AWS AppConfig

ユーザーガイド

AWS AppConfig: ユーザーガイド

Table of Contents

AWS AppConfig とは? 1
を使用したタスクの簡素化AWS AppConfig 1
AWS AppConfig ユースケース 1
AWS AppConfig を使用する利点 1
AWS AppConfig の使用を開始 2
How AWS AppConfig works 2
アプリケーションで動作するように AWS AppConfig を設定します。 3
アプリケーションコードを有効にして、設定データを確認して取得する 4
新規または更新された構成をデプロイする 4
AWS AppConfig の料金 5
AWS AppConfig のクォータ 5
開始方法 6
AWS コマンドラインツールをインストールまたはアップグレードする 6
AWS AppConfig のアクセス許可の設定 8
(オプション) に基づいてロールバック用のアクセス許可の設定 CloudWatch アラーム 9
ステップ 1: に基づいてロールバック用のアクセス権限ポリシーを作成します。 CloudWatch アラーム 9
ステップ 2: に基づいてロールバック用の IAM ロールを作成します。 CloudWatch アラーム 10
ステップ 3: 信頼関係を追加する 10
AWS AppConfigの使用 12
ステップ 1: の作成AWS AppConfig応用 12
AWS AppConfig アプリケーションの作成 (コンソール) 12
AWS AppConfig アプリケーションの作成 (コマンドライン) 13
ステップ 2: 環境を作成する 14
AWS AppConfig 環境の作成 (コンソール) 14
AWS AppConfig 環境の作成 (コマンドライン) 15
ステップ 3: 設定プロファイルと機能フラグの作成 16
設定例 17
設定プロファイルの IAM ロールについて 19
Amazon S3 に保存された設定について 20
バリデータについて 22
機能フラグ設定プロファイルの作成 24
フリーフォーム構成プロファイルの作成 34
ステップ 4: デプロイ戦略の作成 39
定義済みのデプロイ戦略 41
デプロイ戦略の作成 41
ステップ 5: 設定のデプロイ 44
設定をデプロイする (コンソール) 45
設定をデプロイする (コマンドライン) 45
ステップ 6: 設定の取得 48
設定例の取得 48
AWS AppConfig と統合するサービス 50
AWS Lambda 拡張機能 50
仕組み 50
始める前に 52
ランタイムのサポート 52
AWS AppConfig Lambda 拡張機能を追加する 52
設定:AWS AppConfigLambda 拡張 53
1 つ以上のフラグを取得する 55
の使用可能なバージョンAWS AppConfigLambda 拡張 55
コンテナイメージを使用して AWS AppConfig Lambda 拡張機能を追加する 63
OpenAPI と統合する 66
アトラシアン Jira 67
許可を設定する 68

統合の設定 70
統合の削除 71
AWS CodePipeline 71
統合の仕組み 71
セキュリティ 73
モニタリング 74
ドキュメント履歴 75
AWS 用語集 78
..... lxxix

AWS AppConfig とは?

を使用するAWS AppConfigの「」の機能であるAWS Systems Managerを使用して、アプリケーション設定を作成、管理し、迅速にデプロイします。設定は、アプリケーションの動作に影響する設定のコレクションです。AWS AppConfig は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、AWS Lambda、コンテナ、モバイルアプリケーション、または IoT デバイスでホストされるアプリケーションで使用できます。を使用して管理できる設定のタイプの例を表示するにはAWS AppConfig「」を参照してください。設定例 (p. 17)。

を使用したタスクの簡素化AWS AppConfig

AWS AppConfigは、次のタスクを簡素化するのに役立ちます。

- 設定する

  Amazon Simple Storage Service (Amazon S3) から設定をソースにAWS AppConfigホストされた構成、パラメータストア、Systems Manager のドキュメントストア。を使用するAWS CodePipelineBitbucket Pipelines から設定をソースするための統合、GitHub, およびAWS CodeCommit。
- 検証する

  アプリケーション設定のデプロイ中、単純なタイプミスにより予期しない停止が起こりうるが、を使用して本番システムでのエラーを防止AWS AppConfigバリデータ。AWS AppConfigバリデーターは、JSON スキーマを使用した構文チェック、またはAWS Lambda機能を使用して、構成が意図したとおりに展開されるようにします。設定データが有効である場合にのみ、設定のデプロイが進行します。
- デプロイと監視

  デプロイ条件とレートコントロールを定義して、ターゲットが新しい構成を取得する方法を決定します。を使用するAWS AppConfigデプロイメントストラテジーは、デプロイ速度、デプロイ時間、およびベイク時間を設定します。を使用して各デプロイメントを監視して、プロアクティブにエラーを検出します。AWS AppConfigAmazon との統合CloudWatch。もしAWS AppConfigエラーが発生した場合、システムはデプロイをロールバックして、アプリケーションユーザーへの影響を最小限に抑えます。

AWS AppConfig ユースケース

AWS AppConfigは、以下のユースケースにお役に立ちます。

- アプリケーションチューニング：本番トラフィックでテストできる変更を、アプリケーションに対して慎重に導入します。
- 機能切り替え：製品の発売や発表など、適切なタイミングでのデプロイが必要となる新機能をオンにします。
- 許可リスト— プレミアム加入者が有料コンテンツにアクセスできるようにします。
- 運用上の問題：依存性やその他の外部要因がシステムに影響を与える場合に、アプリケーションに対するストレスを軽減します。

AWS AppConfig を使用する利点

AWS AppConfigには、組織に次のような利点があります。

- 設定変更のエラーを減らす

  AWS AppConfig を使用すると設定を検証するルールを作成できるため、アプリケーションのダウンタイムを短縮できます。有効でない設定はデプロイできません。AWS AppConfigには、設定を検証するための次の 2 つのオプションがあります。
  - 構文検証には、JSON スキーマを使用することができます。AWS AppConfig は JSON スキーマを使用して設定を検証し、設定の変更がアプリケーションの要件に準拠していることを確認します。
  - セマンティック検証では、設定を実行する AWS Lambda 関数をデプロイする前に呼び出すことができます。
- 複数のターゲットにわたって迅速に変更をデプロイする

  AWS AppConfigでは、設定変更を一元的な場所からデプロイすることで、大規模なアプリケーションの管理が簡素化されます。AWS AppConfigは、Systems Manager パラメータストア、Systems Manager (SSM) ドキュメント、および Amazon S3 に保存された設定をサポートします。AWS AppConfig は、EC2 インスタンス、AWS Lambda、コンテナ、モバイルアプリケーション、または IoT デバイスでホストされているアプリケーションで使用できます。

  ターゲットは、Systems Manager SSM エージェントまたはAWS Identity and Access Management他の Systems Manager 機能で必要な (IAM) インスタンスプロファイル。つまり、AWS AppConfig はアンマネージド型インスタンスで動作します。
- 中断することなくアプリケーションを更新する

  AWS AppConfig は、重いビルドプロセスを実行したりターゲットをサービスから外したりせずに、実行時にターゲットに設定変更をデプロイします。
- アプリケーション全体で変更のデプロイを制御する

  設定変更をターゲットにデプロイするときは、AWS AppConfigでは、展開戦略を使用してリスクを最小限に抑えることができます。デプロイ戦略のレート制御を使用して、アプリケーションターゲットが構成変更を取得する速度を決定できます。

AWS AppConfig の使用を開始

以下のリソースは、AWS AppConfig を直接使用する場合に役立ちます。

AWS AppConfig概要 (ビデオ)

もっと見るAWSの動画アマゾンウェブサービスYouTubeチャネル。

詳細については、以下のブログを参照してください。AWS AppConfigその機能は次のとおりです。

- についてAWS AppConfig配備— アプリケーション構成設定の安全な展開についてAWS AppConfig。
- を使用した機能のリリースの自動化AWS AppConfig配備— を使用して機能のリリースを自動化する方法について説明します。AWS AppConfigと の統合AWS CodePipeline。
- サーバーレスワークロードへのアプリケーション構成のデプロイ— を使用する方法について説明します。AWS AppConfigアプリケーション設定をサーバーレスワークロードにデプロイする Lambda 拡張機能。

How AWS AppConfig works

大まかに言って、を操作するプロセスは 3 つあります。AWS AppConfig:

1. 設定 (p. 3) AWS AppConfigを使用して、アプリケーションで動作します。

2. の有効化 (p. 4)定期的にを確認して設定データを取得するためのアプリケーションコードAWS AppConfig。
3. デプロイ (p. 4)新規または更新された設定。

以降のセクションでは、各ステップについて説明します。

アプリケーションで動作するように AWS AppConfig を設定します。

を設定するにはAWS AppConfigアプリケーションで動作するには、次の表で説明する 3 つのタイプのリソースを設定します。

リソース	詳細
Application	EclipseAWS AppConfigでは、アプリケーションは単にフォルダのような組織構造です。この組織構成は、実行可能コードの単位との関係を持っています。たとえば、「」というアプリケーションを作成できます。MyMobileユーザーによってインストールされたモバイルアプリケーションの構成データを整理および管理するためのアプリ。
環境	アプリケーションごとに、1 つ以上の環境を定義します。環境は、`Beta` または `Production` 環境内のアプリケーションなど、AWS AppConfig アプリケーションの論理的なデプロイメントグループです。アプリケーションの `Web`、`Mobile`、および `Back-end` といったコンポーネントを含む、アプリケーションのサブコンポーネントの環境を定義することもできます。Amazon を設定できます。CloudWatch各環境のアラーム。システムは、設定のデプロイ中にアラームをモニタリングします。アラームがトリガーされると、システムは設定をロールバックします。
設定プロファイル	設定プロファイルにより AWS AppConfig は保存場所の設定にアクセスできます。設定を保存できる形式と場所は次のとおりです。 • AWS AppConfig でホストされた設定ストア内の YAML、JSON、またはテキストドキュメント • Amazon S3 バケット内のオブジェクト • Systems Manager ドキュメントストア内のドキュメント • パラメータストア内のパラメータ 設定プロファイルにオプションのバリデータを含めて、設定データが構文的にもセマンティック的にも正しいことを確認することもできます。AWS AppConfig は、デプロイの開始時にバリデータを使用してチェックを実行します。エラーが検出されると、設定のターゲットに変更が加えられる前にデプロイが停止します。

アプリケーションコードを有効にして、設定データを確認して取得する

アプリケーションは、まず設定セッションを確立して、設定データを取得します。AWS AppConfigデータ`StartConfigurationSession`API アクション。その後、セッションのクライアントは定期的にを呼び出します`GetLatestConfiguration`をクリックして、利用可能な最新のデータを確認して取得します。

