um sie in Ihre Container-Registry hochzuladen, die auf Amazon Elastic Container Registry (Amazon ECR) gehostet wird.

So fügen Sie die AWS AppConfig Agent Lambda-Erweiterung als Lambda-Container-Image hinzu

1. Geben Sie den AWS-Region und den Amazon-Ressourcennamen (ARN) in das AWS Command Line Interface (AWS CLI) ein, wie unten gezeigt. Ersetzen Sie die Region und den ARN-Wert durch Ihre Region und den passenden ARN, um eine Kopie der Lambda-Schicht abzurufen. AWS AppConfig [bietet ARNs für x86-64 - und ARM64-Plattformen](#).

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

Ein Beispiel:

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

Die Antwort sieht wie folgt aus:

```
{  
  "LayerVersionArn": "arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-  
Extension:128",  
  "Description": "AWS AppConfig extension: Use dynamic configurations deployed via  
AWS AppConfig for your AWS Lambda functions",  
  "CreateDate": "2021-04-01T02:37:55.339+0000",  
  "LayerArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension",  
  
  "Content": {  
    "CodeSize": 5228073,  
    "CodeSha256": "8ot0gbLQbexpUm3rK1MhvcE6Q5TvwCLCKrc40e+vmMY=",  
    "Location" : "S3-Bucket-Location-URL"  
  },  
}
```

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

- In der obigen Antwort `Location` ist der zurückgegebene Wert für die URL des Amazon Simple Storage Service (Amazon S3) -Buckets, der die Lambda-Erweiterung enthält. Fügen Sie die URL in Ihren Webbrowser ein, um die ZIP-Datei mit der Lambda-Erweiterung herunterzuladen.

Note

Die Amazon S3 S3-Bucket-URL ist nur für 10 Minuten verfügbar.

(Optional) Alternativ können Sie auch den folgenden `curl` Befehl verwenden, um die Lambda-Erweiterung herunterzuladen.

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

(Optional) Alternativ können Sie Schritt 1 und Schritt 2 kombinieren, um den ARN abzurufen und die ZIP-Erweiterungsdatei auf einmal herunterzuladen.

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

- Fügen Sie die folgenden Zeilen hinzu `Dockerfile`, um die Erweiterung zu Ihrem Container-Image hinzuzufügen.

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

4. Stellen Sie sicher, dass für die Lambda-Funktionsausführungsrolle der [GetConfigurationBerechtigungssatz appconfig: festgelegt](#) ist.

Beispiel

Dieser Abschnitt enthält ein Beispiel für die Aktivierung der AWS AppConfig Agent Lambda-Erweiterung für eine auf Container-Images basierende Python-Lambda-Funktion.

1. Erstellen Sie eine `Dockerfile`, die der folgenden ähnelt.

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

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

2. Laden Sie die Erweiterungsebene in dasselbe Verzeichnis herunter wie die `Dockerfile`.

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

3. Erstellen Sie eine Python-Datei mit dem Namen `index.py` im selben Verzeichnis wie die `Dockerfile`.

```
import urllib.request

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

def get_configuration(app, env, config):
```

```
url = f'http://localhost:2772/applications/{app}/environments/{env}/
configurations/{config}'
return urllib.request.urlopen(url).read()
```

4. Führen Sie die folgenden Schritte aus, um das docker Image zu erstellen und auf Amazon ECR hochzuladen.

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

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

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

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

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

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

5. Verwenden Sie das Amazon ECR-Image, das Sie oben erstellt haben, um die Lambda-Funktion zu erstellen. Weitere Informationen zu einer Lambda-Funktion als Container finden Sie unter [Erstellen einer Lambda-Funktion, die als Container-Image definiert](#) ist.
6. Stellen Sie sicher, dass für die Lambda-Funktionsausführungsrolle der [GetConfigurationBerechtigungssatz appconfig: festgelegt](#) ist.

Integration mit OpenAPI

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

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

Abrufen von Konfigurationsdaten von Amazon EC2 EC2-Instances

Mithilfe von AWS AppConfig Agent können Sie Anwendungen integrieren AWS AppConfig , 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 abrufen. 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 Amazon EC2 EC2-Instances installieren und starten](#)
- [Schritt 3: \(Optional, aber empfohlen\) Senden von Protokolldateien an CloudWatch Logs](#)
- [Schritt 4: \(Optional\) Verwenden von Umgebungsvariablen zur Konfiguration AWS AppConfig des Agenten für 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 Amazon EC2 EC2-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 Amazon EC2 EC2-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 Amazon EC2 EC2-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 ARM64-Architekturen.
- Für verteilte Anwendungen empfehlen wir, die Installations- und Startbefehle zu den Amazon EC2 EC2-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 Amazon EC2 EC2-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 Instanzen als Ziel verwenden, indem Sie Run Command verwenden. Dies ist eine Funktion von AWS Systems Manager. Weitere Informationen finden Sie unter [AWS Systems Manager Run Command](#) im Benutzerhandbuch für AWS Systems Manager .
- AWS AppConfig Der Agent auf Amazon EC2 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 den folgenden Befehl mit Administratorrechten für x86_64-Architekturen aus:

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

Führen Sie für ARM64-Architekturen den folgenden Befehl aus:

```
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 [beschrieben Schritt 4: \(Optional\) Verwenden von Umgebungsvariablen zur Konfiguration AWS AppConfig des Agenten für Amazon EC2](#).

Informationen zu den ersten Schritten mit dem CloudWatch Agenten finden Sie unter [Sammeln von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agenten](#) im CloudWatch Amazon-Benutzerhandbuch. Sie können Quick Setup, eine Funktion von 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 Protokolldateien auf die Festplatte schreiben möchten, 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 Agent-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 `auto_removal` ist `true`, löscht der CloudWatch Agent automatisch rotierte AWS AppConfig Agenten-Protokolldateien.

Schritt 4: (Optional) Verwenden von Umgebungsvariablen zur Konfiguration AWS AppConfig des Agenten für Amazon EC2

Sie können AWS AppConfig Agent for Amazon EC2 mithilfe 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. Ersetzen Sie `file_name` 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 festgelegtDEBUG.

```
[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 EC2 konfigurieren, indem Sie die folgenden Umgebungsvariablen in einer Drop-In-Unit-Datei angeben.

Umgebungsvariable	Details	Standardwert
ACCESS_TOKEN	Diese Umgebungsvariable definiert ein Token, das bereitgestellt werden muss, wenn Konfigurationsdate n vom Agent-HTTP-Server angefordert werden. Der Wert des Tokens muss im Autorisierungsheader für HTTP-Anfragen mit dem Autorisierungstyp festgelegt werden Bearer. Ein Beispiel.	None

```
GET /applications/my_a
pp/...
```

Umgebungsvariable	Details	Standardwert
	<p style="text-align: right;">Host :</p> <p>localhost:2772</p> <p>Authorization: Bearer <token value></p>	
<p>BACKUP_DIRECTORY</p>	<p>Diese Umgebungsvariable ermöglicht es dem AWS AppConfig Agenten, eine Sicherungskopie jeder abgerufenen Konfiguration im angegebenen Verzeichnis zu speichern.</p> <div style="border: 1px solid #f08080; padding: 10px; margin-top: 10px;"> <p>⚠ Important</p> <p>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 anzuwenden. Weitere Informationen finden Sie unter Sicherheit in AWS AppConfig.</p> </div>	<p>None</p>

Umgebungsvariable	Details	Standardwert
HTTP_PORT	Diese Umgebungsvariable gibt den Port an, auf dem der HTTP-Server für den Agenten läuft.	2772
LOG_LEVEL	Diese Umgebungsvariable gibt den Detaillierungsgrad an, den der Agent protokolliert. Jede Ebene umfasst die aktuelle Ebene und alle höheren Ebenen. Bei den Variablen wird zwischen Groß- und Kleinschreibung unterschieden. Die Protokollebenen, von den meisten bis hin zu den am wenigsten detaillierten Debug, lauten wie folgt: <code>infowarn,error,,undnone</code> . Debug enthält detaillierte Informationen, einschließlich Zeitinformationen, über den Agenten.	info
LOG_PATH	Der Speicherort auf der Festplatte, in den die Protokolle geschrieben werden. Wenn nicht angegeben, werden Protokolle auf <code>stderr</code> geschrieben.	None

Umgebungsvariable	Details	Standardwert
MANIFEST	<p>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. Sie können einen der folgenden Werte eingeben:</p> <ul style="list-style-type: none">• "app:env:manifest-config"• "file:/fully/qualified/path/to/manifest.json" <p>Weitere Informationen zu diesen Funktionen finden Sie unter Zusätzliche Abruffunktionen.</p>	true
MAX_CONNECTIONS	<p>Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, von denen der Agent Konfigurationen AWS AppConfig abruft.</p>	3

Umgebungsvariable	Details	Standardwert
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
PREFETCH_LIST	Diese Umgebungsvariable gibt die Konfigurationsdaten an, die der Agent anfordert, AWS AppConfig sobald er gestartet wird.	None

Umgebungsvariable	Details	Standardwert
PRELOAD_BACKUPS	<p>Wenn auf gesetzt <code>true</code>, lädt der AWS AppConfig Agent die im gefundenen Konfigurationssicherungen <code>BACKUP_DIRECTORY</code> in den Speicher und überprüft sofort, ob eine neuere Version des Dienstes existiert. Wenn diese Option auf gesetzt ist <code>false</code>, 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.</p>	true
PROXY_HEADERS	<p>Diese Umgebungsvariable gibt Header an, die von dem Proxy benötigt werden, auf den in der <code>PROXY_URL</code> Umgebungsvariablen verwiesen wird. Der Wert ist eine durch Kommas getrennte Liste von Headern. Jeder Header verwendet das folgende Formular.</p> <div data-bbox="591 1478 1029 1558" style="border: 1px solid #ccc; border-radius: 10px; padding: 5px; margin: 10px 0;"> <pre>"header: value"</pre> </div>	None

Umgebungsvariable	Details	Standardwert
PROXY_URL	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen vom Agenten zu verwendet werden soll AWS-Services, einschließlich AWS AppConfig. HTTPS und HTTP URLs werden unterstützt.	None

Umgebungsvariable	Details	Standardwert
REQUEST_TIMEOUT	<p>Diese Umgebungsvariable steuert, wie lange der Agent auf eine Antwort wartet. AWS AppConfig Wenn der Dienst nicht antwortet, schlägt die Anfrage fehl.</p> <p>Wenn es sich bei der Anfrage um den ersten Datenabruf handelt, gibt der Agent einen Fehler an Ihre Anwendung zurück.</p> <p>Wenn das Timeout während einer Hintergrundüberprüfung auf aktualisierte Daten auftritt, protokolliert der Agent den Fehler und versucht es nach einer kurzen Verzögerung erneut.</p> <p>Sie können die Anzahl der Millisekunden für das Timeout angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: ms für Millisekunden und s für Sekunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardmäßig Millisekunden. Beispiel: 5000, 5000 ms und 5 Sekunden führen zu demselben Wert für das Anforderungs-Timeout.</p>	3000 Millisekunden

Umgebungsvariable	Details	Standardwert
ROLE_ARN	Diese Umgebungsvariable gibt den Amazon-Ressourcenamen (ARN) einer IAM-Rolle an. AWS AppConfig Der Agent übernimmt diese Rolle, um Konfigurationsdaten abzurufen.	None
ROLE_EXTERNAL_ID	Diese Umgebungsvariable gibt die externe ID an, die mit dem angenommenen Rollen-ARN verwendet werden soll.	None
ROLE_SESSION_NAME	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	None
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.	None

Umgebungsvariable	Details	Standardwert
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

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 vom Typ **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 Amazon EC2 EC2-Benutzerdaten hinzuzufügen. Sie können die Befehle zu den Benutzerdaten entweder auf der Instance oder in der EC2 Auto Scaling Scaling-Gruppe hinzufügen. Das Skript installiert und startet bei jedem Start oder Neustart einer Instance 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
```

Abrufen von Konfigurationsdaten von Amazon ECS und Amazon EKS

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 Konfigurationstoken, die für nachfolgende Serviceanfragen benötigt werden.
- 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 abrufen. Aus diesem Grund ist der erste Aufruf zum Abrufen von Daten langsamer als nachfolgende Aufrufe.

Themen

- [Bevor Sie beginnen](#)
- [Den AWS AppConfig Agenten für die Amazon ECS-Integration starten](#)
- [Den AWS AppConfig Agenten für die Amazon EKS-Integration starten](#)
- [Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon ECS und Amazon EKS](#)
- [Konfigurationsdaten werden abgerufen](#)

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 (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.

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 den AWS AppConfig Agent-Sidecar-Container zu verwenden, müssen Sie ihn starten.

Um Amazon ECS (Konsole) zu starten

1. Öffnen Sie die Konsole unter <https://console.aws.amazon.com/ecs/v2>.
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. 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 aktuell ausgeführte 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

Sie können das Standardverhalten des AWS AppConfig Agenten anpassen, indem Sie Umgebungsvariablen eingeben oder ändern. Hinweise zu den verfügbaren Umgebungsvariablen finden Sie unter [Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for 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 den AWS AppConfig Agent-Sidecar-Container zu verwenden, müssen Sie ihn starten. Das folgende Verfahren beschreibt, wie Sie das Amazon `kubectl` EKS-Befehlszeilentool verwenden, um Änderungen an der `kubeconfig` Datei für Ihre Container-Anwendung vorzunehmen. 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 Ihre kubeconfig Datei und stellen Sie sicher, dass Ihre Amazon EKS-Anwendung als Einzelcontainer-Bereitstellung ausgeführt wird. Der Inhalt der Datei sollte in 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 Ihrer YAML-Bereitstellungsdatei 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: 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 [Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon ECS und Amazon EKS](#).
- Geben Sie für **SERVICE_REGION** den AWS-Region Code an (z. B. us-west-1), aus dem der AWS AppConfig Agent die Konfigurationsdaten abrufen.

3. Führen Sie den folgenden Befehl im `kubectl` Tool aus, um die Änderungen auf Ihren Cluster anzuwenden.

```
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 [Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon ECS und Amazon EKS](#).

Verwenden von Umgebungsvariablen zur Konfiguration von AWS AppConfig Agent for Amazon ECS und Amazon EKS

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

Umgebungsvariable	Details	Standardwert
ACCESS_TOKEN	<p>Diese Umgebungsvariable definiert ein Token, das bereitgestellt werden muss, wenn Konfigurationsdate n vom Agent-HTTP-Server angefordert werden. Der Wert des Tokens muss im Autorisierungsheader für HTTP-Anfragen mit dem Autorisierungstyp festgelegt werden Bearer. Ein Beispiel.</p> <pre> GET /applications/my_a pp/... Host: localhost:2772 Authorization: Bearer <token value> </pre>	None
BACKUP_DIRECTORY	<p>Diese Umgebungsvariable ermöglicht es dem AWS AppConfig Agenten, eine Sicherungskopie jeder abgerufenen Konfiguration im angegebenen Verzeichnis zu speichern.</p>	None

Umgebungsvariable	Details	Standardwert
	<p> Important</p> <p>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 anzuwenden. Weitere Informationen finden Sie unter Sicherheit in AWS AppConfig.</p>	
HTTP_PORT	Diese Umgebungsvariable gibt den Port an, auf dem der HTTP-Server für den Agenten läuft.	2772

Umgebungsvariable	Details	Standardwert
LOG_LEVEL	<p>Diese Umgebungsvariable gibt den Detaillierungsgrad an, den der Agent protokolliert. Jede Ebene umfasst die aktuelle Ebene und alle höheren Ebenen. Bei den Variablen wird zwischen Groß- und Kleinschreibung unterschieden. Die Protokollebenen, von den meisten bis hin zu den am wenigsten detaillierten Debug, lauten wie folgt: <code>info</code>, <code>warn</code>, <code>error</code>, und <code>none</code>. Debug enthält detaillierte Informationen, einschließlich Zeitinformationen, über den Agenten.</p>	<code>info</code>

Umgebungsvariable	Details	Standardwert
MANIFEST	<p>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. Sie können einen der folgenden Werte eingeben:</p> <ul style="list-style-type: none">• "app:env:manifest-config"• "file:/fully/qualified/path/to/manifest.json" <p>Weitere Informationen zu diesen Funktionen finden Sie unter Zusätzliche Abruffunktionen.</p>	true
MAX_CONNECTIONS	<p>Diese Umgebungsvariable konfiguriert die maximale Anzahl von Verbindungen, von denen der Agent Konfigurationen AWS AppConfig abrufen kann.</p>	3

Umgebungsvariable	Details	Standardwert
POLL_INTERVAL	<p>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.</p>	45 Sekunden
PREFETCH_LIST	<p>Diese Umgebungsvariable gibt die Konfigurationsdaten an, die der Agent anfordert, AWS AppConfig sobald er gestartet wird.</p>	None

Umgebungsvariable	Details	Standardwert
PRELOAD_BACKUPS	<p>Wenn auf gesetzt <code>true</code>, lädt der AWS AppConfig Agent die im gefundenen Konfigurationssicherungen <code>BACKUP_DIRECTORY</code> in den Speicher und überprüft sofort, ob eine neuere Version des Dienstes existiert. Wenn diese Option auf gesetzt ist <code>false</code>, 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.</p>	true
PROXY_HEADERS	<p>Diese Umgebungsvariable gibt Header an, die von dem Proxy benötigt werden, auf den in der <code>PROXY_URL</code> Umgebungsvariablen verwiesen wird. Der Wert ist eine durch Kommas getrennte Liste von Headern. Jeder Header verwendet das folgende Formular.</p> <div data-bbox="594 1478 1027 1556" style="border: 1px solid #ccc; border-radius: 10px; padding: 5px; margin: 10px auto; width: fit-content;"><code>"header: value"</code></div>	None

Umgebungsvariable	Details	Standardwert
PROXY_URL	Diese Umgebungsvariable gibt die Proxy-URL an, die für Verbindungen vom Agenten zu verwendet werden soll AWS-Services, einschließlich AWS AppConfig. HTTPS und HTTP URLs werden unterstützt.	None

Umgebungsvariable	Details	Standardwert
REQUEST_TIMEOUT	<p>Diese Umgebungsvariable steuert, wie lange der Agent auf eine Antwort wartet. AWS AppConfig Wenn der Dienst nicht antwortet, schlägt die Anfrage fehl.</p> <p>Wenn es sich bei der Anfrage um den ersten Datenabruf handelt, gibt der Agent einen Fehler an Ihre Anwendung zurück.</p> <p>Wenn das Timeout während einer Hintergrundüberprüfung auf aktualisierte Daten auftritt, protokolliert der Agent den Fehler und versucht es nach einer kurzen Verzögerung erneut.</p> <p>Sie können die Anzahl der Millisekunden für das Timeout angeben. Sie können auch eine Zahl mit einer Zeiteinheit angeben: ms für Millisekunden und s für Sekunden. Wenn keine Einheit angegeben ist, verwendet der Agent standardmäßig Millisekunden. Beispiel: 5000, 5000 ms und 5 Sekunden führen zu demselben Wert für das Anforderungs-Timeout.</p>	3000 Millisekunden

Umgebungsvariable	Details	Standardwert
ROLE_ARN	Diese Umgebungsvariable gibt den Amazon-Ressourcenamen (ARN) einer IAM-Rolle an. AWS AppConfig Der Agent übernimmt diese Rolle, um Konfigurationsdaten abzurufen .	None
ROLE_EXTERNAL_ID	Diese Umgebungsvariable gibt die externe ID an, die mit dem angenommenen Rollen-ARN verwendet werden soll.	None
ROLE_SESSION_NAME	Diese Umgebungsvariable gibt den Sitzungsnamen an, der den Anmeldeinformationen für die angenommene IAM-Rolle zugeordnet werden soll.	None
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.	None

Umgebungsvariable	Details	Standardwert
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

Konfigurationsdaten werden abgerufen

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 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 vom Typ **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"
```

Zusätzliche Abruffunktionen

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

- [Abruf mehrerer Konten](#): Verwenden Sie den AWS AppConfig Agenten von einem Primärserver oder einem Abruf aus AWS-Konto , um Konfigurationsdaten von Konten mehrerer Anbieter abzurufen.
- [Schreiben Sie die Konfigurationskopie auf die Festplatte](#): 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 Anwendungen AWS AppConfig.

Über Agentenmanifeste

Um diese AWS AppConfig Agentenfunktionen 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 Schreibkonfiguration auf Festplatte und die Funktionen 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	<i>Anwendungsname</i> : <i>Umgebungsname</i> : <i>Konfigurationsname</i>	Verwenden Sie diese Methode für dynamische Updates. Sie können ein in einer Konfiguration gespeichertes Manifest AWS AppConfig genauso

Speicherort des Manifests	Wert der Umgebungsvariablen	Anwendungsfall
		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. Diese Methode ist in Containerumgebungen nützlich, in denen es einfacher ist, eine Umgebungsvariable festzulegen, 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 EC2 verwenden](#)
- [AWS AppConfig Agent mit Amazon ECS und Amazon EKS verwenden](#)

Abruf mehrerer Konten

Sie können den AWS AppConfig Agenten so konfigurieren, dass er Konfigurationen von mehreren abrufen, AWS-Konten indem Sie das Außerkraftsetzen von Anmeldeinformationen in das Agent-Manifest eingeben. AWS AppConfig 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:

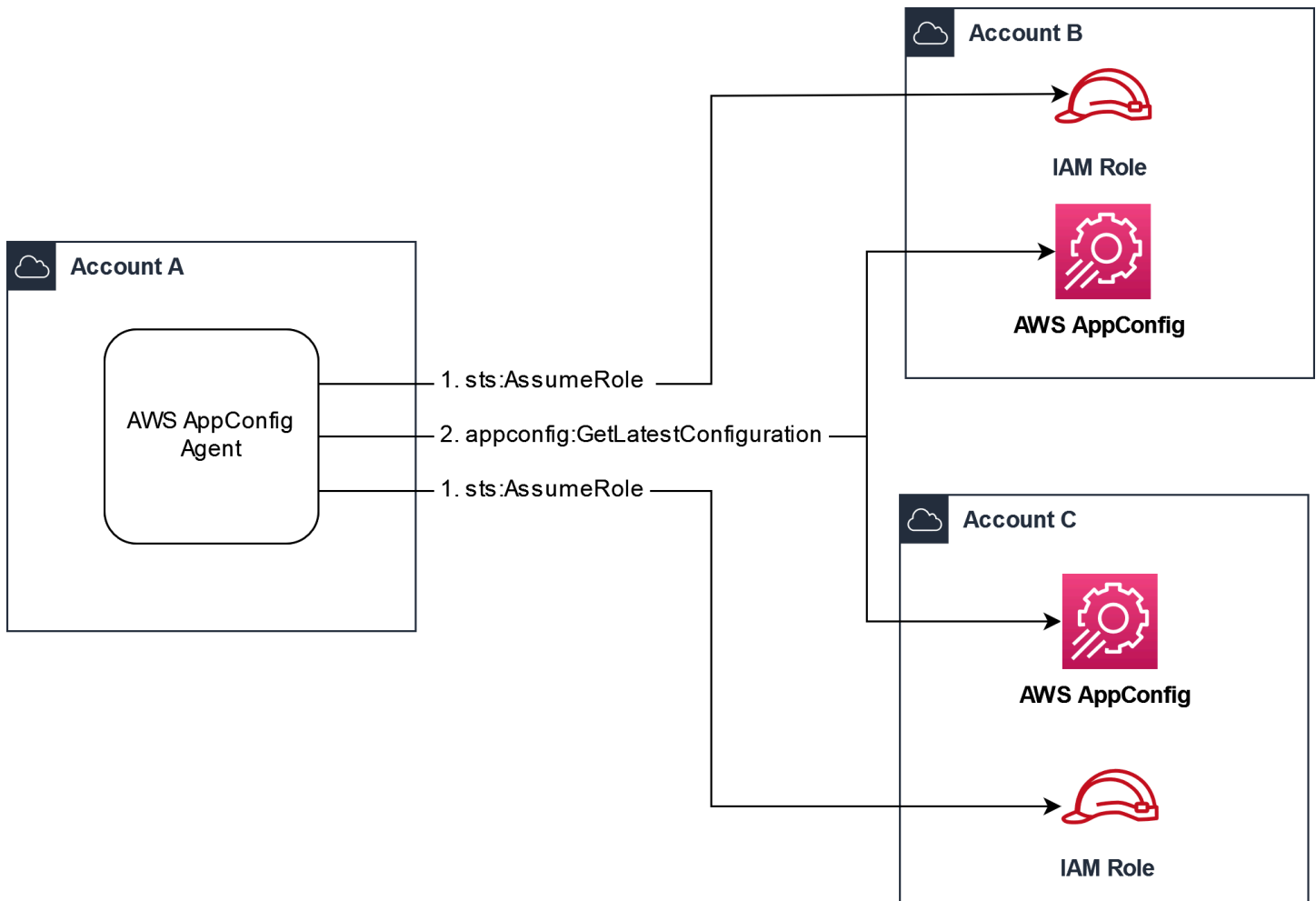
```
{
  "application_name:environment_name:configuration_name": {
```

```
    "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 Lieferantenkonto) angegeben sind, und dann den [GetLatestKonfigurations-API-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 zur Abfrage von Konfigurationsdaten aufruft. Die Lieferantenkonto 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 die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
3. Wählen Sie im Navigationsbereich Policies (Richtlinien) und dann Create policy (Richtlinie erstellen).
4. Wählen Sie die JSON-Option.
5. Ersetzen Sie im Policy-Editor das Standard-JSON durch die folgende Richtlinienanweisung. Aktualisieren Sie jeden *Beispielplatzhalter für Ressourcen* 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 die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
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 jeden *Beispielplatzhalter für Ressourcen* 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 Roles (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 werden diese Informationen im nächsten Verfahren angeben.

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
```

Schreiben Sie die Konfigurationskopie auf die Festplatte

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 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](#).

Um die Schreibkonfiguration zu aktivieren, kopieren Sie sie auf die Festplatte.

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 Konfigurationsskopie 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 '*Anwendung: Umgebung: Konfiguration*' nach *file_path* geschrieben" weist darauf hin, dass der AWS AppConfig Agent Konfigurationsskopien 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
```

AWS AppConfig Lokale Entwicklung des 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. 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. Fügen LOCAL_DEVELOPMENT_DIRECTORY Sie der Liste der Umgebungsvariablen hinzu. Geben Sie ein Verzeichnis im Dateisystem an, das dem Agenten Leserechte gewährt. z. B. /tmp/local_configs.
 4. Erstellen Sie eine Datei im Verzeichnis. Der Dateiname muss das folgende Format haben:

```
application_name:environment_name:configuration_profile_name
```

Ein Beispiel:

```
Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
```

Note

(Optional) Sie können den Inhaltstyp steuern, den der Agent für Ihre Konfigurationsdaten zurückgibt, basierend auf der Erweiterung, die Sie der Datei geben. 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.


Konfigurationen durch direktes Aufrufen von APIs abrufen

Ihre Anwendung ruft Konfigurationsdaten ab, indem sie zunächst eine Konfigurationssitzung mithilfe des [StartConfigurationSession-API-Vorgangs](#) einrichtet. Der Client Ihrer Sitzung ruft dann in regelmäßigen Abständen die [GetLatestKonfiguration](#) auf, um nach den neuesten verfügbaren Daten zu suchen und diese abzurufen.

Beim Aufrufen `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 `GetLatestKonfiguration` warten muss.

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

 Important

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

Wenn Sie aufrufen `GetLatestKonfiguration`, 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 [StartConfigurationSitzungs-API](#) 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 Abrufen einer Konfiguration.
- Wenn Ihre Konfiguration mithilfe von bereitgestellt wird `KmsKeyId`, 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.