呼び出し時のエラー`StartConfigurationSession`では、コードは次の情報を送信します。

- の識別子 (ID または名前)AWS AppConfigセッションが追跡するアプリケーション、環境、設定プロファイル。
- (オプション) セッションのクライアントがを呼び出してから待機する必要がある最短時間`GetLatestConfiguration`。

それに応えて、AWS AppConfigにはが含まれています。`InitialConfigurationToken`セッションのクライアントに与えられ、初めて呼び出すときに使用されます。`GetLatestConfiguration`そのセッションのために。

呼び出し時のエラー`GetLatestConfiguration`の場合、クライアントコードは最新のものを送信します`ConfigurationToken`その値が受け取り、それに応答して受け取る値:

- `NextPollConfigurationToken`:`ConfigurationToken`次の呼び出しで使用する値`GetLatestConfiguration`。
- `NextPollIntervalInSeconds`: クライアントが次の呼び出しを行う前に待機する時間`GetLatestConfiguration`。この期間は、セッション中に異なる場合があるため、で送信される値の代わりに使用する必要があります。`StartConfigurationSession`を呼び出します。
- 構成:セッションを対象とした最新のデータ。クライアントにすでに最新バージョンの設定がある場合、これは空になることがあります。

詳細と例を表示する方法AWS CLIを使用して設定を取得する方法を示すコマンドAWS AppConfigデータ`StartConfigurationSession`そして`GetLatestConfiguration`API アクション、「」を参照してください。設定を受信する。

新規または更新された構成をデプロイする

AWS AppConfig を使用すると、アプリケーションのユースケースに最適な方法で設定をデプロイできます。変更を数秒でデプロイすることも、時間をかけてデプロイして変更の影響を評価することもできます。デプロイの制御に役立つ AWS AppConfig リソースを、デプロイ戦略と呼びます。展開戦略には、次の情報が含まれます。

- デプロイにかかる合計時間。(DeploymentDurationInMinutes).
- 各間隔でデプロイされた設定を取得するためのターゲットの割合。(GrowthFactor).
- デプロイが完了し、自動ロールバックの対象とならないとみなすまで、アラームのために AWS AppConfig が監視する時間の合計。(FinalBakeTimeIn分).

一般的なシナリオをカバーする組み込みのデプロイ戦略を使用することも、独自のデプロイ戦略を作成することもできます。デプロイ戦略を作成または選択したら、デプロイを開始します。デプロイを開始すると、StartDeploymentAPI アクション。呼び出しには、デプロイするアプリケーション、環境、設定プロファイル、および設定データバージョン (オプション) の ID が含まれます。この呼び出しには、使用するデプロイ戦略の ID も含まれます。ID は、設定データのデプロイ方法を決定します。

Note

についての情報AWS AppConfig言語固有の SDK については、「」を参照してください。AWS AppConfigSDK。

AWS AppConfig の料金

AWS AppConfig 使用には料金が発生します。詳細については、AWS Systems Manager の料金を参照してください。

AWS AppConfig のクォータ

に関する情報AWS AppConfigエンドポイントとサービスクォータと他の Systems Manager のクォータは、Amazon Web Services 全般リファレンス。

Note

AWS AppConfig 設定を格納するサービスのクォータについては、「設定ストアのクォータと制限について (p. 34)」を参照してください。

AWS AppConfig の使用開始

以下のトピックでは、AWS AppConfig の使用を開始するための重要なタスクについて説明します。

トピック

- AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)
- AWS AppConfig のアクセス許可の設定 (p. 8)
- (オプション) に基づいてロールバック用のアクセス許可の設定 CloudWatch アラーム (p. 9)

AWS コマンドラインツールをインストールまたはアップグレードする

このトピックは、AWS AppConfig (または他の AWS のサービス) を使用するためのプログラムによるアクセス権があり、ローカルマシンから AWS CLI コマンドまたは AWS Tools for Windows PowerShell コマンドを実行するユーザーを対象としています。

> Note
>
> プログラムによるアクセスと コンソールによるアクセス は、異なるアクセス許可であり、AWS アカウントの管理者よりユーザーアカウントに付与することができます。ユーザーには、1 つまたは両方のアクセスタイプを付与できます。詳細については、「AWS Systems Manager ユーザーガイド」の「Systems Manager の管理者以外の IAM ユーザーとグループを作成する」を参照してください。

AWS CLI の詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。AWS Tools for Windows PowerShell の詳細については、「AWS Tools for Windows PowerShell ユーザーガイド」を参照してください。

AWS CLI を使用して実行できるすべての AWS AppConfig コマンドについては、「AWS CLI コマンドリファレンス」の AWS AppConfig のセクションを参照してください。AWS Tools for PowerShell を使用して実行できるすべての AWS AppConfig コマンドについては、「AWS Tools for PowerShell コマンドレットリファレンス」の AWS AppConfig のセクションを参照してください。

AWS CLI

AWS CLI をインストールまたはアップグレードしてから設定するには

1. ローカルマシンに AWS CLI をインストールまたはアップグレードするには、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interface のインストールの手順」に従ってください。

   > Tip
   >
   > AWS CLI は、新しい機能で定期的に更新されます。すべての最新機能にアクセスできるように、CLI を定期的にアップグレード (再インストール) します。

2. AWS CLI を設定するには、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface の設定手順」を参照してください。

   このステップでは、組織の AWS 管理者から付与された認証情報を次の形式で指定します。

```
AWS Access Key ID: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Important

AWS CLI を設定するときに AWS リージョン を指定するよう求められます。「AWS 全般リファレンス」に記載されている、Systems Manager のサポート対象リージョンを 1 つ選択します。まず、必要に応じて、選択する必要のある AWS アカウントについて管理者に確認します。

アクセスキーの詳細については、「IAM ユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

3. インストールまたはアップグレードを確認するには、AWS CLI から次のコマンドを呼び出します。

```
aws appconfig help
```

成功すると、このコマンドは、使用できる AWS AppConfig コマンドのリストを表示します。

AWS Tools for PowerShell

AWS Tools for Windows PowerShell をインストールまたはアップグレードしてから設定するには

1. ローカルマシンで AWS Tools for PowerShell をインストールまたはアップグレードするには、「AWS Tools for Windows PowerShell ユーザーガイド」の「AWS Tools for Windows PowerShell または AWS Tools for PowerShell Core のセットアップ」の手順に従ってください。

   Tip

   AWS Tools for PowerShell は、新しい機能で定期的に更新されます。すべての最新機能にアクセスできるように、AWS Tools for PowerShell を定期的にアップグレード (再インストール) します。

2. AWS Tools for PowerShell を設定するには、「AWS Tools for Windows PowerShell ユーザーガイド」の「AWS 認証情報を使用する手順」を参照してください。

   このステップでは、組織の AWS 管理者から付与された認証情報を次のコマンドを使用して指定します。

```
Set-AWSCredential `
-AccessKey AKIAIOSFODNN7EXAMPLE `
-SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY `
-StoreAs MyProfileName
```

   Important

   AWS Tools for PowerShell を設定するときに、`Set-DefaultAWSRegion` を実行して AWS リージョン を指定できます。「AWS 全般リファレンス」に記載されている、AWS AppConfig のサポート対象リージョンを 1 つ選択します。まず、必要に応じて、AWS アカウント の管理者にどのリージョンを選択すべきかについて確認します。

   アクセスキーの詳細については、IAM ユーザーガイドの「IAM ユーザーのアクセスキーの管理」を参照してください。

3. インストールまたはアップグレードを確認するには、AWS Tools for PowerShell から次のコマンドを呼び出します。

```
Get-AWSCmdletName -Service AppConfig
```

成功すると、このコマンドは、使用できる AWS AppConfig コマンドレットのリストを表示します。

AWS AppConfig のアクセス許可の設定

デフォルトでは、のみAWS アカウント管理者は、AWS AppConfig。あなたは許可することができますAWS Identity and Access Management(IAM) ユーザー、グループ、ロールがにアクセスするAWS AppConfigユーザー、グループ、またはロールに割り当てる IAM アクセス権限ポリシーでリソース、アクション、および条件コンテキストキーを指定します。詳細については、「」を参照してください。アクション、リソース、条件キー:AWS AppConfigのサービス認証リファレンス。

> **Important**
>
> セキュリティは、AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。を使用する際に共有責任モデルを適用する方法を理解するにはAWS AppConfig「」を参照してください。セキュリティAWS Systems ManagerのAWS Systems Managerユーザーガイド。AWS AppConfigの一機能ですAWS Systems Manager。

ユーザー、グループ、ロールに、AWS AppConfig で目的のアクションを実行するために必要な最小限の特権を付与する制限付き IAM アクセス許可ポリシーを作成することをお勧めします。

例えば次のように、AWS AppConfig により使用される `Get` および `List` API アクションのみを含む読み取り専用の IAM アクセス許可ポリシーを作成できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetDocument",
        "ssm:ListDocuments",
        "appconfig:GetLatestConfiguration",
        "appconfig:StartConfigurationSession",
        "appconfig:ListApplications",
        "appconfig:GetApplication",
        "appconfig:ListEnvironments",
        "appconfig:GetEnvironment",
        "appconfig:ListConfigurationProfiles",
        "appconfig:GetConfigurationProfile",
        "appconfig:ListDeploymentStrategies",
        "appconfig:GetDeploymentStrategy",
        "appconfig:GetConfiguration",
        "appconfig:ListDeployments",
        "appconfig:GetDeployment"

      ],
      "Resource": "*"
    }
  ]
}
```

> **Important**
>
> StartDeployment および StopDeployment API アクションへのアクセスを、ターゲットに新しい設定をデプロイする責任と結果を理解している信頼できるユーザーにのみ制限します。

IAM ユーザーポリシーの作成と編集の詳細については、IAM ユーザーガイドの「IAM ポリシーの作成」を参照してください。このポリシーを IAM グループに割り当てる方法については、「IAM グループへのポリシーのアタッチ」を参照してください。

AWS AppConfig を使用するアクセス許可を持つ IAM ユーザーアカウントを設定するには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます https://console.aws.amazon.com/iam/。
2. ナビゲーションペインで [Users] を選択します。
3. リストで、名前を選択します。
4. [Permissions] タブを選択します。
5. ページ右側にある [Permission policies (権限のポリシー)] で、[Add inline policy (インラインポリシーの追加)] を選択します。
6. [JSON] タブを選択します。
7. デフォルトのコンテンツをカスタムアクセス許可ポリシーに置き換えます。
8. [ポリシーの確認] を選択します。
9. [Review policy (ポリシーの確認)] ページで、[Name (名前)] にインラインポリシーの名前を入力します。例: `AWSAppConfig-<action>-Access`。
10. [Create policy] を選択します。