Ein Konfigurationsbeispiel wird abgerufen

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, der Umgebung und des Konfigurationsprofils. 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 \  
  --profile profile_name_or_ID \  
  --session-token session_token
```

```
--configuration-profile-identifier configuration_profile_name_or_ID
```

Das System gibt Informationen im folgenden Format zurück.

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

Rufen Sie nach dem Start einer Sitzung mithilfe von [InitialConfigurationToken GetLatest Configuration](#) auf, um 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 von `ConfigurationToken` erhaltene `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 neuesGetLatestConfiguration.
- 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.

Erweitern von Workflows mithilfe von Erweiterungen

Eine Erweiterung erweitert Ihre Fähigkeit, Logik oder Verhalten an verschiedenen Stellen während des AWS AppConfig Workflows der Erstellung oder Bereitstellung einer Konfiguration einzubringen. Sie können beispielsweise 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.
- Bereinigen Sie den Inhalt eines Konfigurationsprofils auf sensible Daten, bevor eine Bereitstellung beginnt.
- Erstellen oder aktualisieren Sie ein Atlassian-Jira-Problem, wenn eine Änderung an einem Feature-Flag vorgenommen wird.
- Führen Sie Inhalte aus einem Service oder einer Datenquelle mit Ihren Konfigurationsdaten zusammen, wenn Sie eine Bereitstellung starten.
- Sichern Sie eine Konfiguration in einem Amazon Simple Storage Service (Amazon S3)-Bucket, wenn eine Konfiguration bereitgestellt wird.

Sie können diese Aufgabentypen mit AWS AppConfig Anwendungen, Umgebungen und Konfigurationsprofilen verknüpfen.

Inhalt

- [Informationen zu AWS AppConfig Erweiterungen](#)
- [Arbeiten mit AWS von erstellten Erweiterungen](#)
- [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#)
- [AWS AppConfig -Erweiterungsintegration mit Atlassian JSpeed](#)

Informationen zu AWS AppConfig Erweiterungen

In diesem Thema werden AWS AppConfig Erweiterungskonzepte und Terminologie vorgestellt. Die Informationen werden im Kontext jedes Schritts erörtert, der zum Einrichten und Verwenden von AWS AppConfig Erweiterungen erforderlich ist.

Themen

- [Schritt 1: Festlegen, was Sie mit Erweiterungen machen möchten](#)
- [Schritt 2: Ermitteln, wann die Erweiterung ausgeführt werden soll](#)
- [Schritt 3: Erstellen einer Erweiterungszuordnung](#)
- [Schritt 4: Bereitstellen einer Konfiguration und Überprüfen, ob die Erweiterungsaktionen ausgeführt werden](#)

Schritt 1: Festlegen, was Sie mit Erweiterungen machen möchten

Möchten Sie eine Benachrichtigung an einen Webhook erhalten, der Nachrichten an Slack sendet, wenn eine AWS AppConfig Bereitstellung abgeschlossen ist? Möchten Sie ein Konfigurationsprofil in einem Amazon Simple Storage Service (Amazon S3)-Bucket sichern, bevor eine Konfiguration bereitgestellt wird? Möchten Sie Konfigurationsdaten auf vertrauliche Informationen entfernen, bevor die Konfiguration bereitgestellt wird? Sie können Erweiterungen verwenden, um diese Arten von Aufgaben und mehr auszuführen. Sie können benutzerdefinierte Erweiterungen erstellen oder die AWS erstellten Erweiterungen verwenden, die in enthalten sind AWS AppConfig.

Note

Für die meisten Anwendungsfälle müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine - AWS Lambda Funktion erstellen, um Berechnungen und Verarbeitungen durchzuführen, die in der Erweiterung definiert sind. Weitere Informationen finden Sie unter [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Die folgenden AWS von erstellten Erweiterungen können Ihnen helfen, Konfigurationsbereitstellungen schnell in andere -Services zu integrieren. Sie können diese Erweiterungen in der - AWS AppConfig Konsole oder durch Aufrufen von Erweiterungs-[API-Aktionen](#) direkt über die AWS Tools for PowerShell AWS CLI, die oder das SDK verwenden.

Erweiterung	Beschreibung
Amazon CloudWatch -Evidently-AB-Tests	Diese Erweiterung ermöglicht es Ihrer Anwendung, Benutzersitzungen lokal Varianten zuzuweisen, anstatt die - EvaluateFeature Operation aufzurufen. Weitere Informati

Erweiterung	Beschreibung
	onen finden Sie unter Arbeiten mit der Amazon CloudWatch -Evidently-Erweiterung .
AWS AppConfig -Bereitstellungseignisse in EventBridge	Diese Erweiterung sendet Ereignisse an den EventBridge Standard-Event-Bus, wenn eine Konfiguration bereitgestellt wird.
AWS AppConfig -Bereitstellungseignisse in Amazon Simple Notification Service (Amazon SNS)	Diese Erweiterung sendet Nachrichten an ein Amazon SNS-Thema, das Sie bei der Bereitstellung einer Konfiguration angeben.
AWS AppConfig -Bereitstellungseignisse für Amazon Simple Queue Service (Amazon SQS)	Diese Erweiterung stellt Nachrichten in Ihre Amazon SQS-Warteschlange, wenn eine Konfiguration bereitgestellt wird.
Integrationserweiterung – Atlassian JSpeed	Mit diesen Erweiterungen kann Probleme erstellen und aktualisieren AWS AppConfig , wenn Sie Änderungen an einem Feature-Flag vornehmen.

Schritt 2: Ermitteln, wann die Erweiterung ausgeführt werden soll

Eine Erweiterung definiert eine oder mehrere Aktionen, die sie während eines AWS AppConfig Workflows ausführt. Die AWS erstellte AWS AppConfig deployment events to Amazon SNS Erweiterung enthält beispielsweise eine Aktion zum Senden einer Benachrichtigung an ein Amazon SNS-Thema. Jede Aktion wird entweder aufgerufen, wenn Sie mit interagieren AWS AppConfig oder wenn in Ihrem Namen einen Prozess AWS AppConfig durchführt. Diese werden als Aktionspunkte bezeichnet. AWS AppConfig Erweiterungen unterstützen die folgenden Aktionspunkte:

- PRE_CREATE_HOSTED_CONFIGURATION_VERSION
- PRE_START_DEPLOYMENT
- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE

- ON_DEPLOYMENT_ROLLED_BACK

Erweiterungsaktionen, die für PRE_* Aktionspunkte konfiguriert sind, werden nach der Anforderungvalidierung angewendet, aber bevor die Aktivität AWS AppConfig ausführt, die dem Aktionspunktnamen entspricht. Diese Aktionsaufrufe werden gleichzeitig mit einer -Anforderung verarbeitet. Wenn mehr als eine Anforderung gestellt wird, werden Aktionsaufrufe sequenziell 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 stattfindet.

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 Service den Fehler und setzt den Workflow fort.

Schritt 3: Erstellen einer 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 beispielsweise die AWS AppConfig deployment events to Amazon SNS Erweiterung ausführen und jedes Mal Benachrichtigungen zu einem Amazon SNS-Thema erhalten, wenn eine Konfigurationsbereitstellung für eine bestimmte Anwendung gestartet wird. Das Definieren, welche Aktionspunkte eine Erweiterung für eine bestimmte AWS AppConfig Ressource aufrufen, wird als Erweiterungszuordnung bezeichnet. Eine Erweiterungszuordnung ist eine angegebene 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 eine Erweiterung einer Anwendung oder Umgebung zuordnen, AWS AppConfig ruft die Erweiterung für alle Workflows auf, die sich auf die Anwendungs- oder Umgebungsressourcen beziehen, falls zutreffend.

Angenommen, Sie haben eine AWS AppConfig Anwendung namens , MobileApps die ein Konfigurationsprofil namens enthält AccessList. Angenommen, die MobileApps Anwendung umfasst Beta-, Integrations- und Produktionsumgebungen. Sie erstellen eine Erweiterungszuordnung für die AWS erstellte Amazon SNS-Benachrichtigungserweiterung und verknüpfen die Erweiterung mit der MobileApps Anwendung. Die Amazon 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 zu verwenden, aber Sie müssen eine Erweiterungszuordnung erstellen.

Schritt 4: Bereitstellen einer Konfiguration und Überprüfen, ob die Erweiterungsaktionen ausgeführt werden

Nachdem Sie eine Zuordnung erstellt haben und eine gehostete Konfiguration erstellt oder eine Konfiguration bereitgestellt wird, AWS AppConfig ruft die Erweiterung auf und führt die angegebenen Aktionen aus. Wenn eine Erweiterung aufgerufen wird und im System während eines PRE - * Aktionspunkts ein Fehler auftritt, AWS AppConfig gibt Informationen zu diesem Fehler zurück.

Arbeiten mit AWS von erstellten Erweiterungen

AWS AppConfig enthält die folgenden - AWS verfassten Erweiterungen. Diese Erweiterungen können Ihnen helfen, den AWS AppConfig Workflow in andere -Services zu integrieren. Sie können diese Erweiterungen in der AWS Management Console oder verwenden AWS CLI AWS Tools for PowerShell, indem Sie Erweiterungs-[API-Aktionen](#) direkt über die oder das SDK aufrufen.

Erweiterung	Beschreibung
Amazon CloudWatch -Evidently-A/B-Tests	Diese Erweiterung ermöglicht es Ihrer Anwendung, Benutzersitzungen lokal Varianten zuzuweisen, anstatt die -EvaluateFeature Operation aufzurufen. Weitere Informationen finden Sie unter Arbeiten mit der Amazon CloudWatch -Evidently-Erweiterung .
AWS AppConfig -Bereitstellungsereignisse in EventBridge	Diese Erweiterung sendet Ereignisse an den EventBridge Standard-Event-Bus, wenn eine Konfiguration bereitgestellt wird.

Erweiterung	Beschreibung
AWS AppConfig -Bereitstellungsereignisse in Amazon Simple Notification Service (Amazon SNS)	Diese Erweiterung sendet Nachrichten an ein Amazon SNS-Thema, das Sie bei der Bereitstellung einer Konfiguration angeben.
AWS AppConfig -Bereitstellungsereignisse in Amazon Simple Queue Service (Amazon SQS)	Diese Erweiterung stellt Nachrichten in Ihre Amazon SQS-Warteschlange, wenn eine Konfiguration bereitgestellt wird.
Integrationserweiterung – Atlassian J Bol	Mit diesen Erweiterungen kann Probleme erstellen und aktualisieren AWS AppConfig , wenn Sie Änderungen an einem Feature-Flag vornehmen.

Arbeiten mit der Amazon CloudWatch -Evidently-Erweiterung

Sie können Amazon CloudWatch Evidently verwenden, um neue Features sicher zu validieren, indem Sie sie einem bestimmten Prozentsatz Ihrer Benutzer bereitstellen, während Sie die Funktion einführen. 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 die [EvaluateFeature](#) Operation aufzurufen. Eine lokale Sitzung mindert die Latenz- und Verfügbarkeitsrisiken, die mit einem API-Aufruf verbunden sind. Informationen zum Konfigurieren und Verwenden der Erweiterung finden Sie unter [Durchführen von Starts und A/B-Experimenten mit CloudWatch Evidently](#) im Amazon- CloudWatch Benutzerhandbuch.

Arbeiten mit der **AWS AppConfig deployment events to Amazon EventBridge** Erweiterung

Die AWS AppConfig deployment events to Amazon EventBridge Erweiterung ist eine AWS von erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig

Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung sendet Ereignisbenachrichtigungen an den EventBridge Standard-Events-Bus, wenn eine Konfiguration bereitgestellt wird. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, AWS AppConfig sendet 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 Amazon-Ressourcennamen (ARN) des EventBridge Standard-Events-Bus für das URI-Feld eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unter [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Important

Diese Erweiterung unterstützt nur den EventBridge Standard-Events-Bus.

Verwenden der Erweiterung

Um die AWS AppConfig deployment events to Amazon EventBridge Erweiterung zu verwenden, fügen Sie zuerst 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 [CreateExtensionAssociation](#) -API-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN einer AWS AppConfig Anwendung, Umgebung oder eines 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.

Nachdem Sie die Zuordnung erstellt haben und eine Konfiguration für die angegebene AWS AppConfig Ressource bereitgestellt wird, AWS AppConfig ruft die Erweiterung auf und sendet Benachrichtigungen gemäß den in der Erweiterung angegebenen Aktionspunkten.

Note

Diese Erweiterung wird durch die folgenden Aktionspunkte 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 [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung zu erstellen, indem Sie entweder die AWS Systems Manager Konsole oder die verwenden AWS CLI.

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 Zu Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource für Ressourcentyp einen - AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource AWS AppConfig werden Sie von aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispielergebnis, das an 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"
}
}
```

Arbeiten mit der **AWS AppConfig deployment events to Amazon SNS** Erweiterung

Die AWS AppConfig deployment events to Amazon SNS Erweiterung ist eine AWS von erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung veröffentlicht Nachrichten in einem Amazon SNS-Thema, wenn eine Konfiguration bereitgestellt wird. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, AWS AppConfig veröffentlicht nach jedem Start, Ende und Rollback der Konfigurationsbereitstellung eine Nachricht zum Thema.

Wenn Sie mehr Kontrolle darüber haben möchten, welche Aktionspunkte Amazon SNS-Benachrichtigungen senden, können Sie eine benutzerdefinierte Erweiterung erstellen und einen Amazon-Ressourcennamen (ARN) des Amazon SNS-Themas für das URI-Feld eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unter [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Verwenden der Erweiterung

In diesem Abschnitt wird beschrieben, wie Sie die AWS AppConfig deployment events to Amazon SNS Erweiterung verwenden.

Schritt 1: Konfigurieren von AWS AppConfig zum Veröffentlichen von Nachrichten in einem Thema

Fügen Sie Ihrem Amazon SNS-Thema eine Zugriffskontrollrichtlinie hinzu, die (appconfig.amazonaws.com) AWS AppConfig Veröffentlichungsberechtigungen erteilt (sns:Publish). Weitere Informationen finden Sie unter [Beispielfälle für Amazon SNS-Zugriffskontrolle](#).

Schritt 2: Erstellen einer 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 [CreateExtensionAssociation](#) -API-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN einer AWS AppConfig Anwendung, Umgebung oder eines 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-Themas enthält, das Sie verwenden möchten.

Nachdem Sie die Zuordnung erstellt haben und eine Konfiguration für die angegebene AWS AppConfig Ressource bereitgestellt wird, AWS AppConfig ruft die Erweiterung auf und sendet Benachrichtigungen gemäß den in der Erweiterung angegebenen Aktionspunkten.

Note

Diese Erweiterung wird durch die folgenden Aktionspunkte 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 [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung zu erstellen, indem Sie entweder die AWS Systems Manager Konsole oder die verwenden AWS CLI.