(オプション) に基づいてロールバック用のアクセス許可の設定 CloudWatch アラーム

設定することが可能です。AWS AppConfig1 つ以上の Amazon に応じて設定の以前のバージョンにロールバックするには CloudWatch アラーム 応答するようにデプロイを構成する場合 CloudWatch アラームの場合は、AWS Identity and Access Management(IAM) ロール。AWS AppConfig監視できるようにするには、このロールが必要です。 CloudWatch アラーム

> Note
>
> IAM ロールは現在のアカウントに属している必要があります。デフォルトでは、AWS AppConfig は、現在のアカウントによって所有されているアラームのみを監視できます。設定したい場合 AWS AppConfig別のアカウントのメトリックに応答してデプロイをロールバックするには、クロスアカウントアラームを設定する必要があります。詳細については、「」を参照してください。クロスアカウントクロスリージョン CloudWatch コンソールのアマゾン CloudWatch ユーザーガイド。

次の手順を使用して IAM ロールを作成し、これを有効にする IAM ロールを作成します。AWS AppConfig に基づいてロールバックするには CloudWatch アラーム このセクションには、以下の手順が含まれます。

1. ステップ 1: に基づいてロールバック用のアクセス権限ポリシーを作成します。 CloudWatch アラーム (p. 9)
2. ステップ 2: に基づいてロールバック用の IAM ロールを作成します。 CloudWatch アラーム (p. 10)
3. ステップ 3: 信頼関係を追加する (p. 10)

ステップ 1: に基づいてロールバック用のアクセス権限ポリシーを作成します。 CloudWatch アラーム

`DescribeAlarms` API アクションを呼び出すために AWS AppConfig アクセス許可を付与する IAM ポリシーを作成するには、以下の手順を使用します。

CloudWatch アラームに基づいてロールバック用の IAM アクセス許可ポリシーを作成するには

1. https://console.aws.amazon.com/iam/ で IAM コンソール を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択してから [Create policy] (ポリシーの作成) を選択します。
3. [Create policy] (ポリシーの作成) ページで、[JSON] タブを選択します。
4. [JSON] タブのデフォルトのコンテンツを次のアクセス権限ポリシーに置き換え、[] を選択します。次へ: タグ

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

5. このロールのタグを入力し、[] を選択します。次へ: 確認.
6. [Review] (確認) ページで、[Name] (名前) フィールドに「**SSMCloudWatchAlarmDiscoveryPolicy**」を入力します。
7. [Create policy] (ポリシーの作成) を選択します。システムによって ポリシー ページに戻ります。

ステップ 2: に基づいてロールバック用の IAM ロールを作成します。 CloudWatch アラーム

次の手順を使用して、IAM ロールを作成し、前の手順で作成したポリシーをそのロールに割り当てます。

に基づいてロールバック用の IAM ロールを作成するには CloudWatch アラーム

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Roles] を選択し、続いて [Create role] を選択します。
3. [Select type of trusted entity] (信頼されたエンティティの種類を選択) の下で、[AWS service] (AWS サービス) を選択します。
4. すぐ下このロールを使用するサービスを選択で、EC2: EC2 インスタンスの呼び出しを許可するAWS ユーザーに代わってサービス[] を選択してから、次へ: アクセス許可.
5. [Attached permissions policy] (アタッチされたアクセス許可ポリシー) ページで、[SSMCloudWatchAlarmDiscoveryPolicy] を検索します。
6. このポリシーを選択してから、[] を選択します。次へ: タグ
7. このロールのタグを入力し、[] を選択します。次へ: 確認.
8. [Create role] (ロールの作成) ページで、[Role name] (ロール名) フィールドに「**SSMCloudWatchAlarmDiscoveryRole**」を入力し、[Create role] (ロールの作成) を選択します。
9. [ロール] ページで、作成したロールを選択します。[Summary (概要)] ページが開きます。

ステップ 3: 信頼関係を追加する

次の手順を使用して、先ほど作成したロールが AWS AppConfig を信頼するように設定します。

AWS AppConfig の信頼関係を追加するには

1. 作成したロールの [Summary] ページで [Trust Relationships] タブを選択し、[Edit Trust Relationship] を選択します。
2. 次の例に示すように、「`appconfig.amazonaws.com`」のみを含めるようにポリシーを編集します。

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

3. [Update Trust Policy] (信頼ポリシーの更新) をクリックします。

AWS AppConfigの使用

このセクションには、AWS AppConfig 機能を使用して設定を作成し、ホストやターゲットにデプロイする方法について説明するトピックが含まれています。次を使用できます。AWS CloudFormationを多数作成してAWS AppConfigこのセクションで説明するアーティファクト。詳細については、「」を参照してください。AWS AppConfigリソースタイプのリファレンスのAWS CloudFormationユーザーガイド。

トピック

- ステップ 1: の作成AWS AppConfig応用 (p. 12)
- ステップ 2: 環境を作成する (p. 14)
- ステップ 3: 設定プロファイルと機能フラグの作成 (p. 16)
- ステップ 4: デプロイ戦略の作成 (p. 39)
- ステップ 5: 設定のデプロイ (p. 44)
- ステップ 6: 設定の取得 (p. 48)

ステップ 1: の作成AWS AppConfig応用

EclipseAWS AppConfigでは、アプリケーションは単にフォルダのような組織構造です。この組織構成は、実行可能コードの単位との関係を持っています。たとえば、というアプリケーションを作成できます。MyMobileApp ユーザーによってインストールされたモバイルアプリケーションの構成データを整理および管理します。

> Note
>
> 次を使用できます。AWS CloudFormation作成するにはAWS AppConfigアプリケーション、環境、構成プロファイル、デプロイ、デプロイ戦略、ホストされた構成バージョンなどのアーティファクト。詳細については、「」を参照してください。AWS AppConfigリソースタイプのリファレンスのAWS CloudFormationユーザーガイド。

AWS AppConfig アプリケーションの作成 (コンソール)

次の手順に従って、AWS Systems Manager コンソールを使用することで AWS AppConfig アプリケーションを作成します。

アプリケーションを作成するには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. ナビゲーションペインで [AWS AppConfig] を選択します。
3. [Applications (アプリケーション)] タブで、[Create application (アプリケーションの作成)] を選択します。
4. [Name (名前)] に、アプリケーションの名前を入力します。
5. [Description] に、アプリケーションに関する情報を入力します。
6. [Tags (タグ)] セクションで、キーとオプションの値を入力します。1 つのリソースに対して最大 50 個のタグを指定できます。
7. [Create application] を選択します。

AWS AppConfig によってアプリケーションが作成され、[Environments (環境)] タブが表示されます。ステップ 2: 環境を作成する (p. 14) に進みます。「[Environments (環境)] タブで...」と記載されている手順を開始することができます。

AWS AppConfig アプリケーションの作成 (コマンドライン)

以下の手順では、AWS CLI (Linux または Windows の場合) または AWS Tools for PowerShell を使用して AWS AppConfig アプリケーションを作成する方法について説明します。

アプリケーションをステップバイステップで作成するには

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。
2. アプリケーションを作成するには、次のコマンドを実行します。

   Linux

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

   Windows

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

   PowerShell

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

   システムが以下のような情報を返します。

   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
```

ステップ 2: 環境を作成する

AWS AppConfig アプリケーションごとに、1 つ以上の環境を定義します。環境は、の論理的なデプロイグループです。 AppConfig ターゲット (内のアプリケーションなど) `Beta`または`Production`環境。アプリケーションの `Web`、`Mobile`、および `Back-end` といったコンポーネントを含む、アプリケーションのサブコンポーネントの環境を定義することもできます。Amazon を設定できます。 CloudWatch 環境ごとのアラーム。システムは、設定のデプロイ中にアラームをモニタリングします。アラームがトリガーされると、システムは設定をロールバックします。

開始する前に

有効にする場合AWS AppConfigへの応答で設定をロールバックするには CloudWatch アラームを設定する場合は、AWS Identity and Access Management(IAM) ロールで有効にする権限を持つAWS AppConfigに対応する CloudWatch アラーム このロールは、次の手順で選択します。詳細については、「(オプション) に基づいてロールバック用のアクセス許可の設定 CloudWatch アラーム (p. 9)」を参照してください。

AWS AppConfig 環境の作成 (コンソール)

次の手順に従って、AWS Systems Manager コンソールを使用することで AWS AppConfig 環境を作成します。

環境を作成するには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. ナビゲーションペインで [AWS AppConfig] を選択します。
3. [Applications (アプリケーション)] タブで、「ステップ 1: の作成AWS AppConfig応用 (p. 12)」で作成したアプリケーションを選択し、[View details (詳細の表示)] を選択します。
4. [Environments (環境)] タブで、[Create environment (環境を作成)] を選択します。
5. [Name (名前)] に、環境の名前を入力します。
6. [Description (説明)] に、環境に関する情報を入力します。
7. [IAM role] (IAM ロール) リストで、アラームがトリガーされたときに設定をロールバックするアクセス許可を持つ IAM ロールを選択します。
8. [CloudWatch alarms] (CloudWatch アラーム) リストで、モニタリングするアラームを 1 つ以上選択します。AWS AppConfig は、これらのアラームの 1 つがアラーム状態になった場合、設定デプロイをロールバックします。
9. [Tags (タグ)] セクションで、キーとオプションの値を入力します。1 つのリソースに対して最大 50 個のタグを指定できます。
10. [Create environment] (環境の作成) を選択します。

AWS AppConfig は環境を作成して、[Environment details (環境詳細)] ページを表示します。ステップ 3: 設定プロファイルと機能フラグの作成 (p. 16) に進みます。

AWS AppConfig 環境の作成 (コマンドライン)

以下の手順では、AWS CLI (Linux または Windows の場合) または AWS Tools for PowerShell を使用して AWS AppConfig 環境を作成する方法について説明します。

環境をステップバイステップで作成する

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。
2. 以下のコマンドを実行して、環境を作成します。

   Linux

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

   Windows

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

   PowerShell

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

   システムが以下のような情報を返します。

   Linux

   ```
   {
      "ApplicationId": "The application ID",
      "Id": "The_environment ID",
   ```