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 Zu Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource für Ressourcentyp einen - AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource AWS AppConfig werden Sie von aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispiel der Nachricht, die an das Amazon 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"
    }
}
}
```

Arbeiten mit der **AWS AppConfig deployment events to Amazon SQS** Erweiterung

Die AWS AppConfig deployment events to Amazon SQS Erweiterung ist eine AWS von erstellte Erweiterung, mit der Sie den Workflow für die AWS AppConfig Konfigurationsbereitstellung überwachen und entsprechend handeln können. Die Erweiterung stellt Nachrichten in Ihre Amazon Simple Queue Service (Amazon SQS)-Warteschlange, wenn eine Konfiguration bereitgestellt wird. Nachdem Sie die Erweiterung einer Ihrer AWS AppConfig Anwendungen, Umgebungen oder Konfigurationsprofile zugeordnet haben, AWS AppConfig stellt nach jedem Start, Ende und Rollback der Konfigurationsbereitstellung eine Nachricht in die Warteschlange.

Wenn Sie mehr Kontrolle darüber haben möchten, welche Aktionspunkte Amazon SQS-Benachrichtigungen senden, können Sie eine benutzerdefinierte Erweiterung erstellen und einen Amazon-Ressourcennamen (ARN) der Amazon SQS-Warteschlange für das URI-Feld eingeben. Informationen zum Erstellen einer Erweiterung finden Sie unter [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Verwenden der Erweiterung

In diesem Abschnitt wird beschrieben, wie Sie die AWS AppConfig deployment events to Amazon SQS Erweiterung verwenden.

Schritt 1: Konfigurieren von AWS AppConfig zur Warteschlange von Nachrichten

Fügen Sie Ihrer Amazon SQS-Warteschlange eine Amazon SQS-Richtlinie hinzu, die (`appconfig.amazonaws.com`) Berechtigungen zum Senden von AWS AppConfig Nachrichten gewährt (`sqs:SendMessage`). Weitere Informationen finden Sie unter [Grundlegende Beispiele für Amazon SQS-Richtlinien](#).

Schritt 2: Erstellen einer 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 [CreateExtensionAssociation](#) -API-Aktion. Wenn Sie die Zuordnung erstellen, geben Sie den ARN einer AWS AppConfig Anwendung, Umgebung oder eines 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-Warteschlange enthält, die Sie verwenden möchten.

Nachdem Sie die Zuordnung erstellt haben und eine Konfiguration für die angegebene AWS AppConfig Ressource erstellt oder bereitgestellt wird, AWS AppConfig ruft die Erweiterung auf und sendet Benachrichtigungen gemäß den in der Erweiterung angegebenen Aktionspunkten.

Note

Diese Erweiterung wird durch die folgenden Aktionspunkte 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 [Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen](#).

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung zu erstellen, indem Sie entweder die AWS Systems Manager Konsole oder die verwenden AWS CLI.

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 Zu Ressource hinzufügen aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource für Ressourcentyp einen - AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource AWS AppConfig werden Sie von aufgefordert, andere Ressourcen auszuwählen.
5. Wählen Sie Zuordnung zur Ressource erstellen aus.

Hier ist ein Beispiel für die Nachricht, die an die Amazon 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"  
}
```

Arbeiten mit der Atlassian JCCP-Erweiterung für AWS AppConfig

Durch die Integration mit Atlassian JSpeed AWS AppConfig kann Probleme in der Atlassian-Konsole erstellen und aktualisieren, wenn Sie Änderungen an einem [Feature-Flag](#) in Ihrem AWS-Konto für das angegebene vornehmen AWS-Region. Jedes Jura-Problem enthält den Flag-Namen, die Anwendungs-ID, die Konfigurationsprofil-ID und die Flag-Werte. Nachdem Sie Ihre Flag-Änderungen aktualisiert, gespeichert und bereitgestellt haben, aktualisiert JCCP die vorhandenen Probleme mit den Details der Änderung.

Note

JSpeed aktualisiert Probleme, wenn Sie ein Feature-Flag erstellen oder aktualisieren. J Bol aktualisiert auch Probleme, wenn Sie ein Flag-Attribut auf untergeordneter Ebene aus einem Flag auf übergeordneter Ebene löschen. J Bol zeichnet keine Informationen auf, wenn Sie ein Flag auf übergeordneter Ebene löschen.

Gehen Sie wie folgt vor, um die Integration zu konfigurieren:

- [Konfigurieren von Berechtigungen für die AWS AppConfig Jura-Integration](#)
- [Konfigurieren der AWS AppConfig Jura-Integrationsanwendung](#)

Konfigurieren von Berechtigungen für die AWS AppConfig Jura-Integration

Wenn Sie die AWS AppConfig Integration mit JCCP konfigurieren, geben Sie Anmeldeinformationen für einen Benutzer an. Insbesondere geben Sie die Zugriffsschlüssel-ID und den geheimen Schlüssel des Benutzers in die Anwendung AWS AppConfig für JSpeed ein. Dieser Benutzer erteilt JCCP die Berechtigung, mit zu kommunizieren AWS AppConfig. AWS AppConfig verwendet diese Anmeldeinformationen einmal, um eine Zuordnung zwischen AWS AppConfig und JSpeed herzustellen. Die Anmeldeinformationen werden nicht gespeichert. Sie können die Zuordnung entfernen, indem Sie die AWS AppConfig für Jura-Anwendung deinstallieren.

Das Benutzerkonto benötigt eine Berechtigungsrichtlinie, 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 die - AWS AppConfig und J Bol-Integration zu erstellen:

Aufgaben

- [Aufgabe 1: Erstellen einer IAM-Berechtigungsrichtlinie für die - AWS AppConfig und Jura-Integration](#)
- [Aufgabe 2: Erstellen eines Benutzers für die - AWS AppConfig und Jura-Integration](#)

Aufgabe 1: Erstellen einer IAM-Berechtigungsrichtlinie für die - AWS AppConfig und Jura-Integration

Gehen Sie wie folgt vor, um eine IAM-Berechtigungsrichtlinie zu erstellen, die es Atlassian JCCP ermöglicht, mit zu kommunizieren AWS AppConfig. Wir empfehlen Ihnen, eine neue Richtlinie zu erstellen und diese Richtlinie an eine neue IAM-Rolle anzuhängen. Das Hinzufügen der erforderlichen Berechtigung zu einer vorhandenen IAM-Richtlinie und -Rolle widerspricht das Prinzip der geringsten Berechtigung und wird nicht empfohlen.

So erstellen Sie eine IAM-Richtlinie für die - AWS AppConfig und Jura-Integration

1. Öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich Policies (Richtlinien) und dann Create policy (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_ID* , *application_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: Markierungen.
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 eines Benutzers für die - AWS AppConfig und Jura-Integration

Gehen Sie wie folgt vor, um einen Benutzer für die Integration von AWS AppConfig und Atlassian JCCP zu erstellen. Nachdem Sie den Benutzer erstellt haben, können Sie die Zugriffsschlüssel-ID und den geheimen Schlüssel kopieren, die Sie beim Abschluss der Integration angeben.

So erstellen Sie einen Benutzer für die - AWS AppConfig und JSpeed-Integration

1. Öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
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 für 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 Vorhandene Richtlinien direkt anfügen aus. Suchen Sie nach und aktivieren Sie das Kontrollkästchen für die Richtlinie, die Sie in erstellt haben [Aufgabe 1: Erstellen einer IAM-Berechtigungsrichtlinie für die - AWS AppConfig und Jura-Integration](#), und wählen Sie dann Weiter: Tags aus.
7. Fügen Sie auf der Seite Tags hinzufügen (optional) ein oder mehrere Tag-Schlüssel-Wert-Paare hinzu, um den Zugriff für diesen Benutzer zu organisieren, zu verfolgen oder zu steuern. 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.

Konfigurieren der AWS AppConfig Jura-Integrationsanwendung

Gehen Sie wie folgt vor, um die erforderlichen Optionen in der AWS AppConfig Anwendung für JSpeed zu konfigurieren. Nachdem Sie dieses Verfahren abgeschlossen haben, erstellt JSpeed ein neues Problem für jedes Feature-Flag in Ihrem AWS-Konto für das angegebene AWS-Region. Wenn Sie Änderungen an einem Feature-Flag in vornehmen AWS AppConfig, zeichnet JSpeed die Details der vorhandenen Probleme auf.

Note

Ein AWS AppConfig Feature-Flag kann mehrere Flag-Attribute auf untergeordneter Ebene enthalten. J Bol erstellt ein Problem für jedes Feature-Flag auf übergeordneter Ebene. Wenn Sie ein Flag-Attribut auf untergeordneter Ebene ändern, können Sie die Details dieser Änderung im Jpir-Problem für das Flag auf übergeordneter Ebene anzeigen.

So konfigurieren Sie die Integration

1. Melden Sie sich beim [Atlassian Marketplace](#) an.
2. Geben Sie **AWS AppConfig** in das Suchfeld ein und drücken Sie die Eingabetaste.
3. Installieren Sie die Anwendung auf Ihrer JSpeed-Instance.
4. Wählen Sie in der Atlassischen Konsole Apps verwalten und dann AWS AppConfig für JCCP aus.
5. Wählen Sie Konfigurieren aus.
6. Wählen Sie unter Konfigurationsdetails die Option JSpeed-Projekt und dann das Projekt aus, das Sie mit Ihrem AWS AppConfig Feature-Flag verknüpfen möchten.
7. Wählen Sie und 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 die Felder Zugriffsschlüssel-ID und Geheimer Schlüssel die Anmeldeinformationen ein, die Sie in kopiert haben [Aufgabe 2: Erstellen eines Benutzers für die - AWS AppConfig und Jura-Integration](#). Optional können Sie auch ein Sitzungstoken angeben.
11. Wählen Sie Absenden aus.
12. Wählen Sie in der Atlassischen Konsole Projekte und dann das Projekt aus, das Sie für die AWS AppConfig Integration ausgewählt haben. Auf der Seite Probleme wird ein Problem für jedes Feature-Flag im angegebenen AWS-Konto und der angegebenen angezeigt AWS-Region.

Löschen der AWS AppConfig Anwendung und der Daten von für JSpeed

Wenn Sie die JSpeed-Integration nicht mehr mit AWS AppConfig Feature-Flags verwenden möchten, können Sie die AWS AppConfig Anwendung für JSpeed in der Atlassischen Konsole löschen. Das Löschen der Integrationsanwendung führt Folgendes aus:

- Löscht die Zuordnung zwischen Ihrer JSpeed-Instance und AWS AppConfig
- Löscht die Details Ihrer JCCP-Instance aus AWS AppConfig

So löschen Sie die AWS AppConfig für die JSpeed-Anwendung

1. Wählen Sie in der Atlassischen Konsole Apps verwalten aus.
2. Wählen Sie AWS AppConfig für Jira aus.
3. Wählen Sie Deinstallieren.

Walkthrough: Erstellen von benutzerdefinierten AWS AppConfig Erweiterungen

Führen Sie die folgenden Schritte aus, um eine benutzerdefinierte AWS AppConfig Erweiterung zu erstellen. Jede Aufgabe wird in späteren Themen ausführlicher beschrieben.

Note

Sie können Beispiele für benutzerdefinierte AWS AppConfig Erweiterungen auf anzeigen GitHub:

- [Beispielweiterung, die Bereitstellungen mit einem blocked day Mauskalender mithilfe von Systems Manager Change Calendar verhindert](#)

- [Beispielweiterung, die verhindert, dass Secrets mithilfe von git-secrets in Konfigurationsdaten gelangen](#)
- [Beispielweiterung, die verhindert, dass persönlich identifizierbare Informationen \(PII\) mithilfe von Amazon Comprehend in Konfigurationsdaten gelangen](#)

1. Erstellen einer - AWS Lambda Funktion

Für die meisten Anwendungsfälle müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine - AWS Lambda Funktion erstellen, um Berechnungen und Verarbeitungen durchzuführen, die in der Erweiterung definiert sind. Eine Ausnahme von dieser Regel ist, wenn Sie benutzerdefinierte Versionen der [AWS erstellten Benachrichtigungserweiterungen](#) erstellen, um Aktionspunkte hinzuzufügen oder zu entfernen. Weitere Informationen zu dieser Ausnahme finden Sie unter [Erstellen einer benutzerdefinierten AWS AppConfig Erweiterung](#).

2. Konfigurieren von Berechtigungen für Ihre benutzerdefinierte Erweiterung

Um 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 enthält.
- Erstellen Sie eine Ressourcenrichtlinie mithilfe der Lambda [AddPermission](#)-API-Aktion .

In dieser Anleitung wird beschrieben, wie Sie die IAM-Servicerolle erstellen.

3. Erstellen einer Erweiterung

Sie können eine Erweiterung erstellen AWS CLI, indem Sie die - AWS AppConfig Konsole verwenden oder die [CreateExtension](#) -API-Aktion über die AWS Tools for PowerShell oder das - SDK aufrufen. Die Anleitung verwendet die -Konsole.

4. Erstellen einer Erweiterungszuordnung

Sie können eine Erweiterungszuordnung erstellen AWS CLI, indem Sie die - AWS AppConfig Konsole oder die [CreateExtensionAssociation](#)-API-Aktion über die AWS Tools for PowerShell oder das -SDK aufrufen. Die Anleitung verwendet die -Konsole.

5. Ausführen einer Aktion, die die Erweiterung aufruft

Nachdem Sie die Zuordnung erstellt haben, AWS AppConfig ruft die Erweiterung auf, wenn die von der Erweiterung definierten Aktionspunkte für diese Ressource

auftreten. 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 gehostete Konfigurationsversion erstellen.

In den Themen in diesem Abschnitt werden alle Aufgaben beschrieben, die an der Erstellung einer benutzerdefinierten AWS AppConfig Erweiterung beteiligt sind. 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 wird (`PRE_START_DEPLOYMENT`).

Themen

- [Erstellen einer Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung](#)
- [Konfigurieren von Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung](#)
- [Erstellen einer benutzerdefinierten AWS AppConfig Erweiterung](#)
- [Erstellen einer Erweiterungszuordnung für eine benutzerdefinierte AWS AppConfig Erweiterung](#)
- [Ausführen einer Aktion, die eine benutzerdefinierte AWS AppConfig Erweiterung aufruft](#)

Erstellen einer Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung

Für die meisten Anwendungsfälle müssen Sie zum Erstellen einer benutzerdefinierten Erweiterung eine - AWS Lambda Funktion erstellen, um Berechnungen und Verarbeitungen durchzuführen, die in der Erweiterung definiert sind. Dieser Abschnitt enthält Beispielcode für eine Lambda-Funktion für eine benutzerdefinierte AWS AppConfig Erweiterung. Dieser Abschnitt enthält auch Details zur Nutzlastanforderung und Antwortreferenz. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter [Erste Schritte mit Lambda](#) im AWS Lambda Entwicklerhandbuch für .

Beispiel-Code

Der folgende Beispielcode für eine Lambda-Funktion sichert beim Aufruf automatisch eine - AWS AppConfig Konfiguration in einem Amazon S3-Bucket. Die Konfiguration wird 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. Durch die Verwendung von Erweiterungsparametern kann der Benutzer die Erweiterung an mehrere

Anwendungen anfügen und Konfigurationen an verschiedene Buckets sichern. Das Codebeispiel enthält Kommentare zur weiteren Erläuterung der Funktion.