```
   "Name": "Name of the environment",
   "State": "The state of the environment",
   "Description": "Description of the environment",

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

Windows

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

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

PowerShell

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

ステップ 3: 設定プロファイルと機能フラグの作成

設定は、アプリケーションの動作に影響する設定のコレクションです。設定プロファイルにより AWS AppConfig は設定にアクセスできます。設定プロファイルには、次に示す情報が含まれます。

- 設定が保存される URI の場所。
- 設定へのアクセスを提供する AWS Identity and Access Management (IAM) ロール。
- 設定データのバリデータ。JSON スキーマまたは AWS Lambda 関数のいずれかを使用して、設定プロファイルを検証できます。設定プロファイルは、最大 2 つのバリデータを持つことができます。

AWS AppConfigでは、次のタイプの設定プロファイルをサポートしています。

- 機能フラグ: 機能フラグ設定を使用して、製品の発売や発表など、適切なタイミングでのデプロイが必要となる新機能をオンにします。
- フリーフォーム: フリーフォーム構成を使用して、アプリケーションに変更を慎重に導入します。

目次

- 設定例 (p. 17)
- 設定プロファイルの IAM ロールについて (p. 19)
- Amazon S3 に保存された設定について (p. 20)
- バリデータについて (p. 22)
- 機能フラグ設定プロファイルの作成 (p. 24)
- フリーフォーム構成プロファイルの作成 (p. 34)

設定例

AWS Systems Manager の機能である AWS AppConfig を使用して、アプリケーション設定を作成、管理し、迅速にデプロイします。設定は、アプリケーションの動作に影響する設定のコレクションです。次に例を示します。

機能フラグの設定

次の機能フラグ設定は、リージョンごとにモバイル決済とデフォルト支払いを有効または無効にします。

JSON

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

YAML

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

運用設定

次の運用構成では、アプリケーションが要求を処理する方法に制限が適用されます。

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
```

アクセスコントロールリストの設定

次のアクセス制御リスト構成では、アプリケーションにアクセスできるユーザーまたはグループを指定します。

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
```

設定プロファイルの IAM ロールについて

次の手順で説明するように、AWS AppConfig を使用して、設定データへのアクセスを提供する IAM ロールを作成できます。自分で IAM ロールを作成して、リストから選択することもできます。AWS AppConfig を使用してロールを作成する場合、システムによってロールが作成され、選択した構成ソースのタイプに応じて、次のいずれかのアクセス許可ポリシーが指定されます。

設定ソースは SSM ドキュメント

```
{

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

設定ソースはパラメータストアパラメータ

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
                "Arn:aws:ssm:AWS-Region:account-number:parameter/parameter-name"
            ]
        }
    ]
    }
```

AWS AppConfig を使用してロールを作成する場合、システムはロールの次の信頼関係も作成します。

```
{

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

Amazon S3 に保存された設定について

設定は、Amazon Simple Storage Service (Amazon S3) バケットに保存できます。設定プロファイルの作成時には、バケット内の 1 つの S3 オブジェクトの URI を指定します。また、そのオブジェクトを取得するために AWS AppConfig のアクセス許可を付与する AWS Identity and Access Management (IAM) ロールの Amazon リソースネーム (ARN) を指定します。Amazon S3 オブジェクトの設定プロファイルを作成する場合は、次の制限事項に注意してください。

制限	詳細
[Size] (サイズ)	S3 オブジェクトとして保存できる設定のサイズは最大 1 MB です。
オブジェクト暗号化	設定プロファイルは、SSE-S3 で暗号化されたオブジェクトのみ対象にすることができます。
ストレージクラス	AWS AppConfig は、S3 ストレージクラスとして、`STANDARD`、`INTELLIGENT_TIERING`、`REDUCED_REDUNDA`をサポートしています。次のクラスはサポートしていません。すべての S3 Glacier クラス (`GLACIER`そして`DEEP_ARCHIVE`).
バージョニング	AWS AppConfig では、S3 オブジェクトでバージョニングが使用されている必要があります。

Amazon S3 オブジェクトとして保存する設定のアクセス許可の設定

S3 オブジェクトとして保存する設定の設定プロファイルを作成する場合は、そのオブジェクトを取得するアクセス許可を AWS AppConfig に付与する IAM ロールの ARN を指定する必要があります。このロールには、以下のアクセス許可が含まれている必要があります。

S3 オブジェクトに対するアクセス許可

- s3:GetObject
- s3:GetObjectVersion

S3 オブジェクトを一覧表示するアクセス許可

s3:ListAllMyBuckets

オブジェクトを保存する S3 バケットへのアクセス許可

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

以下の手順を実行して、S3 オブジェクトに保存されている設定を AWS AppConfig が取得できるようにするロールを作成します。

S3 オブジェクトにアクセスするための IAM ポリシーの作成

S3 オブジェクトに保存されている設定を AWS AppConfig が取得できるようにする IAM ポリシーを作成するには、以下の手順に従います。

S3 オブジェクトにアクセスするための IAM ポリシーを作成するには

1. https://console.aws.amazon.com/iam/ で IAM コンソール を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択してから [Create policy] (ポリシーの作成) を選択します。
3. [Create policy] (ポリシーの作成) ページで、[JSON] タブを選択します。
4. S3 バケットおよび設定オブジェクトについての情報を、次のサンプルポリシーで更新します。次に、そのポリシーを [JSON] タブのテキストフィールドに貼り付けます。

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

5. レビューポリシーを選択します。
6. [ポリシーの確認] ページの [名前] ボックスに名前を入力し、続いて説明を入力します。
7. [Create policy] (ポリシーの作成) を選択します。[Roles] (ロール) ページが再度表示されます。

S3 オブジェクトにアクセスするための IAM ロールの作成

S3 オブジェクトに保存されている設定を AWS AppConfig が取得できるようにする IAM ロールを作成するには、以下の手順に従います。

Amazon S3 オブジェクトにアクセスするための IAM ポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Roles] を選択し、続いて [Create role] を選択します。
3. [Select type of trusted entity] (信頼するエンティティのタイプを選択) で [AWS service] (AWS サービス) を選択します。

4. 左ユースケースを選択するセクション、[]一般的ユースケースで、EC2[] を選択してから、次へ: アクセス許可.
5. [アクセス権限ポリシーをアタッチする] ページで、検索ボックスに前の手順で作成したポリシーの名前を入力します。
6. ポリシーを選択してから、次へ: タグ
7. [Add tags (optional) (タグ (任意) の追加)] ページでキーと任意の値を入力して、[Next:Review (次へ: 確認)] を選択します。
8. [Review (確認)] ページの [ロール名] フィールドに名前を入力し、続いて説明を入力します。
9. [Create role] (ロールの作成) を選択します。[Roles] (ロール) ページが再度表示されます。
10. [Roles] ページで作成したロールを選択して、[Summary] ページを開きます。[ロール名] と [ロール ARN] を書き留めます。このロール ARN は、このトピックで後述する設定プロファイルの作成時に指定します。

信頼関係の作成

次の手順を使用して、先ほど作成したロールが AWS AppConfig を信頼するように設定します。

信頼関係を追加するには

1. 作成したロールの [Summary] ページで [Trust Relationships] タブを選択し、[Edit Trust Relationship] を選択します。
2. 次の例に示すように、`"ec2.amazonaws.com"` を削除して `"appconfig.amazonaws.com"` を追加します。

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

3. [Update Trust Policy] (信頼ポリシーの更新) をクリックします。

バリデータについて

設定および設定プロファイルを作成する場合、最大 2 つのバリデータを指定できます。バリデータは、設定データが構文的かつ意味的に正しいことを保証します。バリデータは、JSON スキーマで、または AWS Lambda 関数として作成できます。

> **Important**
>
> SSM ドキュメントに保存された設定データは、設定をシステムに追加する前に、関連付けられた JSON スキーマに対して検証する必要があります。SSM パラメータには検証方式は必要ありませんが、AWS Lambda を使用して、新規または更新された SSM パラメータ設定の検証チェックを作成することを推奨します。

JSON スキーマバリデータ

SSM ドキュメントで設定を作成する場合は、その設定の JSON スキーマを指定または作成する必要があります。JSON スキーマは、アプリケーション構成設定ごとに許可されるプロパティを定義します。この

JSON スキーマは、新規または更新された構成設定がアプリケーションに必要なベストプラクティスに準拠するようにするための一連のルールのように機能します。以下はその例です。

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

SSM ドキュメントから設定を作成する場合、システムは設定がスキーマの要件に準拠していることを確認します。そうでない場合は、AWS AppConfig は、検証エラーを返します。

Note

JSON スキーマバリデータに関する以下の情報に注意してください。

- SSM ドキュメント内の設定は、`ApplicationConfiguration`ドキュメントタイプ。対応する JSON スキーマでは、`ApplicationConfigurationSchema`ドキュメントタイプ。
- AWS AppConfig は、インラインスキーマの JSON スキーマバージョン 4.X をサポートします。アプリケーション設定で JSON スキーマの異なるバージョンが必要な場合は、Lambda バリデータを作成する必要があります。

AWS Lambda バリデータ

Lambda 関数バリデータは、次のイベントスキーマで設定する必要があります。AWS AppConfig はこのスキーマを使用して Lambda 関数を呼び出します。中身は base64 でエンコードされた文字列で、URI は文字列です。

```
{
    "ApplicationId": "The application Id of the configuration profile being validated",
    "ConfigurationProfileId": "The configuration profile Id of the configuration profile
 being validated",
    "ConfigurationVersion": "The configuration version of the configuration profile being
 validated",
    "Content": "Base64EncodedByteString",
    "Uri": "The uri of the configuration"
}
```

AWS AppConfig は、Lambda `X-Amz-Function-Error` ヘッダーが応答に設定されていることを検証します。Lambda は、関数が例外をスローした場合にこのヘッダーを設定します。`X-Amz-Function-Error` の詳細については、「AWS Lambda デベロッパーガイド」の「エラー処理と AWS Lambda での自動再試行」を参照してください。

検証を成功させる Lambda 応答コードの簡単な例を以下に示します。

```
import json

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

検証を失敗させる Lambda 応答コードの簡単な例を以下に示します。

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

設定パラメータが素数の場合にのみ有効とする、別の例を次に示します。

```
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 は、`StartDeployment` および `ValidateConfigurationActivity` API アクションを呼び出すときに検証 Lambda を呼び出します。Lambda を呼び出すには、`appconfig.amazonaws.com` アクセス許可を提供する必要があります。詳細については、「AWS のサービスへのアクセス権限を関数に付与する」を参照してください。AWS AppConfig は検証 Lambda の実行時間を、起動レイテンシーを含めて 15 秒に制限します。

機能フラグ設定プロファイルの作成

機能フラグを使用して、アプリケーション内の機能を有効または無効にしたり、フラグ属性を使用してアプリケーション機能のさまざまな特性を設定したりできます。AWS AppConfig は、機能フラグ設定を、フラグとフラグ属性に関するデータおよびメタデータを含む機能フラグ形式で、AWS AppConfig でホストされた設定ストアに保存します。AWS AppConfig でホストされた設定ストアの詳細については、「AWS AppConfig でホストされた設定ストアについて (p. 35)」のセクションを参照してください。

> **Important**
>
> 機能フラグ設定データを取得するには、アプリケーションが`GetLatestConfiguration`アピ。を呼び出して機能フラグの設定データを取得することはできません。`GetConfiguration`。詳細については、「」を参照してください。最新設定を取得のAWS AppConfigAPI リファレンス。

トピック

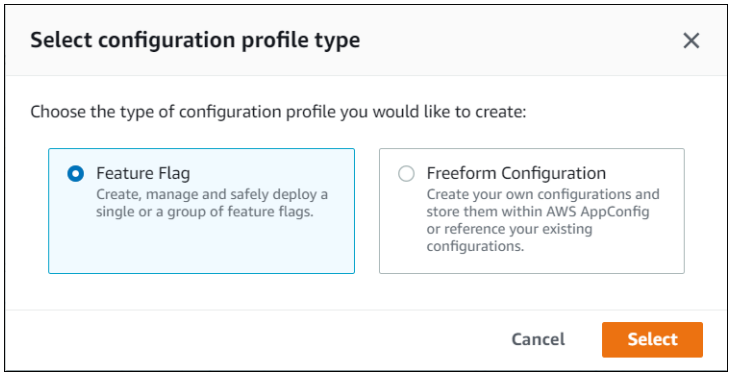
- 機能フラグおよび機能フラグ設定プロファイルの作成 (コンソール) (p. 25)
- 機能フラグと機能フラグの設定プロファイルの作成 (コマンドライン) (p. 26)
- AWS.AppConfig.FeatureFlags のタイプリファレンス (p. 28)

機能フラグおよび機能フラグ設定プロファイルの作成 (コンソール)

AWS AppConfig コンソールを使用して AWS AppConfig 機能フラグの設定プロファイルと機能フラグ設定を作成するには、次の手順に従います。

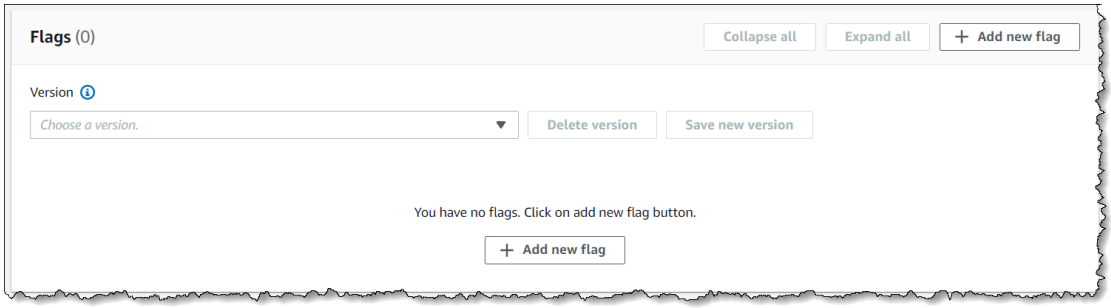
設定プロファイルを作成するには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. [Applications] (アプリケーション) タブで、AWS AppConfig 設定を作成する (p. 12)で作成したアプリケーションを選択し、[Configuration profiles and feature flags] (設定プロファイルと機能フラグ) タブを選択します。
3. [Create] を選択します。
4. [Feature flag] (機能フラグ) を選択します。

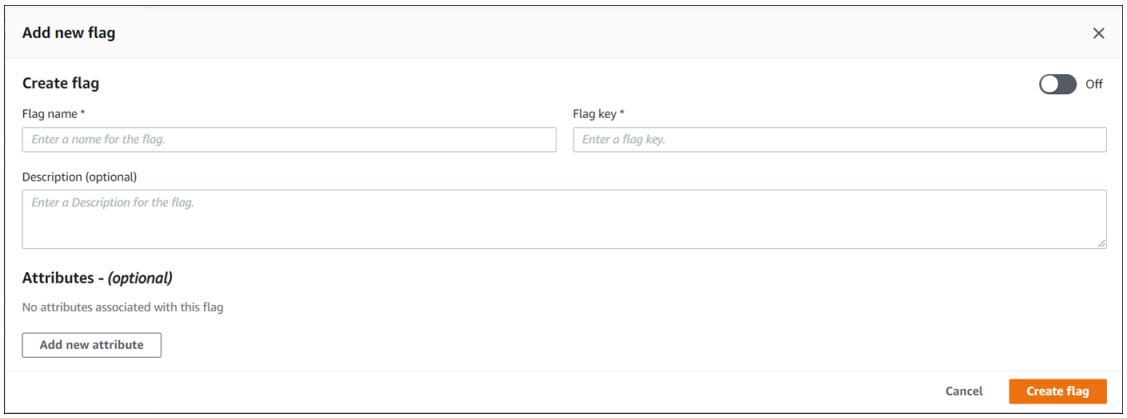


機能フラグを作成する

1. 作成した設定で、[Add new flag] (新しいフラグを追加する) を選択します。



2. [Flag name] (フラグの名前) と (オプションで) [Description] (説明) を入力します。フラグキーは、入力した名前のスペースをアンダースコアに置き換えることで自動入力されます。別の値または形式を使用する場合は、フラグキーを編集できます。フラグを作成した後、フラグ名は編集できますが、フラグキーは編集できません。



3. トグルボタンを使用して、機能フラグが [Enabled] (有効) または [Disabled] (無効) のいずれであるかを指定します。
4. (オプション) 機能フラグに [Attributes] (属性) と [Constraints] (制約) を追加します。属性を使用すると、フラグ内に追加の値を指定できます。オプションで、属性値を指定した制約と比較して検証することができます。制約により、予期しない値がアプリケーションにデプロイされないようにします。

   AWS AppConfig 機能フラグでは、次のタイプの属性および対応する制約をサポートします。

型	制約事項	説明
[String] (文字列)	正規表現	文字列の正規表現パターン
	列挙型	文字列に許容される値のリスト
数値	最小	属性の最小数値
	[Maximum] (最大)	属性の最大数値
[Boolean] (ブール値)	[None] (なし)	[None] (なし)
文字列配列	正規表現	配列の要素の正規表現パターン
	列挙型	配列の要素に許容される値のリスト
数値配列	最小	配列の要素の最小数値
	[Maximum] (最大)	配列の要素の最大数値

   Note

   [Required value] (必須の値) を選択して、属性値が必須かどうかを指定します。
5. [Save new version] (新しいバージョンを保存する) を選択します。

ステップ 4 に進みます。デプロイ戦略の作成。

機能フラグと機能フラグの設定プロファイルの作成 (コマンドライン)

次の手順では、を使用する方法を説明します。AWS Command Line Interface(Linux または Windows の場合) または Tools for Windows PowerShell 作成するにはAWS AppConfig機能フラグ設定プロファイル。

希望する場合には、AWS CloudShellをクリックして、以下に示すコマンドを実行します。詳細については、AWS CloudShell ユーザーガイドの「AWS CloudShell とは何か」を参照してください。

機能フラグの設定をステップバイステップで作成する

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。

2. [Type] (タイプ) に `AWS.AppConfig.FeatureFlags` を指定して、機能フラグの設定プロファイルを作成します。設定プロファイルでは、ロケーション URI に `hosted` を使用する必要があります。

   Linux

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

   Windows

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

   PowerShell

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

3. 機能フラグの設定データを作成します。データは JSON 形式であり、`AWS.AppConfig.FeatureFlags` JSON スキーマに準拠していることが必要です。スキーマの詳細については、「AWS.AppConfig.FeatureFlags のタイプリファレンス (p. 28)」を参照してください。

4. `CreateHostedConfigurationVersion` API を使用して、機能フラグの設定データを AWS AppConfig に保存します。

   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
```

システムが以下のような情報を返します。

Linux

```
{
   "ApplicationId"          : "The application ID",
   "ConfigurationProfileId" : "The configuration profile ID",
   "VersionNumber"          : "The configuration version number",
   "ContentType"            : "application/json"
}
```

Windows

```
{
   "ApplicationId"          : "The application ID",
   "ConfigurationProfileId" : "The configuration profile ID",
   "VersionNumber"          : "The configuration version number",
   "ContentType"            : "application/json"
}
```

PowerShell

```
ApplicationId          : The application ID
ConfigurationProfileId : The configuration profile ID
VersionNumber          : The configuration version number
ContentType            : application/json
```

`service_returned_content_file` には、AWS AppConfig によって生成されたメタデータをいくつか含む設定データが含まれます。

> **Note**
>
> ホストされた設定バージョンを作成すると、AWS AppConfig は、データが `AWS.AppConfig.FeatureFlags` JSON スキーマに準拠していることを検証します。さらに、AWS AppConfig は、データ内の各機能フラグ属性が、各属性に定義された制約を満たすことを検証します。

AWS.AppConfig.FeatureFlags のタイプリファレンス

`AWS.AppConfig.FeatureFlags` JSON スキーマを、機能フラグの設定データを作成するためのリファレンスとして使用します。

```
{
      "$schema": "http://json-schema.org/draft-07/schema#",
      "definitions": {
        "flagSetDefinition": {
          "type": "object",
          "properties": {
            "version": {
              "$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",
          "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

機能フラグ設定データを取得するには、アプリケーションが`GetLatestConfiguration`アピ。を呼び出して機能フラグの設定データを取得することはできません。`GetConfiguration`。詳細については、「」を参照してください。最新設定を取得のAWS AppConfigAPI リファレンス。

アプリケーションが呼び出すとき最新設定を取得が新しくデプロイされた設定を受信すると、機能フラグと属性を定義する情報が削除されます。簡略化された JSON には、指定した各フラグキーと一致するキーのマップが含まれています。簡略化された JSON には、のマップされた値も含まれます。`true`または`false`向けの`enabled`属性。フラグが設定されている場合`enabled`に`true`の場合、フラグのアトリビュートも存在することになります。次の JSON スキーマは、JSON 出力の形式を示しています。

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