Beispiel für eine Lambda-Funktion für eine AWS AppConfig Erweiterung

```
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 während dieser Anleitung 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)-Übernahmerolle im nächsten Abschnitt erstellen. Sie geben den ARN und den Namen an, wenn Sie die Erweiterung erstellen.

Nutzlastreferenz

Dieser Abschnitt enthält Details zur Nutzlastanforderung und Antwortreferenz für die Arbeit mit benutzerdefinierten AWS AppConfig Erweiterungen.

Anforderungsstruktur

PreCreateHostedConfigurationVersion

```
{
```

```

'InvocationId': 'vlns753', // id for specific invocation
'Parameters': {
  'ParameterOne': 'ValueOne',
  'ParameterTwo': 'ValueTwo'
},
'ContentType': 'text/plain',
'ContentVersion': '2',
'Content': 'SGVsbG8gZWYdGgh', // 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': 'SGVsbG8gZWYdGgh',
  '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': 'o2xbtn7',
  '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'
}
```

Antwortstruktur

Die folgenden Beispiele zeigen, was Ihre Lambda-Funktion als Antwort auf die Anforderung einer benutzerdefinierten AWS AppConfig Erweiterung zurückgibt.

Synchrone Ereignisse – erfolgreiche Antwort

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

Nichts zurückgeben.

Alle Fehlerereignisse

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

Konfigurieren von Berechtigungen für eine benutzerdefinierte AWS AppConfig Erweiterung

Gehen Sie wie folgt vor, um eine AWS Identity and Access Management (IAM)-Servicerolle zu erstellen und zu konfigurieren (oder Rolle zu übernehmen). AWS AppConfig verwendet diese Rolle, um die Lambda-Funktion aufzurufen.

So erstellen Sie eine IAM-Servicerolle und erlauben AWS AppConfig, sie zu übernehmen

1. Öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und dann Create role (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. Ersetzen Sie *Ihren Lambda-ARN* durch den Amazon-Ressourcennamen (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: Markierungen.
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 Symbol Aktualisieren 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 aus.
12. Geben Sie auf der Seite Name, Überprüfung und Erstellung einen Namen in das Feld Rollenname ein und geben Sie dann eine Beschreibung ein.
13. Wählen Sie Create role (Rolle erstellen) aus. Das System leitet Sie zur Seite Roles (Rollen) zurück. Wählen Sie im Banner Rolle anzeigen aus.
14. Kopieren Sie den ARN. Sie geben diesen ARN an, wenn Sie die Erweiterung erstellen.

Erstellen einer benutzerdefinierten AWS AppConfig Erweiterung

Eine Erweiterung definiert eine oder mehrere Aktionen, die sie während eines AWS AppConfig Workflows ausführt. Die AWS erstellte AWS AppConfig deployment events to Amazon SNS Erweiterung enthält beispielsweise eine Aktion zum Senden einer Benachrichtigung an ein Amazon SNS-Thema. Jede Aktion wird entweder aufgerufen, wenn Sie mit interagieren AWS AppConfig oder wenn in Ihrem Namen einen Prozess AWS AppConfig durchführt. Diese werden als Aktionspunkte bezeichnet. AWS AppConfig Erweiterungen unterstützen die folgenden Aktionspunkte:

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

Erweiterungsaktionen, die für PRE_* Aktionspunkte konfiguriert sind, werden nach der Anforderungsvalidierung angewendet, aber bevor die Aktivität AWS AppConfig ausführt, die dem Aktionspunktnamen entspricht. Diese Aktionsaufrufe werden gleichzeitig mit einer -Anforderung verarbeitet. Wenn mehr als eine Anforderung gestellt wird, werden Aktionsaufrufe sequenziell 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 stattfindet.

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 Service den Fehler und setzt den Workflow fort.

Die folgende Beispielerweiterung definiert eine Aktion, die den PRE_CREATE_HOSTED_CONFIGURATION_VERSION Aktionspunkt aufruft. Im Uri Feld gibt die Aktion den Amazon-Ressourcennamen (ARN) der MyS3ConfigurationBackUpExtension Lambda-Funktion an, die zuvor in dieser Anleitung erstellt wurde. Die Aktion gibt auch den ARN der AWS Identity and Access Management (IAM)-Übernehmerrolle an, der zuvor in dieser Anleitung erstellt wurde.

AWS AppConfig Beispielerweiterung

```
{
  "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 zum Anzeigen der Anforderungssyntax und Feldbeschreibungen beim Erstellen einer Erweiterung finden Sie im [CreateExtension](#) Thema in der APIAWS AppConfig -Referenz zu .

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 für Erweiterungsname einen eindeutigen Namen ein. Geben Sie für die Zwecke dieser Anleitung ein **MyS3ConfigurationBackupExtension**. 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 Sie für die Zwecke dieser Anleitung ein **PreCreateHostedConfigVersionActionForS3Backup**. Dieser Name beschreibt den von der Aktion verwendeten Aktionspunkt und den Erweiterungszweck.
7. Wählen Sie in der Liste Aktionspunkt die Option **PRE_CREATE_HOSTED_CONFIGURATION_VERSION** aus.
8. Wählen Sie für URI die Option Lambda-Funktion und dann die Funktion in der Liste der Lambda-Funktionen aus. Wenn Ihre Funktion nicht angezeigt wird, stellen Sie sicher, dass Sie sich in derselben befinden AWS-Region , in der Sie die Funktion erstellt haben.
9. Wählen Sie für IAM-Rolle die Rolle aus, die Sie zuvor in dieser Anleitung erstellt haben.
10. Wählen Sie im Abschnitt Erweiterungsparameter (optional) die Option Neuen Parameter hinzufügen aus.
11. Geben Sie unter Parametername einen Namen ein. Geben Sie für die Zwecke dieser Anleitung ein **S3_BUCKET**.
12. Wiederholen Sie die Schritte 5 bis 11, um eine zweite Aktion für den **PRE_START_DEPLOYMENT** Aktionspunkt zu erstellen.
13. Wählen Sie Erweiterung erstellen aus.

Anpassen von Erweiterungen AWS für erstellte Benachrichtigungen

Sie müssen kein Lambda oder eine Erweiterung erstellen, um [AWS die von erstellten Benachrichtigungserweiterungen](#) zu verwenden. Sie können einfach eine Erweiterungszuordnung erstellen und dann einen Vorgang ausführen, der einen der unterstützten Aktionspunkte aufruft. Standardmäßig unterstützen die AWS Erweiterungen für erstellte Benachrichtigungen 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 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 Standardaktionspunkte entfernen möchten, die der AWS erstellten Version zugewiesen sind.

Sie müssen keine Lambda-Funktion erstellen, wenn Sie benutzerdefinierte Versionen der AWS erstellten Benachrichtigungserweiterungen erstellen. Sie müssen nur einen Amazon-Ressourcennamen (ARN) im `Uri` Feld für die neue Erweiterungsversion angeben.

- Geben Sie für eine benutzerdefinierte EventBridge Benachrichtigungserweiterung den ARN der EventBridge Standardereignisse in das `Uri` Feld ein.
- Geben Sie für eine benutzerdefinierte Amazon SNS-Benachrichtigungserweiterung den ARN eines Amazon SNS-Themas in das `Uri` Feld ein.
- Geben Sie für eine benutzerdefinierte Amazon SQS-Benachrichtigungserweiterung den ARN einer Amazon SQS-Nachrichtenwarteschlange in das `Uri` Feld ein.

Erstellen einer Erweiterungszuordnung für eine benutzerdefinierte AWS AppConfig Erweiterung

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 beispielsweise die AWS AppConfig deployment events to Amazon SNS Erweiterung ausführen und jedes Mal Benachrichtigungen zu einem Amazon SNS-Thema erhalten, wenn eine Konfigurationsbereitstellung für eine bestimmte Anwendung gestartet wird. Das Definieren, welche Aktionspunkte eine Erweiterung für eine bestimmte AWS AppConfig Ressource aufrufen, wird als Erweiterungszuordnung bezeichnet. Eine Erweiterungszuordnung ist eine angegebene 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 eine Erweiterung einer Anwendung oder Umgebung zuordnen, AWS AppConfig ruft die Erweiterung für alle Workflows auf, die sich auf die Anwendungs- oder Umgebungsressourcen beziehen, falls zutreffend.

Angenommen, Sie haben eine AWS AppConfig Anwendung namens , MobileApps die ein Konfigurationsprofil namens enthält AccessList. Angenommen, die MobileApps Anwendung umfasst Beta-, Integrations- und Produktionsumgebungen. Sie erstellen eine Erweiterungszuordnung für die AWS erstellte Amazon SNS-Benachrichtigungserweiterung und verknüpfen die Erweiterung mit der MobileApps Anwendung. Die Amazon SNS-Benachrichtigungserweiterung wird jedes Mal aufgerufen, wenn die Konfiguration für die Anwendung in einer der drei Umgebungen bereitgestellt wird.

Gehen Sie wie folgt vor, um eine AWS AppConfig Erweiterungszuordnung mithilfe der AWS AppConfig Konsole 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 eine Optionsschaltfläche für eine Erweiterung und dann Zur Ressource hinzufügen aus. Wählen Sie für die Zwecke dieser Anleitung MyS3ConfigurationBackUpExtension aus.
4. Wählen Sie im Abschnitt Details zur Erweiterungsressource für Ressourcentyp einen - AWS AppConfig Ressourcentyp aus. Abhängig von der ausgewählten Ressource AWS AppConfig

werden Sie von aufgefordert, andere Ressourcen auszuwählen. Wählen Sie für die Zwecke dieser Anleitung Anwendung aus.

5. Wählen Sie eine Anwendung in der Liste aus.
6. Überprüfen Sie im Abschnitt Parameter, ob S3_BUCKET im Feld Schlüssel aufgeführt ist. Fügen Sie im Feld Wert den ARN der Lambda-Erweiterungen ein. Zum Beispiel: `arn:aws:lambda:aws-region:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Wählen Sie Zuordnung zur Ressource erstellen aus.

Ausführen einer Aktion, die eine benutzerdefinierte AWS AppConfig Erweiterung aufruft

Nachdem Sie die Zuordnung erstellt haben, können Sie die `MyS3ConfigurationBackUpExtension` Erweiterung aufrufen, indem Sie ein neues Konfigurationsprofil erstellen, das `hosted` für die `angibtSourceUri`. Im Rahmen des Workflows zum Erstellen der neuen Konfiguration AWS AppConfig erfährt den `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` Aktionspunkt. Wenn Sie diesen Aktionspunkt aufrufen, wird die `MyS3ConfigurationBackUpExtension` Erweiterung aufgerufen, die die neu erstellte Konfiguration automatisch in dem S3-Bucket sichert, der im `Parameter` Abschnitt der Erweiterungszuordnung angegeben ist.

AWS AppConfig -Erweiterungsintegration mit Atlassian JSpeed

AWS AppConfig lässt sich in Atlassian J Bol integrieren. Integration ermöglicht es AWS AppConfig , Probleme in der Atlassian-Konsole zu erstellen und zu aktualisieren, wenn Sie Änderungen an einem Feature-Flag in Ihrem AWS-Konto für das angegebene vornehmen AWS-Region. Jedes Jura-Problem enthält den Flag-Namen, die Anwendungs-ID, die Konfigurationsprofil-ID und die Flag-Werte. Nachdem Sie Ihre Flag-Änderungen aktualisiert, gespeichert und bereitgestellt haben, aktualisiert JCCP die vorhandenen Probleme mit den Details der Änderung. Weitere Informationen finden Sie unter [Arbeiten mit der Atlassian JCCP-Erweiterung für AWS AppConfig](#).

AWS AppConfig-Codebeispiele

Dieser Abschnitt enthält Codebeispiele für die programmgesteuerte Ausführung gängiger AWS AppConfig Aktionen. Wir empfehlen Ihnen, diese Beispiele mit den [-Java-](#), [PythonJavaScript-](#) und [-](#) SDKs zu verwenden, um die Aktionen in einer Testumgebung auszuführen. Dieser Abschnitt enthält ein Codebeispiel zum Bereinigen Ihrer Testumgebung, nachdem Sie fertig sind.

Themen

- [Erstellen oder Aktualisieren einer im gehosteten Konfigurationsspeicher gespeicherten Freiformkonfiguration](#)
- [Erstellen eines Konfigurationsprofils für ein in Secrets Manager gespeichertes Secret](#)
- [Bereitstellen eines Konfigurationsprofils](#)
- [Verwenden von -AWS AppConfigAgent zum Lesen eines Freiform-Konfigurationsprofils](#)
- [Verwenden von AWS AppConfig Agent zum Lesen eines bestimmten Feature-Flags](#)
- [Verwenden der GetLatestConfig API-Aktion zum Lesen eines Freiform-Konfigurationsprofils](#)
- [Bereinigen Ihrer Umgebung](#)

Erstellen oder Aktualisieren einer im gehosteten Konfigurationsspeicher gespeicherten Freiformkonfiguration

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die vom Code ausgeführt werden. Die Beispiele in diesem Abschnitt rufen die folgenden APIs auf:

- [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 Secret

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die vom Code ausgeführt werden. Die Beispiele in diesem Abschnitt rufen die folgenden APIs auf:

- [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 vom Code ausgeführt werden. Die Beispiele in diesem Abschnitt rufen die folgenden APIs auf:

- [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 AppConfigAgent zum Lesen eines Freiform-Konfigurationsprofils

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die vom 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 Simplified retrieval methods
    */

    // The agent runs a local HTTP server that serves configuration data
    // Make a GET request to the agent's local server to retrieve the
    configuration data
    URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyConfigProfile");
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestMethod("GET");
    StringBuilder content;
    try (BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()))) {
        content = new StringBuilder();
        int ch;
        while ((ch = in.read()) != -1) {
            content.append((char) ch);
        }
    }
    con.disconnect();
    System.out.println("Configuration from agent via HTTP: " + content);
}
```

Python

```
# in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
```

```
# the agent is a sidecar process that handles retrieving configuration data from AWS
AppConfig
# for you in a way that implements best practices like configuration caching.
#
# for more information about the agent, see
# Simplified retrieval methods
#

import requests

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

# the agent runs a local HTTP server that serves configuration data
# make a GET request to the agent's local server to retrieve the configuration data
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}")
config = response.content
```

JavaScript

```
// in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
// the agent is a sidecar process that handles retrieving configuration data from
AppConfig
// for you in a way that implements best practices like configuration caching.