```
    }
  }
}
```

フリーフォーム構成プロファイルの作成

フリーフォーム設定プロファイルにより、AWS AppConfig は指定されたソースロケーションから設定にアクセスすることができます。フリーフォーム設定を保存できる形式と場所は次のとおりです。

- AWS AppConfig でホストされた設定ストア内の YAML、JSON、またはテキストドキュメント。
- Amazon Simple Storage Service (Amazon S3) バケット内のオブジェクト。
- Systems Manager ドキュメントストア内のドキュメント。
- 統合ソースアクションはすべて AWS CodePipeline のサポート対象です。

AWS AppConfig でホストされた設定ストアまたは SSM ドキュメントに保存するフリーフォーム設定の場合、設定プロファイルの作成時に Systems Manager コンソールを使用してフリーフォーム設定を作成できます。このプロセスについては、このトピックの後半で説明します。

SSM パラメータまたは S3 に保存する設定の場合は、最初にパラメータまたはオブジェクトを作成してから、パラメータストアまたは S3 に追加する必要があります。パラメータまたはオブジェクトを作成したら、このトピックの手順に従って設定プロファイルを作成できます。パラメータストアでのパラメータの作成方法については、「AWS Systems Manager ユーザーガイド」の「Systems Manager パラメータを作成する」を参照してください。

トピック

- 設定ストアのクォータと制限について (p. 34)
- AWS AppConfig でホストされた設定ストアについて (p. 35)
- フリーフォーム構成および構成プロファイルの作成 (p. 35)

設定ストアのクォータと制限について

AWS AppConfig でサポートされる設定ストアには、以下のクォータと制限があります。

	AWS AppConfig でホストされた設定ストア	S3	パラメータストア	ドキュメントストア	AWS CodePipeline
設定サイズの制限	1 MB	1 MB S3 ではなく AWS AppConfig によって適用	4 KB (無料利用枠)/8 KB (詳細パラメータ)	64 KB	1 MB CodePipeline ではなく、AWS AppConfig によって適用
リソースストレージの制限	1 GB	無制限	10,000 パラメータ (無料利用枠)/100,000 パラメータ (詳細パラメータ)	500 ドキュメント	アプリケーションごとの設定プロファイルの数によって制限される (アプリケーションあたり 100 プロファイル)

	AWS AppConfig でホストされた設定ストア	S3	パラメータストア	ドキュメントストア	AWS CodePipeline
サーバー側の暗号化	はい	SSE-S3	いいえ	いいえ	はい
AWS CloudFormation のサポート	はい	データの作成または更新用ではありません	はい	いいえ	はい
create または update API アクションを検証する	サポート外	サポート外	サポートされている正規表現	すべての put および update API アクションに必要な JSON スキーマ	サポート外
料金表	無料	「Amazon S3 の料金」を参照してください	「AWS Systems Manager 料金表」を参照してください	無料	「AWS CodePipeline 料金表」を参照してください

AWS AppConfig でホストされた設定ストアについて

AWS AppConfig には、内部の設定ストアまたはホストされた設定ストアが含まれています。設定は 1 MB 以下である必要があります。AWS AppConfig でホストされた設定ストアには、他の設定ストアオプションと比べて次のような利点があります。

- Amazon Simple Storage Service (Amazon S3) やパラメータストアなど、他のサービスをセットアップして設定する必要はありません。
- 設定ストアを使用するために AWS Identity and Access Management (IAM) アクセス許可を設定する必要はありません。
- 設定を YAML、JSON、またはテキストドキュメントとして保存できます。
- ストアを使用してもコストは発生しません。
- 設定プロファイルを作成したら、作成した設定をストアに追加できます。

フリーフォーム構成および構成プロファイルの作成

このセクションでは、フリーフォーム設定および設定プロファイルを作成する方法について説明します。開始する前に、以下の情報に注意してください。

- 次の手順では、選択した設定ストアの設定データに AWS AppConfig がアクセスできるように、IAM サービスロールを指定する必要があります。AWS AppConfig でホストされた設定ストアを使用する場合、このロールは必要ありません。S3、パラメータストア、または Systems Manager のドキュメントストアを選択した場合は、既存の IAM ロールを選択するか、システムによってロールを自動的に作成するオプションを選択する必要があります。詳細については、「設定プロファイルの IAM ロールについて (p. 19)」を参照してください。
- S3 に保存されている設定用の設定プロファイルを作成する場合は、アクセス許可を設定する必要があります。S3 を設定ストアとして使用するためのアクセス許可およびその他の要件の詳細については、「Amazon S3 に保存された設定について (p. 20)」を参照してください。
- バリデータを使用する場合は、バリデータを使用するための詳細と要件を確認してください。詳細については、「バリデータについて (p. 22)」を参照してください。

トピック

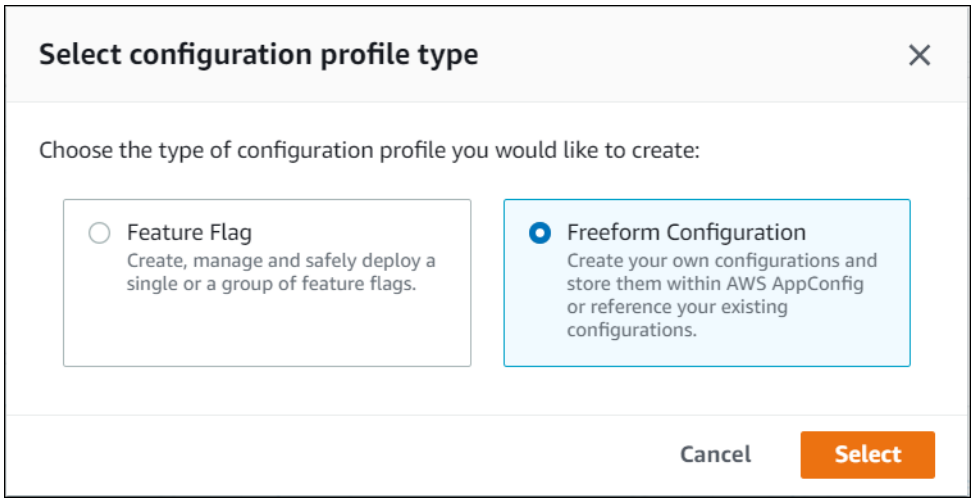
- AWS AppConfig フリーフォーム設定プロファイルの作成 (コンソール) (p. 36)
- AWS AppConfig フリーフォーム設定プロファイルの作成 (コマンドライン) (p. 38)

AWS AppConfig フリーフォーム設定プロファイルの作成 (コンソール)

AWS Systems Manager コンソールを使用して AWS AppConfig フリーフォーム設定プロファイルと (オプションで) フリーフォーム設定を作成するには、次の手順に従います。

設定プロファイルを作成するには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. [Applications] (アプリケーション) タブで、AWS AppConfig 設定を作成する (p. 12)で作成したアプリケーションを選択し、[Configuration profiles and feature flags] (設定プロファイルと機能フラグ) タブを選択します。
3. [Create freeform configuration profile] (フリーフォーム設定プロファイルの作成) を選択します。



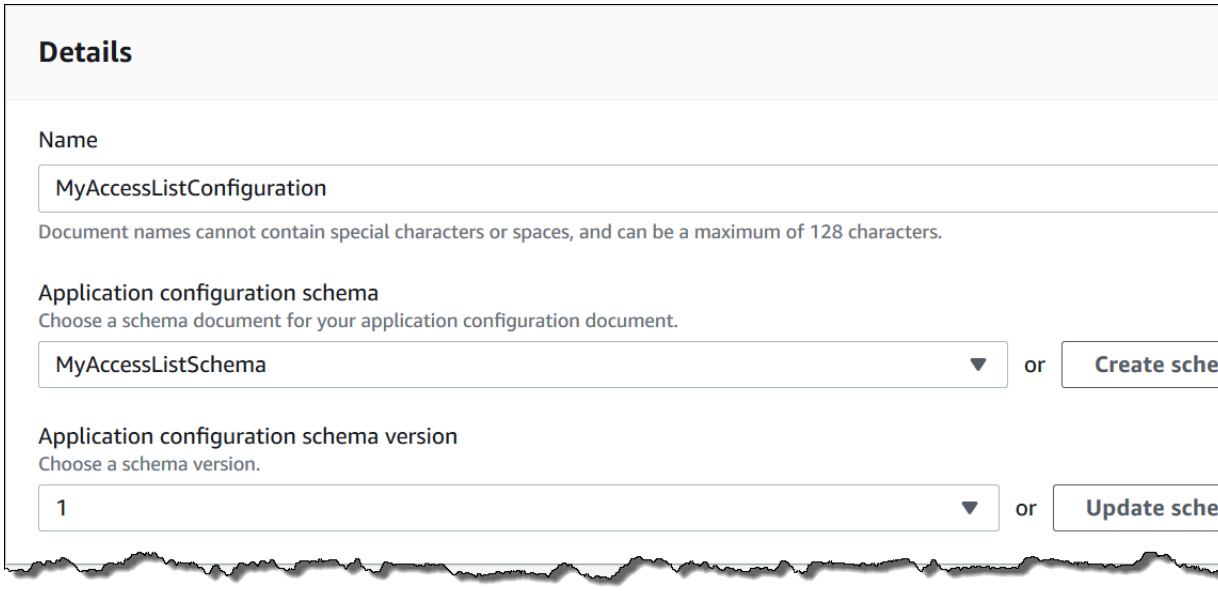
4. [Name (名前)] に、設定プロファイルの名前を入力します。
5. [Description (説明)] に、設定プロファイルに関する情報を入力します。
6. [Select configuration source (設定のソースの選択)] ページで、オプションを選択します。
7. - [AWS AppConfig hosted configuration] (AWS AppConfig でホストされた設定) を選択した場合は、[YAML]、[JSON]、または [Text] (テキスト) のいずれかを選択し、フィールドに設定を入力します。[次へ] を選択し、この手順のステップ 10 に進みます。
   - [Amazon S3 object] (Amazon S3 オブジェクト) を選択した場合は、オブジェクトの URI を入力します。[次へ] を選択します。
   - [AWS Systems Manager parameter] (AWS Systems Manager パラメータ) を選択した場合は、リストからパラメータの名前を選択します。[次へ] を選択します。
   - [AWS CodePipeline] を選択した場合、[Next] (次へ) を選択して、この手順のステップ 10 に進みます。
   - [AWS Systems Manager document] (AWS Systems Manager ドキュメント) を選択した場合は、次の手順を実行します。

     a. [Document source (ドキュメントのソース)] セクションで、[Saved document (保存されたドキュメント)] または [New document (新しいドキュメント)] を選択します。

b. [Saved document] (保存されたドキュメント) を選択した場合は、リストから SSM ドキュメントを選択します。[New document (新しいドキュメント)] を選択すると、[Details (詳細)] セクションと [Content (コンテンツ)] セクションが表示されます。

c. [Details (詳細)] セクションで、新しいアプリケーション設定の名前を入力します。

d. [Application configuration schema (アプリケーション設定スキーマ)] セクションで、リストを使用して JSON スキーマを選択するか、[Create schema (スキーマの作成)] を選択します。[Create schema] (スキーマの作成) を選択すると、Systems Manager は [Create schema] (スキーマの作成) ページを開きます。[Content (コンテンツ)] セクションにスキーマの詳細を入力し、[Create schema (スキーマの作成)] を選択します。



e. [Application configuration schema version (アプリケーション設定スキーマのバージョン)] では、リストからバージョンを選択するか、[Update schema (スキーマを更新)] を選択してスキーマを編集し、新しいバージョンを作成します。

f. [Content (コンテンツ)] セクションで、[YAML] または [JSON] を選択し、フィールドに設定データを入力します。

g. [次へ] を選択します。

8. [Service role] (サービスロール) セクションで、[New service role] (新しいサービスロール) を選択して AWS AppConfig によって、設定データへのアクセスを提供する IAM ロールを作成します。AWS AppConfig は前に入力した名前を基に自動的に [Role name] (ロール名) フィールドに入力します。または、IAM に既に存在するロールを選択するには、[Existing service role] (既存のサービスロール) を選択します。[Role ARN (ロール ARN)] リストを使用してロールを選択します。
9. [Add validators (バリデータの追加)] ページで、[JSON Schema (JSON スキーマ)] または [AWS Lambda] を選択します。[JSON Schema (JSON スキーマ)] を選択した場合、フィールドに JSON スキーマを入力します。[AWS Lambda] を選択した場合は、リストから関数 Amazon リソースネーム (ARN) とバージョンを選択します。

   Important

   SSM ドキュメントに保存された設定データは、設定をシステムに追加する前に、関連付けられた JSON スキーマに対して検証する必要があります。SSM パラメータには検証方式は必要ありませんが、AWS Lambda を使用して、新規または更新された SSM パラメータ設定の検証チェックを作成することを推奨します。
10. [Tags (タグ)] セクションで、キーとオプションの値を入力します。1 つのリソースに対して最大 50 個のタグを指定できます。
11. [Create configuration profile (設定プロファイルの作成)] を選択します。

    Important

    の設定プロファイルを作成した場合AWS CodePipelineの順にクリックし、次のセクションで説明するように、デプロイストラテジーを作成した後、でパイプラインを作成する必要があります。CodePipeline 指定するAWS AppConfigとしてプロバイダをデプロイする。を指定するパイプラインを作成する方法については、AWS AppConfigデプロイプロバイダーとして、を参照してください。チュートリアル: を使用するパイプラインを作成するAWS AppConfigデプロイプロバイダーとしてのAWS CodePipelineユーザーガイド。

ステップ 4: デプロイ戦略の作成 (p. 39) に進みます。

AWS AppConfig フリーフォーム設定プロファイルの作成 (コマンドライン)

以下の手順では、AWS CLI (Linux または Windows の場合) または AWS Tools for PowerShell を使用して AWS AppConfig フリーフォーム設定プロファイルを作成する方法について説明します。希望する場合には、AWS CloudShellをクリックして、以下に示すコマンドを実行します。詳細については、AWS CloudShell ユーザーガイドの「AWS CloudShell とは何か」を参照してください。

設定プロファイルをステップバイステップで作成するには

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。
2. 次のコマンドを実行して、フリーフォーム設定プロファイルを作成します。

   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_Locati
 \
  --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_Locati
 ^
  --tags User_defined_key_value_pair_metadata_of_the_configuration_profile ^
  --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

PowerShell

```
New-APPCConfigurationProfile `
  -Name A_name_for_the_configuration_profile `
  -ApplicationId The_application_ID `
  -Description Description_of_the_configuration_profile `
  -LocationUri A_URI_to_locate_the_configuration or hosted `
  -
RetrievalRoleArn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_spe
 `
  -
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"
```

Note

でコンフィギュレーションを作成した場合AWS AppConfigホストされた構成ストアでは、を使用して、構成の新しいバージョンを作成できます。ホスト構成バージョンの作成API アクション。を表示するにはAWS CLIこの API アクションの詳細とサンプルコマンドについては、を参照してください。ホスト構成バージョンの作成のAWS CLIコマンドリファレンス。

ステップ 4: デプロイ戦略の作成

AWS AppConfig デプロイ戦略は設定デプロイの次の重要な側面を定義します。

設定	説明
デプロイタイプ	デプロイの種類は、設定のデプロイ方法またはロールアウト方法を定義します。AWS AppConfig は、線形および指数デプロイタイプをサポートしています。 • Linear (リニア): このタイプでは、AWS AppConfigは、がデプロイ全体に均等に分散される増加係数の増分によってデプロイを処理します。たとえば、ステップパーセンテージとして 20 を使用する線形デプロイでは、最初はター

設定	説明

	ゲットの 20% に対して設定を利用可能にします。デプロイ時間の 5 分の 1 が経過すると、システムによりパーセンテージが 40% に更新されます。これは、ターゲットの 100% がデプロイされた設定を受信するように設定されるまで続きます。
	• 指数的: このタイプでは、AWS AppConfigは、次の式を使用してデプロイを指数的に処理します。`G*(2^N)`。 この式では、`G`ユーザーが指定したステップパーセンテージであり、`N`は、設定がすべてのターゲットにデプロイされるまでのステップ数です。たとえば、増加係数に 2 を指定すると、次のように設定がロールアウトされます。
	`2*(2^0)`
	`2*(2^1)`
	`2*(2^2)`
	デプロイは、ターゲットの 2%、ターゲットの 4%、ターゲットの 8%、のようにロールアウトされ、設定がすべてのターゲットにデプロイされるまで続行されます。

ステップパーセンテージ (増加係数)	この設定では、デプロイの各ステップでターゲットとする発信者の割合を指定します。 Note SDK および AWS AppConfig API リファレンス では、`step percentage` を `growth factor` と呼んでいます。
デプロイ時間	この設定では、AWS AppConfig がホストにデプロイする時間の合計を指定します。これはタイムアウト値ではありません。これは、デプロイが間隔を置いて処理される時間枠です。
ベイク時間	この設定では、時間の長さを指定します。AWS AppConfigAmazon のモニター CloudWatch がデプロイの次のステップに進む前、またはデプロイを完了とみなす前にアラームを鳴らします。この間にアラームがトリガーされた場合、AWS AppConfig はデプロイをロールバックします。CloudWatch アラームに基づいて AWS AppConfig をロールバックするアクセス許可を設定する必要があります。詳細については、「(オプション) に基づいてロールバック用のアクセス許可の設定 CloudWatch アラーム (p. 9)」を参照してください。

定義済みのデプロイ戦略

AWS AppConfig には、構成をすばやく展開するために役立つ定義済みのデプロイ戦略が含まれています。設定をデプロイするときには、独自の戦略を作成する代わりに、次のいずれかを選択できます。

デプロイ戦略	説明
AppConfig.AllAtOnce	クイック: この戦略では、すべてのターゲットにただちに設定をデプロイします。システムは Amazon を監視します。 CloudWatch 10 分間アラームします。この時間内にアラームが受信されなければ、デプロイは完了です。この間にアラームがトリガーされた場合、 AppConfig はデプロイをロールバックします。
AppConfig.Linear50PercentEvery30Seconds	テスト/デモンストレーション: この戦略では、30 秒ごとに設定をすべてのターゲットの半分にデプロイし、1 分間のデプロイを行います。システムは Amazon を監視します。CloudWatch アラームは 1 分間行います。この時間内にアラームが受信されなければ、デプロイは完了です。この間にアラームがトリガーされた場合、 AppConfig はデプロイをロールバックします。 この戦略では、継続時間と処理時間が短いため、テストまたはデモンストレーションの目的でのみ使用することをお勧めします。
AppConfig.Canary10Percent20Minutes	AWS の推奨: この戦略では、20 分間にわたって 10% の増加係数を使用し、デプロイを指数関数的に処理します。システムは Amazon を監視します。CloudWatch 10 分間アラームします。この時間内にアラームが受信されなければ、デプロイは完了です。この間にアラームがトリガーされた場合、AppConfig はデプロイをロールバックします。 この戦略は、設定のデプロイに関する AWS のベストプラクティスに沿っているため、本稼働環境のデプロイに使用することをお勧めします。

デプロイ戦略の作成

最大 20 のデプロイ戦略を作成できます。設定をデプロイするときに、アプリケーションおよび環境に最適なデプロイ戦略を選択できます。

AWS AppConfig デプロイ戦略の作成 (コンソール)

AWS Systems Manager コンソールを使用して AWS AppConfig デプロイ戦略を作成するには、次の手順に従います。

デプロイ戦略を作成するには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. ナビゲーションペインで、[AWS AppConfig] を選択します。
3. [Deployment Strategies (デプロイ戦略)] タブを選択し、[Create deployment strategy (デプロイ戦略の作成)] を選択します。
4. [Name (名前)] に、デプロイ戦略の名前を入力します。
5. [Description (説明)] に、デプロイ戦略に関する情報を入力します。
6. [Deployment type (デプロイタイプ)] で、タイプを選択します。
7. [Step percentage (ステップパーセンテージ)] で、デプロイの各ステップでターゲットとする発信者の割合を選択します。
8. [Deployment time (デプロイ時間)] に、デプロイの合計期間を分または時間単位で入力します。
9. を使用する場合ベイク時間で、Amazon のモニタリングにかかる合計時間を分または時間単位で入力します。 CloudWatch がデプロイの次のステップに進む前、またはデプロイを完了とみなす前にアラームを鳴らします。
10. [Tags (タグ)] セクションで、キーとオプションの値を入力します。1 つのリソースに対して最大 50 個のタグを指定できます。
11. [Create deployment strategy (デプロイ戦略の作成)] を選択します。

    Important

    の設定プロファイルを作成した場合AWS CodePipelineでパイプラインを作成する必要があります。 CodePipeline 指定するAWS AppConfigとしてプロバイダをデプロイする。ステップ 5: 設定のデプロイ (p. 44) を実行する必要はありません。ただし、「ステップ 6: 設定の取得 (p. 48)」で説明されているように、アプリケーション設定の更新を受け取るようにクライアントを設定する必要があります。を指定するパイプラインを作成する方法については、AWS AppConfigデプロイプロバイダーとして、を参照してください。チュートリアル: を使用するパイプラインを作成するAWS AppConfigデプロイプロバイダーとしてのAWS CodePipelineユーザーガイド。

ステップ 5: 設定のデプロイ (p. 44) に進みます。

AWS AppConfig デプロイ戦略の作成 (コマンドライン)

以下の手順では、AWS CLI (Linux または Windows の場合) または AWS Tools for PowerShell を使用して AWS AppConfig デプロイ戦略を作成する方法について説明します。

デプロイ戦略をステップバイステップで作成するには

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。
2. 以下のコマンドを実行して、デプロイ戦略を作成します。

   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-APPCDeploymentStrategy `
  --Name A_name_for_the_deployment_strategy `
  --Description A_description_of_the_deployment_strategy `
  --DeploymentDurationInMinutes Total_amount_of_time_for_a_deployment_to_last `
  --FinalBakeTimeInMinutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete `
  --
GrowthFactor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_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
```