// for more information about the agent, see
// Simplified retrieval methods

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// the agent runs a local HTTP server that serves configuration data
// make a GET request to the agent's local server to retrieve the configuration data
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}`;
const response = await fetch(url);
const config = await response.text(); // (use `await response.json()` if your config
is json)
```


Verwenden von AWS AppConfig Agent zum Lesen eines bestimmten Feature-Flags

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die vom Code ausgeführt werden.

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 der GetLatestConfig API-Aktion zum Lesen eines Freiform-Konfigurationsprofils

Jedes der folgenden Beispiele enthält Kommentare zu den Aktionen, die vom Code ausgeführt werden. Die Beispiele in diesem Abschnitt rufen die folgenden APIs auf:

- [GetLatestConfiguration](#)
- [StartConfigurationSession](#)

Java

```
public void retrieveConfigFromApi() {
    /*
       The example below uses two AppConfigData APIs: StartConfigurationSession and
       GetLatestConfiguration.
       For more information on these APIs, see AWS AppConfig Data */
    AppConfigDataClient appConfigData = AppConfigDataClient.create();

    /*
```

Start a new configuration session using the `StartConfigurationSession` API. This operation does not return configuration data.

Rather, it returns an initial configuration token that should be passed to `GetLatestConfiguration`.

IMPORTANT: This operation should only be performed once (per configuration), prior to the first `GetLatestConfiguration`

call you perform. Each `GetLatestConfiguration` will return a new configuration token that you should then use in the next `GetLatestConfiguration` call.

```
*/
```

```
StartConfigurationSessionResponse session =
    appConfigData.startConfigurationSession(req -> req
        .applicationIdentifier("MyDemoApp")
        .configurationProfileIdentifier("MyConfigProfile")
        .environmentIdentifier("Beta"));
```

```
/*
```

Retrieve configuration data using the `GetLatestConfiguration` API. The first time you call this API your configuration data will be returned. You should cache that data (and the configuration token) and update that cache asynchronously by regularly polling the `GetLatestConfiguration` API in a background thread. If you already have the latest configuration data, subsequent `GetLatestConfiguration` calls will return an empty response. If you then deploy updated configuration data the next time you call `GetLatestConfiguration` it will return that updated data.

You can also avoid all the complexity around writing this code yourself by leveraging our agent instead.

For more information about the agent, see [Simplified retrieval methods](#)

```
*/
```

```
// The first getLatestConfiguration call uses the token from
StartConfigurationSession
String configurationToken = session.initialConfigurationToken();
GetLatestConfigurationResponse configuration =

appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationToken

System.out.println("Configuration retrieved via API: " +
configuration.configuration().asUtf8String());
```

```

        // You'll want to hold on to the token in the getLatestConfiguration
response because you'll need to use it
        // the next time you call
        configurationToken = configuration.nextPollConfigurationToken();
        configuration =

appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationToken

        // Try creating a new deployment at this point to see how the output below
changes.
        if (configuration.configuration().asByteArray().length != 0) {
            System.out.println("Configuration contents have changed
since the last GetLatestConfiguration call, new contents = " +
configuration.configuration().asUtf8String());
        } else {
            System.out.println("GetLatestConfiguration returned an empty response
because we already have the latest configuration");
        }
    }
}

```

Python

```

# the example below uses two AppConfigData APIs: StartConfigurationSession and
GetLatestConfiguration.
#
# for more information on these APIs, see
# AWS AppConfig Data
#

import boto3

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

appconfigdata = boto3.client('appconfigdata')

# start a new configuration session.
# this operation does not return configuration data.
# rather, it returns an initial configuration token that should be passed to
GetLatestConfiguration.
#
# note: this operation should only be performed once (per configuration).

```

```
# all subsequent calls to AppConfigData should be via GetLatestConfiguration.
scs = appconfigdata.start_configuration_session(
    ApplicationIdentifier=application_name,
    EnvironmentIdentifier=environment_name,
    ConfigurationProfileIdentifier=config_profile_name)
initial_token = scs['InitialConfigurationToken']

# retrieve configuration data from the session.
# this operation returns your configuration data.
# each invocation of this operation returns a unique token that should be passed to
# the subsequent invocation.
#
# note: this operation does not always return configuration data after the first
# invocation.
# data is only returned if the configuration has changed within AWS AppConfig
# (i.e. a deployment occurred).
# therefore, you should cache the data returned by this call so that you can use
# it later.
glc = appconfigdata.get_latest_configuration(ConfigurationToken=initial_token)
config = glc['Configuration'].read()
```

JavaScript

```
// the example below uses two AppConfigData APIs: StartConfigurationSession and
// GetLatestConfiguration.

// for more information on these APIs, see
// AWS AppConfig Data

import {
    AppConfigDataClient,
    GetLatestConfigurationCommand,
    StartConfigurationSessionCommand,
} from "@aws-sdk/client-appconfigdata";

const appconfigdata = new AppConfigDataClient();

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// start a new configuration session.
// this operation does not return configuration data.
```

```
// rather, it returns an initial configuration token that should be passed to
// GetLatestConfiguration.
//
// note: this operation should only be performed once (per configuration).
// all subsequent calls to AppConfigData should be via GetLatestConfiguration.
const scs = await appconfigdata.send(
  new StartConfigurationSessionCommand({
    ApplicationIdentifier: application_name,
    EnvironmentIdentifier: environment_name,
    ConfigurationProfileIdentifier: config_profile_name,
  })
);
const { InitialConfigurationToken } = scs;

// retrieve configuration data from the session.
// this operation returns your configuration data.
// each invocation of this operation returns a unique token that should be passed to
// the subsequent invocation.
//
// note: this operation does not always return configuration data after the first
// invocation.
// data is only returned if the configuration has changed within AWS AppConfig
// (i.e. a deployment occurred).
// therefore, you should cache the data returned by this call so that you can use
// it later.
const glc = await appconfigdata.send(
  new GetLatestConfigurationCommand({
    ConfigurationToken: InitialConfigurationToken,
  })
);
const config = glc.Configuration.transformToString();
```

Bereinigen Ihrer Umgebung

Wenn Sie eines oder mehrere der Codebeispiele in diesem Abschnitt ausgeführt haben, empfehlen wir Ihnen, eines der folgenden Beispiele zu verwenden, um die von diesen Codebeispielen erstellten AWS AppConfig Ressourcen zu finden und zu löschen. Die Beispiele in diesem Abschnitt rufen die folgenden APIs auf:

- [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().forEach(
-> {
        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['Id'])]
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=application['Id'],
    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})`)
}
}
```

Sicherheit in AWS AppConfig

Cloud-Sicherheit hat bei AWS höchste Priorität. Als AWS-Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die eingerichtet wurde, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit ist eine übergreifende Verantwortlichkeit zwischen AWS und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud selbst und als Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der [AWS-Compliance-Programme](#) regelmäßig. Informationen zu den Compliance-Programmen, die für AWS Systems Manager gelten, finden Sie unter [Im Rahmen des Compliance-Programms zugelassene AWS-Services](#).
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Service bestimmt, den Sie verwenden. 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 eine Funktion von AWS Systems Manager. Informationen zur Anwendung des Modells der geteilten Verantwortung bei der Verwendung von AWS AppConfig finden Sie unter [Sicherheit in AWS Systems Manager](#). In diesem Abschnitt wird beschrieben, wie Sie Systems Manager konfigurieren, um die Sicherheits- und Compliance-Ziele für zu erreichen AWS AppConfig.

Implementieren des Zugriffs mit geringsten Berechtigungen

Erteilen Sie als bewährte Sicherheitsmethode die mindestens erforderlichen Berechtigungen, die Identitäten benötigen, um bestimmte Aktionen für bestimmte Ressourcen unter bestimmten Bedingungen durchzuführen. AWS AppConfig Agent bietet zwei Funktionen, mit denen der Agent auf das Dateisystem einer Instance oder eines Containers zugreifen kann: Sicherung und Schreiben auf die 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 Zugriff mit den geringsten Rechten gewähren](#) im AWS Well-Architected Tool - Benutzerhandbuch. Weitere Informationen zu den in diesem Abschnitt erwähnten AWS AppConfig Kundendienstmitarbeiterfunktionen finden Sie unter [Zusätzliche Abruffunktionen](#).

Datenverschlüsselung im Ruhezustand für AWS AppConfig

AWS AppConfig bietet standardmäßig Verschlüsselung, um Kundendaten im Ruhezustand mit zu schützen 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 nicht anzeigen, verwalten oder verwenden AWS-eigene Schlüssel oder deren Verwendung überprüfen. 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 alternativen Verschlüsselungstyp auswählen, Sie können jedoch einen vom Kunden verwalteten Schlüssel angeben, der beim Speichern von im AWS AppConfig Datenspeicher gehosteten Konfigurationsdaten und bei der Bereitstellung Ihrer Konfigurationsdaten verwendet werden soll.

Kundenverwaltete Schlüssel – AWS AppConfig unterstützt die Verwendung eines symmetrischen kundenverwalteten Schlüssels, den Sie erstellen, besitzen und verwalten, um eine zweite Verschlüsselungsebene über die vorhandenen hinzuzufügen AWS-eigener Schlüssel. Da Sie die volle Kontrolle über diese Verschlüsselungsebene haben, können Sie beispielsweise folgende Aufgaben ausführen:

- Einrichtung und Pflege wichtiger Richtlinien und Erteilungen
- Einrichtung und Pflege 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 [Kundenverwalteter 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 eine `KmsKeyId` für das entsprechende Konfigurationsprofil festlegen. Jedes Mal, wenn eine neue Version von Konfigurationsdaten mit der `CreateHostedConfigurationVersion`-API-Operation erstellt wird, AWS AppConfig generiert einen AWS KMS Datenschlüssel aus dem `KmsKeyId` um die Daten vor dem Speichern zu verschlüsseln. Wenn später entweder während der `GetHostedConfigurationVersion` oder `StartDeployment`-API-Operationen auf die Daten zugegriffen wird, AWS AppConfig entschlüsselt die Konfigurationsdaten mithilfe von Informationen über den generierten Datenschlüssel.

AWS AppConfig bietet auch Unterstützung für die vom Kunden verwaltete Schlüsselverschlüsselung für bereitgestellte Konfigurationsdaten. Um Konfigurationsdaten zu verschlüsseln, können Kunden einen `KmsKeyId` für ihre Bereitstellung bereitstellen. AWS AppConfig generiert den AWS KMS Datenschlüssel `aws:kms:GenerateDataKey`, um Daten im `APIStartDeployment`-Vorgang zu verschlüsseln.

AWS AppConfig Verschlüsselungszugriff

Verwenden Sie beim Erstellen eines kundenverwalteten 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 Identitätsaufruf die folgende Richtlinienanweisung, 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-Secret oder andere Konfigurationsdaten verwenden, `kms:Decrypt` die mit einem vom Kunden verwalteten Schlüssel verschlüsselt `retrievalRoleArn` sind, muss Ihr die Daten entschlüsseln und abrufen.

```

{
  "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 [StartDeployment](#) -API-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 der AWS AppConfig [GetLatestConfiguration](#) -API-Operation `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 zur Verschlüsselung von Daten aufnehmen, bindet AWS KMS den Verschlüsselungskontext an die verschlüsselten Daten. Zur Entschlüsselung von Daten müssen Sie denselben Verschlüsselungskontext in der Anfrage übergeben.

AWS AppConfig Verschlüsselungskontext : AWS AppConfig verwendet einen AWS KMS Verschlüsselungskontext in allen kryptografischen Operationen für verschlüsselte gehostete Konfigurationsdaten und Bereitstellungen. Der Kontext enthält einen Schlüssel, der dem Datentyp entspricht, und einen Wert, der das spezifische Datenelement identifiziert.

Überwachen Ihrer Verschlüsselungsschlüssel für AWS

Wenn Sie einen vom AWS KMS Kunden verwalteten Schlüssel mit verwenden AWS AppConfig, können Sie AWS CloudTrail oder Amazon CloudWatch Logs verwenden, um Anforderungen zu verfolgen, die an AWS AppConfig sendet AWS KMS.

Das folgende Beispiel ist ein CloudTrail Ereignis für Decrypt, um AWS KMS Vorgänge zu überwachen, die von 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 herzustellen AWS AppConfig. Sie können auf zugreifen, AWS AppConfig als wäre es in Ihrer VPC, ohne die Verwendung eines Internet-Gateways, NAT-Geräts, einer VPN-Verbindung oder einer -AWS Direct Connect Verbindung. Instances in Ihrer VPC benötigen für den Zugriff auf keine öffentlichen IP-Adressen AWS AppConfig.

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.

Hinweise zu AWS AppConfig

Bevor Sie einen Schnittstellenendpunkt für einrichten AWS AppConfig, lesen Sie [Überlegungen](#) im AWS PrivateLink -Leitfaden.

AWS AppConfig unterstützt Aufrufe der [appconfigdata](#) Services [appconfig](#) und über den Schnittstellenendpunkt.

Einen Schnittstellen-Endpunkt für AWS AppConfig erstellen

Sie können einen Schnittstellenendpunkt für entweder AWS AppConfig über die Amazon-VPC-Konsole oder die AWS Command Line Interface (AWS CLI) erstellen. Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im AWS PrivateLink-Leitfaden.

Erstellen Sie einen Schnittstellenendpunkt für AWS AppConfig unter Verwendung der folgenden Servicenamen:

```
com.amazonaws.region.appconfig
```

```
com.amazonaws.region.appconfigdata
```

Wenn Sie privates DNS für den Schnittstellenendpunkt aktivieren, können Sie API-Anforderungen an AWS AppConfig unter Verwendung des standardmäßigen regionalen DNS-Namens senden. 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 Standard-Endpunktrichtlinie ermöglicht vollen Zugriff auf AWS AppConfig über den Schnittstellenendpunkt. Um den Zugriff auf AWS AppConfig von Ihrer VPC aus zu steuern, 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 AWS AppConfig-Aktionen

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. Weitere Informationen zu Secrets Manager finden Sie unter [Was ist AWS Secrets Manager?](#) im AWS Secrets Manager -Benutzerhandbuch.

Einrichten der automatischen Drehung von Secrets-Manager-Secrets, die von bereitgestellt werden AWS AppConfig

Rotation ist der Prozess der regelmäßigen Aktualisierung eines in Secrets Manager gespeicherten Secrets. Wenn Sie ein Secret rotieren, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank oder im Service aktualisiert. Sie können die automatische Secret-Rotation in Secrets Manager konfigurieren, indem Sie eine -AWS LambdaFunktion verwenden, um das Secret und die Datenbank zu aktualisieren. Weitere Informationen finden Sie unter [Rotieren von AWS Secrets Manager Secrets](#) im AWS Secrets Manager -Benutzerhandbuch.

Um die Schlüsselrotation von Secrets-Manager-Secrets zu aktivieren, die von bereitgestellt werden AWS AppConfig, aktualisieren Sie Ihre Lambda-Funktion für die Drehung und stellen Sie das rotierte Secret bereit.

Note

Stellen Sie Ihr AWS AppConfig Konfigurationsprofil bereit, nachdem Ihr Secret gedreht und vollständig auf die neue Version aktualisiert wurde. Sie können feststellen, ob das Secret gedreht wurde, weil sich der Status von von `AWSPENDING` zu `VersionStage` ändert `AWSCURRENT`. Der Abschluss der Secret-Rotation erfolgt innerhalb der `finish_secret` Secrets-Manager-Funktion Rotationsvorlagen.

Hier ist eine Beispielfunktion, die eine -AWS AppConfigBereitstellung startet, nachdem ein Secret gedreht 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 von AWS AppConfig

Die Überwachung ist wesentlich zur Wahrung der Zuverlässigkeit, Verfügbarkeit und Leistung von AWS AppConfig und Ihren anderen AWS-Lösungen. AWS bietet folgende Überwachungswerkzeuge, mit denen Sie AWS AppConfig beobachten, Missstände melden und ggf. automatisch Maßnahmen ergreifen können:

- AWS CloudTrail erfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS-Kontos erfolgten, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon-S3-Bucket. Sie können die Benutzer und Konten, die AWS aufgerufen haben, identifizieren, sowie die Quell-IP-Adresse, von der diese Aufrufe stammen, und den Zeitpunkt der Aufrufe ermitteln. Weitere Informationen finden Sie im [AWS CloudTrail-Benutzerhandbuch](#).
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von Amazon EC2-Instances und anderen Quellen aus überwachen CloudTrail, speichern und darauf zugreifen. - CloudWatch Protokolle können 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](#).

Themen

- [Protokollieren von AWS AppConfig-API-Aufrufen mithilfe von AWS CloudTrail](#)
- [Protokollieren von Metriken für Aufrufe auf AWS AppConfig Datenebene](#)

Protokollieren von AWS AppConfig-API-Aufrufen mithilfe von AWS CloudTrail

AWS AppConfig ist in integriert, einem ServiceAWS CloudTrail, der die Aktionen eines Benutzers, einer Rolle oder eines -AWSServices in aufzeichnetAWS AppConfig. CloudTrail erfasst alle API-Aufrufe für AWS AppConfig als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS AppConfig-Konsole und Code-Aufrufe der AWS AppConfig-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3-Bucket aktivieren, einschließlich Ereignissen für AWS AppConfig. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem die neuesten Ereignisse in der CloudTrail Konsole unter Ereignisverlauf anzeigen. Anhand der von CloudTrailgesammelten Informationen können Sie die an

gestellte Anfrage AWS AppConfig, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrail finden Sie im [AWS CloudTrail -Benutzerhandbuch](#).

AWS AppConfig Informationen in CloudTrail

CloudTrail wird beim Erstellen des Kontos AWS-Konto auf Ihrem aktiviert. Wenn eine Aktivität in auftritt AWS AppConfig, wird diese Aktivität in einem - CloudTrail Ereignis zusammen mit anderen -AWS Serviceereignissen im Ereignisverlauf aufgezeichnet. Sie können in Ihrem AWS-Konto die neusten Ereignisse anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter [Anzeigen von Ereignissen mit dem CloudTrail Ereignisverlauf](#).

Zur kontinuierlichen Aufzeichnung von Ereignissen in Ihrem AWS-Konto, einschließlich Ereignissen für AWS AppConfig, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien an einen Amazon S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon-S3-Bucket bereit. Darüber hinaus können Sie andere -AWS Services konfigurieren, um die in den CloudTrail Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- [Übersicht zum Erstellen eines Trails](#)
- [CloudTrail Von unterstützte Services und Integrationen](#)
- [Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail](#)
- [Empfangen von CloudTrail Protokolldateien aus mehreren Regionen](#) und [Empfangen von CloudTrail Protokolldateien aus mehreren Konten](#)

Alle AWS AppConfig Aktionen werden von protokolliert CloudTrail und sind in der [AWS AppConfig API-Referenz](#) zu dokumentiert. Aufrufe der ListApplications Aktionen CreateApplication, GetApplication und erzeugen beispielsweise Einträge in den CloudTrail Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Benutzeranmeldeinformationen ausgeführt wurde.

- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter [CloudTrail -Element userIdentity](#).

AWS AppConfig -Datenereignisse in CloudTrail

[Datenereignisse](#) liefern Informationen über die Ressourcenoperationen, die für oder in einer Ressource ausgeführt werden (z. B. das Abrufen der neuesten bereitgestellten Konfiguration durch Aufrufen von `GetLatestConfiguration`). Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Standardmäßig protokolliert CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zu CloudTrail Preisen finden Sie unter [-AWS CloudTrailPreise](#).

Sie können Datenereignisse für die AWS AppConfig Ressourcentypen mithilfe der CloudTrail Konsole, der AWS CLI oder der API CloudTrail -Operationen protokollieren. Die [Tabelle](#) in diesem Abschnitt zeigt die Ressourcentypen, die für verfügbar sind AWS AppConfig.

- Um Datenereignisse mit 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 in der Liste Datenereignistyp die Option aus 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 `resources.ARN` Feldern `readOnly`, `eventName` und zu filtern.
 4. Geben Sie für Selektornamen `AppConfigDataEvents`. Informationen zum Aktivieren von Amazon CloudWatch Logs für Ihren Datenereignis-Trail finden Sie unter [Protokollieren von Metriken für Aufrufe auf AWS AppConfig Datenebene](#).
- Um Datenereignisse mit der zu protokollieren AWS CLI, konfigurieren Sie den `--advanced-event-selector` Parameter so, dass das `-eventCategory` Feld auf `Data` und das

`-resources.type` Feld auf den Ressourcentypwert gesetzt wird (siehe [Tabelle](#)). Sie können Bedingungen hinzufügen, um nach den Werten der `resources.ARN` Felder `readOnlyEventName`, und zu filtern.

- Um einen Trail für die Protokollierung von Datenereignissen zu konfigurieren, führen Sie den [put-event-selectors](#) Befehl aus. Weitere Informationen finden Sie unter [Protokollieren von Datenereignissen für Trails mit der AWS CLI](#).
- Um einen Ereignisdatenspeicher für die Protokollierung von Datenereignissen zu konfigurieren, führen Sie den [create-event-data-store](#) Befehl aus, um einen neuen Ereignisdatenspeicher für die Protokollierung von Datenereignissen zu erstellen, oder führen Sie den [update-event-data-store](#) Befehl aus, um einen vorhandenen Ereignisdatenspeicher zu aktualisieren. Weitere Informationen finden Sie unter [Protokollieren von Datenereignissen für Ereignisdatenspeicher mit der 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 in der CloudTrail Konsole ausgewählt werden kann. Die Spalte `resources.type value` zeigt den `resources.type` Wert an, den Sie bei der Konfiguration erweiterter Ereignisselektoren mit der AWS CLI oder CloudTrail APIs angeben würden. Die Spalte Daten-APIs, die in protokolliert CloudTrail wurden, zeigt die API-Aufrufe an, CloudTrail die für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	<code>resources.type</code> -Wert	Daten-APIs, die bei CloudTrail* protokolliert wurden
AWS AppConfig	<code>AWS::AppConfig::Configuration</code>	<ul style="list-style-type: none"> • GetLatestConfiguration • StartConfigurationSession

* Sie können erweiterte Ereignisselektoren konfigurieren `eventName`, um nach den `resources.ARN` Feldern `readOnly`, und zu filtern, um nur die Ereignisse zu protokollieren, die für Sie wichtig sind. Weitere Informationen zu diesen Feldern finden Sie unter [AdvancedFieldSelector](#).

AWS AppConfig -Verwaltungsereignisse in CloudTrail

[Verwaltungsereignisse](#) liefern Informationen zu Verwaltungsvorgängen, die für Ressourcen im AWS-Konto ausgeführt wurden. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. Standardmäßig CloudTrail protokolliert Verwaltungsereignisse.

AWS AppConfig protokolliert alle Operationen auf AWS AppConfig Steuerebene als Verwaltungsereignisse. Eine Liste der Operationen auf AWS AppConfig Steuerebene, die in AWS AppConfig protokolliert CloudTrail, finden Sie in der API [AWS AppConfig-Referenz zu](#) .

Grundlagen zu AWS AppConfig-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Bereitstellung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen - CloudTrail Protokolleintrag, der die [StartConfigurationSession](#) Aktion 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": "mexamplee0",
    "configurationProfileIdentifier": "3eexamplee1"
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::AppConfig::Configuration",
      "ARN": "arn:aws:appconfig:us-east-1:123456789012:application/rrfexample/
environment/mexamplee0/configuration/3eexamplee1"
    }
  ],
  "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"
  }
}
```

Protokollieren von Metriken für Aufrufe auf AWS AppConfig Datenebene

Wenn Sie für AWS CloudTrail die Protokollierung von AWS AppConfig Datenereignissen konfiguriert haben, können Sie Amazon CloudWatch Logs aktivieren, um Metriken für Aufrufe an die AWS AppConfig Datenebene zu protokollieren. Anschließend können Sie Protokolldaten in - CloudWatch Protokollen suchen und filtern, indem Sie einen oder mehrere Metrikfilter erstellen. Metrikfilter definieren die Begriffe und Muster, nach denen in Protokolldaten gesucht werden soll, wenn sie an CloudWatch Logs gesendet werden. CloudWatch Logs verwendet Metrikfilter, 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. Im folgenden Verfahren wird beschrieben, wie Sie die Metrikprotokollierung für einen vorhandenen AWS AppConfig Trail in aktivieren CloudTrail. Informationen zum Aktivieren der CloudTrail Protokollierung für AWS AppConfig Datenplanaufrufe finden Sie unter [AWS AppConfig -Datenereignisse in CloudTrail](#).

Gehen Sie wie folgt vor, um CloudWatch Logs zu ermöglichen, Metriken für Aufrufe an die AWS AppConfig Datenebene zu protokollieren.

So aktivieren Sie CloudWatch Protokolle, um Metriken für Aufrufe an die AWS AppConfig Datenebene zu protokollieren

1. Öffnen Sie die - CloudTrail Konsole unter <https://console.aws.amazon.com/cloudtrail/>.
2. Wählen Sie im Dashboard Ihren AWS AppConfig Trail aus.
3. Wählen Sie im Abschnitt CloudWatch Protokolle die Option Bearbeiten aus.
4. Wählen Sie Aktiviert.
5. Behalten Sie für Protokollgruppenname entweder den Standardnamen bei oder geben Sie einen Namen ein. Notieren Sie den Namen. Sie wählen die Protokollgruppe später in der CloudWatch Logs-Konsole aus.
6. Geben Sie in Role name (Name der Rolle) einen Namen ein.
7. Wählen Sie Änderungen speichern aus.

Gehen Sie wie folgt vor, um eine Metrik und einen Metrikfilter für AWS AppConfig in - CloudWatch Protokollen zu erstellen. Das Verfahren beschreibt, wie Sie einen Metrikfilter für Aufrufe von operation und (optional) Aufrufe von operation und erstellenAmazon Resource Name (ARN).

So erstellen Sie eine Metrik und einen Metrikfilter für AWS AppConfig in - CloudWatch Protokollen

1. Öffnen Sie die - CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Log groups (Protokollgruppen) aus.
3. Aktivieren Sie das Kontrollkästchen neben der AWS AppConfig Protokollgruppe.
4. Wählen Sie Aktionen und dann Metrikfilter erstellen.
5. Geben Sie unter Filtername einen Namen ein.
6. Geben Sie für Filtermuster Folgendes ein:

```
{ $.eventSource = "appconfig.amazonaws.com" }
```

7. (Optional) Wählen Sie im Abschnitt Testmuster Ihre Protokollgruppe aus der Liste Protokolldaten zum Testen auswählen aus. Wenn keine Aufrufe protokolliert CloudTrail hat, können Sie diesen Schritt überspringen.
8. Wählen Sie Weiter aus.
9. Geben Sie für Metrik-Namespace ein **AWS AppConfig**.
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 für Dimensionsname ein **operation**.
14. Geben Sie für Dimensionswert ein **\$.eventName**.

(Optional) Sie können eine zweite Dimension eingeben, die den Amazon-Ressourcennamen (ARN) enthält, der den Aufruf durchführt. Um eine zweite Dimension hinzuzufügen, geben Sie für Dimensionsname ein **resource**. Geben Sie für Dimensionswert ein **\$.resources[0].ARN**.

Wählen Sie Weiter aus.

15. Überprüfen Sie die Details des Filters und Metrikfilter erstellen .

(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 für Filtermuster Folgendes ein:

```
{ $.errorCode = "codename" }
```


Beispiel

```
{ $.errorCode = "AccessDenied" }
```

3. Geben Sie für Metrik-Namespace ein **AWS AppConfig**.
4. Geben Sie bei Metric name den Metriknamen **Errors** ein.

5. Geben Sie für Metric value (Metrikwert) **1** ein.
6. Geben Sie für Standardwert eine Null (0) ein.
7. Überspringen Sie Einheit , Dimensionen und Alarme .

Nachdem API-Aufrufe CloudTrail protokolliert hat, können Sie Metriken in anzeigen CloudWatch. Weitere Informationen finden Sie unter [Anzeigen Ihrer Metriken und Protokolle in der Konsole](#) im Amazon- CloudWatch Benutzerhandbuch. Informationen zum Suchen einer von Ihnen erstellten Metrik finden Sie unter [Suchen nach verfügbaren Metriken](#).

 Note

Wenn Sie die Fehlermetrik ohne Dimension einrichten, wie hier beschrieben, können Sie diese Metriken auf der Seite Metriken ohne Dimension anzeigen.

Erstellen eines Alarms für eine CloudWatch Metrik

Nachdem Sie Metriken erstellt haben, können Sie Metrikalarme in erstellen CloudWatch. Sie können beispielsweise einen Alarm für die AWS AppConfig Aufrufmetrik erstellen, die Sie im vorherigen Verfahren erstellt haben. Insbesondere können Sie einen Alarm für Aufrufe an die 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 basierend auf einem statischen Schwellenwert](#) im Amazon- CloudWatch Benutzerhandbuch. Informationen zu Standardlimits für Aufrufe an die AWS AppConfig Datenebene finden Sie unter [Standardlimits auf Datenebene](#) im Allgemeine Amazon Web Services-Referenz.

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
AWS AppConfig Beispiele für benutzerdefinierte Erweiterungen	<p>Das Thema Walkthrough: Erstellen von AWS AppConfig benutzerdefinierten Erweiterungen enthält jetzt Links zu den folgenden Beispielerweiterungen auf GitHub:</p> <ul style="list-style-type: none">• Beispielerweiterung, die Bereitstellungen mit einem blocked day Mauskalender mithilfe von Systems Manager Change Calendar verhindert• Beispielerweiterung, die verhindert, dass Secrets mithilfe von git-secrets in Konfigurationsdaten gelangen• Beispielerweiterung, die verhindert, dass persönlich identifizierbare Informationen (PII) mithilfe von Amazon Comprehend in Konfigurationsdaten gelangen	28. Februar 2024

[Neues Thema: Protokollieren von AWS AppConfig API-Aufrufen mit AWS CloudTrail](#)

AWS AppConfig ist integriert, einem Service AWS CloudTrail, der die Aktionen eines Benutzers, einer Rolle oder eines - AWS Services in aufzeichnet AWS AppConfig . CloudTrail erfasst alle API-Aufrufe für AWS AppConfig als Ereignisse. Dieses neue Thema enthält AWS AppConfig-spezifische Inhalte, anstatt mit dem entsprechenden Inhalt im AWS Systems Manager -Benutzerhandbuch zu verknüpfen. Weitere Informationen finden Sie unter [Protokollieren von AWS AppConfig API-Aufrufen mit AWS CloudTrail](#).

18. Januar 2024

[AWS AppConfig unterstützt jetzt AWS PrivateLink](#)

Sie können verwenden AWS PrivateLink , um eine private Verbindung zwischen Ihrer VPC und herzustellen AWS AppConfig. Sie können auf zugreifen AWS AppConfig , als wäre es in Ihrer VPC, ohne die Verwendung eines Internet-Gateways, NAT-Gerät s, einer VPN-Verbindung oder einer - AWS Direct Connect Verbindung. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um auf zuzugreifen AWS AppConfig. Weitere Informationen finden Sie unter [Zugriff AWS AppConfig über einen Schnittstellenendpunkt \(AWS PrivateLink\)](#).

6. Dezember 2023


[Zusätzliche Funktionen zum Abrufen von AWS AppConfig Kundendienstmitarbeitern und ein neuer lokaler Entwicklungsmodus](#)

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

1. Dezember 2023

[Zusätzliche Abruffunktionen](#)

- **Abruf mehrerer Konten:**
Verwenden Sie den - AWS AppConfig Agenten von einem primären oder -Abruf AWS-Konto , um Konfigurationsdaten von mehreren Anbieterkonten abzurufen.
- **Schreiben einer Konfigurationsskopie auf die Festplatte:** Verwenden Sie - AWS AppConfig Agent, um Konfigurationsdaten auf die Festplatte zu schreiben. Diese Funktion ermöglicht es Kunden mit Anwendungen, die Konfigurationsdateien von der Festplatte lesen, in zu integrieren AWS AppConfig.

 Note

Die Schreibkonfiguration auf die Festplatte ist nicht als Konfigurationssicherungsfunktion konzipiert. AWS

AppConfig Agent liest nicht von den auf die Festplatte kopierten Konfigurationsdateien. Wenn Sie Konfigurationen auf der Festplatte sichern möchten, lesen Sie die `PRELOAD_BACKUP` Umgebungsvariablen `BACKUP_DIRECTORY` und für [Verwenden von - AWS AppConfig Agent mit Amazon EC2](#) oder [Verwenden von - AWS AppConfig Agent mit Amazon ECS und Amazon EKS](#).

Lokaler Entwicklungsmodus

AWS AppConfig Agent unterstützt einen lokalen Entwicklungsmodus. Wenn Sie den lokalen Entwicklungsmodus aktivieren, liest der Agent Konfigurationsdaten aus einem bestimmten Verzeichnis auf der Festplatte. Es 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 mit bereitgestellten AWS AppConfig.
- Testen Sie verschiedene Konfigurationsoptionen für ein neues Feature, bevor Sie Änderungen an Ihrem Code-Repository übergeben.
- Testen Sie verschiedene Konfigurationsszenarien, um sicherzustellen, dass sie wie erwartet funktionieren.

[Neues Thema zu Codebeispielen](#)

In diesem Handbuch wurde ein neues Thema zu [Codebeispielen](#) hinzugefügt. Das Thema enthält Beispiele in Java, Python und JavaScript für die programmgesteuerte Ausführung von sechs AWS AppConfig häufigen Aktionen.

17. November 2023

[Das Inhaltsverzeichnis wurde überarbeitet, um den Workflow besser wiederzugeben AWS AppConfig](#)

Die Inhalte in diesem Benutzerhandbuch sind jetzt unter den Überschriften Erstellen, Bereitstellen, Abrufen und Erweitern von Workflows gruppiert. Diese Organisation spiegelt den Workflow für die Verwendung von besser wider AWS AppConfig und versucht, Inhalte leichter auffindbar zu machen.

7. November 2023

[Nutzlastreferenz hinzugefügt](#)

Das Thema [Erstellen einer Lambda-Funktion für ein AWS AppConfig benutzerdefiniertes Erweiterungsthema](#) enthält jetzt eine Nutzlastreferenz für Anforderungen und Antworten.

7. November 2023

[Neue AWS vordefinierte Bereitstellungsstrategie](#)

AWS AppConfig bietet und empfiehlt jetzt die AppConfig `.Linear20PercentEvery6Minutes` vordefinierte Bereitstellungsstrategie. Weitere Informationen finden Sie unter [Vordefinierte Bereitstellungsstrategien](#).

11. August 2023

[AWS AppConfig -Integration mit Amazon EC2](#)

Sie können AWS AppConfig mit Agent in Anwendungen integrieren, die auf Ihren Amazon Elastic Compute Cloud (Amazon EC2) Linux-Instances ausgeführt werden AWS AppConfig . 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 ein Feature-Flag](#)

AWS CloudFormation unterstützt jetzt die [AWS::AppConfig::ExtensionAssociation](#) Ressourcen [AWS::AppConfig::Extension](#) und , um Ihnen den Einstieg in AWS AppConfig Erweiterungen zu erleichtern.

12. April 2023

Die - [AWS::AppConfig::ConfigurationProfile](#) und [-AWS::AppConfig::HostedConfigurationVersion](#) Ressourcen enthalten jetzt ein Beispiel für die AWS AppConfig Erstellung eines Feature-Flag-Konfigurationsprofils im gehosteten Konfigurationsspeicher.

[AWS AppConfig -Integration mit AWS Secrets Manager](#)

2. Februar 2023

AWS AppConfig lässt sich integrieren mit AWS Secrets Manager. Secrets Manager hilft Ihnen, Anmeldeinformationen für Ihre Datenbanken und andere Services 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, den Zugriff auf Ihre IT-Ressourcen und -Daten zu schützen, indem es Ihnen ermöglicht, den Zugriff auf Ihre Secrets zu rotieren und zu verwalten.

Wenn Sie ein Freiform-Konfigurationsprofil erstellen, können Sie Secrets Manager als Quelle Ihrer Konfigurationsdaten wählen. Sie müssen Secrets Manager einbinden und ein Secret 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
finden Sie unter [Erstellen
eines Freiform-Konfigura
tionsprofils](#).

[AWS AppConfig -Integration mit Amazon ECS und Amazon EKS](#)

2. Dezember 2022

Sie können AWS AppConfig mit Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) integrieren, indem Sie den AWS AppConfig Agenten verwenden. Der Agent fungiert als Sidecar-Container, der zusammen mit Ihren Amazon-ECS- und Amazon-EKS-Containeranwendungen ausgeführt wird. Der Agent verbessert die Verarbeitung und Verwaltung von containerisierten Anwendungen auf folgende Weise:

- Der Agent ruft AWS AppConfig in Ihrem Namen auf, 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, um Konfigurationsdaten zu verwalten, ruft Konfigurationsdaten in Millisekunden ab und ist nicht von Netzwerkproblemen

betroffen, die Aufrufe für solche Daten unterbrechen können.

- Der Agent bietet eine native Erfahrung zum Abrufen und Auflösen von AWS AppConfig Feature-Flags.
- Standardmäßig bietet der Agent bewährte Methoden für Caching-Strategien, Abfrageintervalle und die lokale Verfügbarkeit von Konfigurationsdaten und verfolgt gleichzeitig die Konfigurations-Token, die für nachfolgende Serviceaufrufe benötigt werden.
- Während der Ausführung im Hintergrund fragt der Agent die AWS AppConfig Datenebene regelmäßig nach Konfigurationsdateaktualisierungen ab. Ihre containerisierte Anwendung kann die Daten abrufen, indem sie eine Verbindung zu localhost auf Port 2772 (einem anpassbaren Standardportwert) 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](#)

13. September 2022

Sie können Amazon CloudWatch Evidently verwenden, um neue Features sicher zu validieren, indem Sie sie einem bestimmten Prozentsatz Ihrer Benutzer bereitstellen, während Sie die Funktion einführen. 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, Benutzeritzungen lokal Varianten zuzuweisen, anstatt die [EvaluateFeature](#) Operation aufzurufen. Eine lokale Sitzung mindert die Latenz- und Verfügbarkeitsrisiken, die mit einem API-Aufruf verbunden sind. Informationen zum Konfigurieren und

Verwenden der Erweiterung finden Sie unter [Durchführen von Starts und A/B-Experimenten mit CloudWatch Evidently](#) im Amazon-CloudWatch Benutzerhandbuch.

[Veraltung der GetConfiguration API-Aktion](#)

Am 18. November 2021 AWS AppConfig veröffentlichte einen neuen Datenebenen-Service. Dieser Service ersetzt den vorherigen Prozess zum Abrufen von Konfigurationsdaten mithilfe der GetConfiguration API-Aktion . Der Datenebenen-Service verwendet zwei neue API-Aktionen, [StartConfigurationSession](#) und [GetLatestConfiguration](#). Der Datenebenen-Service verwendet auch [neue Endpunkte](#).

13. September 2022

Weitere Informationen finden Sie unter [Informationen zum AWS AppConfig Datenebenen-Service](#) .

[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 verschiedene Amazon-Ressourcennamen (ARNs). Weitere Informationen finden Sie in den [AWS AppConfig Versionshinweisen zur Agent-Lambda-Erweiterung](#) .

23. August 2022

[Starten von AWS AppConfig
Erweiterungen](#)

Eine Erweiterung erweitert Ihre Fähigkeit, Logik oder Verhalten an verschiedenen Stellen während der AWS AppConfig Erstellung oder Bereitstellung einer Konfiguration einzubringen. Sie können AWS von autorisierte Erweiterungen verwenden oder Ihre eigenen erstellen. Weitere Informationen finden Sie unter [Arbeiten mit AWS AppConfig Erweiterungen](#).

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 verschiedene Amazon-Ressourcennamen (ARNs). Weitere Informationen finden Sie unter [Verfügbare Versionen der AWS AppConfig Lambda-Erweiterung](#) .

3. Mai 2022

[AWS AppConfig -Integration mit Atlassian JSpeed](#)

7. April 2022

Durch die Integration mit Atlassian JCCP können Probleme in der Atlassian-Konsole AWS AppConfig erstellt und aktualisiert werden, wenn Sie Änderungen an einem [Feature-Flag](#) in Ihrem AWS-Konto für das angegebene vornehmen AWS-Region. Jedes Jura-Problem enthält den Flag-Namen, die Anwendungs-ID, die Konfigurationsprofil-ID und die Flag-Werte. Nachdem Sie Ihre Flag-Änderungen aktualisiert, gespeichert und bereitgestellt haben, aktualisiert JCCP die vorhandenen Probleme mit den Details der Änderung. Weitere Informationen finden Sie unter [-AWS AppConfig Integration mit Atlassian Jura](#).

[Allgemeine Verfügbarkeit von Feature-Flags und Lambda-Erweiterungsunterstützung für ARM64-Prozessoren \(Graviton 2\)](#)

Mit AWS AppConfig Feature-Flags können Sie ein neues Feature entwickeln und es in der Produktion bereitstellen, während Sie das Feature vor Benutzern verbergen. Sie beginnen damit, das Flag AWS AppConfig als Konfigurationsdaten zu hinzuzufügen. Sobald die Funktion zur Veröffentlichung bereit ist, können Sie die Flag-Konfigurationsdaten aktualisieren, ohne Code bereitzustellen. Diese Funktion verbessert die Sicherheit Ihrer Entwicklungsumgebung, 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 die Option, ein Flag als kurzfristiges Flag zu kennzeichnen. Sie können die Liste der Flags nach kurzfristigen Flags filtern und sortieren.
- Kunden, die Feature-Flags in verwenden AWS

15. März 2022

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-Prozessoren (Graviton 2) 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 die GetLatestConfiguration APIs StartConfigurationSession und verwenden. Weitere Informationen zu diesen APIs und deren Verwendung finden Sie unter [Abrufen der Konfiguration](#).

28. Januar 2022

[Neuer Regions-ARN für die AWS AppConfig Lambda-Erweiterung](#)

AWS AppConfig Die Lambda-Erweiterung ist in der neuen Region Asien-Pazifik (Osaka) verfügbar. Der Amazon-Resourcenname (ARN) ist erforderlich, um ein Lambda in der Region zu erstellen. Weitere Informationen zum ARN der Region Asien-Pazifik (Osaka) finden Sie unter [Hinzufügen der AWS AppConfig Lambda-Erweiterung](#).

4. März 2021

[AWS AppConfig Lambda-Erweiterung](#)

Wenn Sie AWS AppConfig zum Verwalten von Konfigurationen für eine Lambda-Funktion verwenden, empfehlen wir Ihnen, die AWS AppConfig Lambda-Erweiterung hinzuzufügen. Diese Erweiterung enthält bewährte Methoden, die die Verwendung von vereinfachten AWS AppConfig und gleichzeitig die Kosten senken. Geringere Kosten resultieren aus weniger API-Aufrufen an den AWS AppConfig Service und separat auch aus geringeren Kosten aufgrund kürzerer Verarbeitungszeiten von 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 zum Einrichten von enthält AWS AppConfig. Weitere Informationen finden Sie unter [Einrichten von AWS AppConfig](#).

30. September 2020

[Befehlszeilenverfahren hinzugefügt](#)

Die Verfahren in diesem Benutzerhandbuch enthalten jetzt Befehlszeilenschritte für die AWS Command Line Interface (AWS CLI) und Tools for Windows PowerShell. Weitere Informationen finden Sie unter [Arbeiten mit AWS AppConfig](#).

30. September 2020

[Start des AWS AppConfig Benutzerhandbuchs](#)

Verwenden Sie AWS AppConfig, eine Funktion von 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 AWS AppConfig mit Anwendungen verwenden, die auf EC2-Instances, AWS Lambda, Containern, mobilen Anwendungen oder IoT-Geräten gehostet werden.

31. Juli 2020

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im [AWS-Glossar](#) in der AWS-Glossar-Referenz.

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.