システムが以下のような情報を返します。

Linux

```
{
   "Id": "Id of the deployment strategy",
   "Name": "Name of the deployment strategy",
   "Description": "Description of the deployment strategy",
   "DeploymentDurationInMinutes": "Total amount of time the deployment lasted",
   "GrowthType": "The linear or exponential algorithm used to define how percentage
 grew over time",
   "GrowthFactor": "The percentage of targets that received a deployed
 configuration during each interval",
```

```
   "FinalBakeTimeInMinutes": "The amount of time AWS AppConfig monitored for alarms
 before considering the deployment to be complete",
   "ReplicateTo": "The Systems Manager (SSM) document where the deployment strategy
 is saved"
}
```

Windows

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

PowerShell

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

ステップ 5: 設定のデプロイ

AWS AppConfig でデプロイを開始すると、StartDeployment API アクションが呼び出されます。この呼び出しには、デプロイする AWS AppConfig アプリケーションの ID、環境、設定プロファイル、および設定データバージョン (オプション) が含まれます。この呼び出しには、使用するデプロイ戦略の ID も含まれます。ID は、設定データのデプロイ方法を決定します。

AWS AppConfig はすべてのホストへの配布をモニタリングし、ステータスをレポートします。ディストリビューションが失敗した場合は、AWS AppConfig は設定をロールバックします。

Note

環境には、一度に 1 つの設定のみデプロイできます。ただし、1 つの設定をそれぞれ異なる環境に同時にデプロイすることができます。

設定をデプロイする (コンソール)

AWS Systems Manager コンソールを使用して AWS AppConfig 設定をデプロイするには、次の手順に従います。

コンソールを使用して設定をデプロイするには

1. AWS Systems Manager コンソール (https://console.aws.amazon.com/systems-manager/appconfig/) を開きます。
2. ナビゲーションペインで、[AWS AppConfig] を選択します。
3. [Applications (アプリケーション)] タブでアプリケーションを選択し、[View details (詳細の表示)] を選択します。
4. [Environments (環境)] タブで環境を選択し、[View details (詳細の表示)] を選択します。
5. [Start deployment (デプロイの開始)] を選択します。
6. [Configuration (設定)] で、リストから設定を選択します。
7. 設定のソースに応じて、[Document version (ドキュメントのバージョン)] リストまたは [Parameter version (パラメータのバージョン)] リストを使用して、デプロイするバージョンを選択します。
8. [Deployment strategy (デプロイ戦略)] で、リストから戦略を選択します。
9. [Deployment description (デプロイの説明)] に、説明を入力します。
10. [Tags (タグ)] セクションで、キーとオプションの値を入力します。1 つのリソースに対して最大 50 個のタグを指定できます。
11. [Start deployment (デプロイの開始)] を選択します。

設定をデプロイする (コマンドライン)

以下の手順では、AWS CLI (Linux または Windows の場合) または AWS Tools for PowerShell を使用して AWS AppConfig 設定をデプロイする方法について説明します。

設定をステップバイステップでデプロイする

1. まだ AWS CLI または AWS Tools for PowerShell をインストールして設定していない場合は、インストールして設定します。

   詳細については、「AWS コマンドラインツールをインストールまたはアップグレードする (p. 6)」を参照してください。
2. 次のコマンドを実行して、設定をデプロイします。

   Linux

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

   Windows

   ```
   aws appconfig start-deployment ^
     --application-id The_application_ID ^
     --environment-id The_environment_ID ^
     --deployment-strategy-id The_deployment_strategy_ID ^
   ```

```
  --configuration-profile-id The_configuration_profile_ID ^
  --configuration-version The_configuration_version_to_deploy ^
  --description A_description_of_the_deployment ^
  --tags User_defined_key_value_pair_metadata_of_the_deployment
```

PowerShell

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

システムが以下のような情報を返します。

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
```

ステップ 6: 設定の取得

アプリケーションは、まず設定セッションを確立して、設定データを取得します。設定セッションの開始API アクション。その後、セッションのクライアントは定期的にを呼び出します最新設定を取得をクリックして、利用可能な最新のデータを確認して取得します。

呼び出し時StartConfigurationSessionでは、コードは次の情報を送信します。

- の識別子 (ID または名前)AWS AppConfigセッションが追跡するアプリケーション、環境、設定プロファイル。
- (オプション) セッションのクライアントがへの呼び出し間で待機する必要のある最短時間GetLatestConfiguration。

それに応えて、AWS AppConfigを提供します。InitialConfigurationTokenセッションのクライアントに与えられ、初めて呼び出すときに使用されます。GetLatestConfigurationそのセッションのために。

> Important
>
> このトークンは、への最初の呼び出しで一度だけ使用してください。GetLatestConfiguration。お客様しなければならない新しいトークンをGetLatestConfigurationレスポンス (NextPollConfigurationToken) 以降の各呼び出しでGetLatestConfiguration。

呼び出し時GetLatestConfigurationの場合、クライアントコードは最新のものを送信しますConfigurationTokenその値が受け取り、それに応答して受け取る値:

- NextPollConfigurationToken:ConfigurationToken次の呼び出しで使用する値GetLatestConfiguration。
- NextPollIntervalInSeconds: クライアントが次の呼び出しを行う前に待機する時間GetLatestConfiguration。この期間は、セッション中に異なる場合があるため、で送信される値の代わりに使用する必要があります。StartConfigurationSessionを呼び出します。
- 構成:セッションを対象とした最新のデータ。クライアントにすでに最新バージョンの設定がある場合、これは空になることがあります。

> Important
>
> 以下の重要な情報に注意してください。
>
> - StartConfigurationSession API は、サービスとのセッションを確立するために、アプリケーション、環境、設定プロファイル、およびクライアントごとに 1 回のみ呼び出す必要があります。これは、通常、アプリケーションの起動時または設定の初回取得の直前に行われます。
> - -InitialConfigurationTokenそしてNextPollConfigurationToken24 時間後に期限切れになります。もしあればGetLatestConfigurationコールは期限切れのトークンを使用し、システムは戻りますBadRequestException。

設定例の取得

以下のようになりますAWS CLI例は、を使用して設定データを取得する方法を示しています。AWS AppConfig データStartConfigurationSessionそしてGetLatestConfigurationAPI アクション。最初のコマンドは、設定セッションを開始します。この呼び出しには、AWS AppConfig アプリケーションの ID (または名前)、環境、および設定プロファイルが含まれます。API は、設定データをフェッチするために使用される InitialConfigurationToken を返します。

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

システムから以下の形式の情報で応答します。

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

セッションを開始したら、設定データをフェッチするために、InitialConfigurationToken を使用して GetLatestConfiguration を呼び出します。設定データは、mydata.json ファイルに保存されます。

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

GetLatestConfiguration への初回呼び出しでは、StartConfigurationSession から取得された ConfigurationToken を使用します。次の情報が返されます。

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

以降のへの呼び出しGetLatestConfiguration しなければならない提供NextPollConfigurationToken前の応答から。

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

Important

GetLatestConfiguration API アクションに関する以下の重要な詳細に留意してください。

- GetLatestConfiguration 応答には、設定データを示す Configuration セクションが含まれています。Configuration セクションは、新規または更新された設定データをシステムが検出した場合にのみ表示されます。新規または更新された設定データをシステムが検出しない場合、Configuration データは空です。
- GetLatestConfiguration からの応答があるたびに新しい ConfigurationToken を受け取ります。
- 予算、予想頻度、設定のターゲット数に基づいて、GetLatestConfiguration API コールのポーリング頻度を調整することをお勧めします。

AWS AppConfig と統合するサービス

AWS AppConfig と他の AWS サービスを統合すると、アプリケーション設定を高速かつ安全に管理できます。以下の情報は、使用する製品やサービスを統合するための AWS AppConfig の設定に役立ちます。

目次

- AWS AppConfig と Lambda 拡張機能の統合 (p. 50)
- AWS AppConfigアトラシアン Jira との統合 (p. 67)
- AWS AppConfig と CodePipeline の統合 (p. 71)

AWS AppConfig と Lambda 拡張機能の統合

AWS Lambda 拡張機能は、Lambda 関数の機能を強化するコンパニオンプロセスです。拡張機能は、関数が呼び出される前に開始し、関数と並行して実行し、関数の呼び出しが処理された後も引き続き実行できます。本質的に、Lambda 拡張機能は Lambda 呼び出しと並行して実行されるクライアントのようなものです。この並列クライアントは、ライフサイクル中の任意の時点で関数とインターフェイスできます。

使用するものAWS AppConfigLambda 関数内の機能フラグまたはその他の動的設定データを使用する場合は、AWS AppConfigLambda 拡張機能を Lambda 関数のレイヤーとして指定します。これにより、機能フラグの呼び出しが簡単になり、拡張機能自体には、の使用を簡素化するベストプラクティスが含まれています。AWS AppConfigコストを削減しながら。への API 呼び出しが少なくなると、コストが削減されます。AWS AppConfigサービスと Lambda 関数の処理時間が短くなります。Lambda 拡張機能の詳細については、「AWS Lambda デベロッパーガイド」の「Lambda 拡張機能」を参照してください。

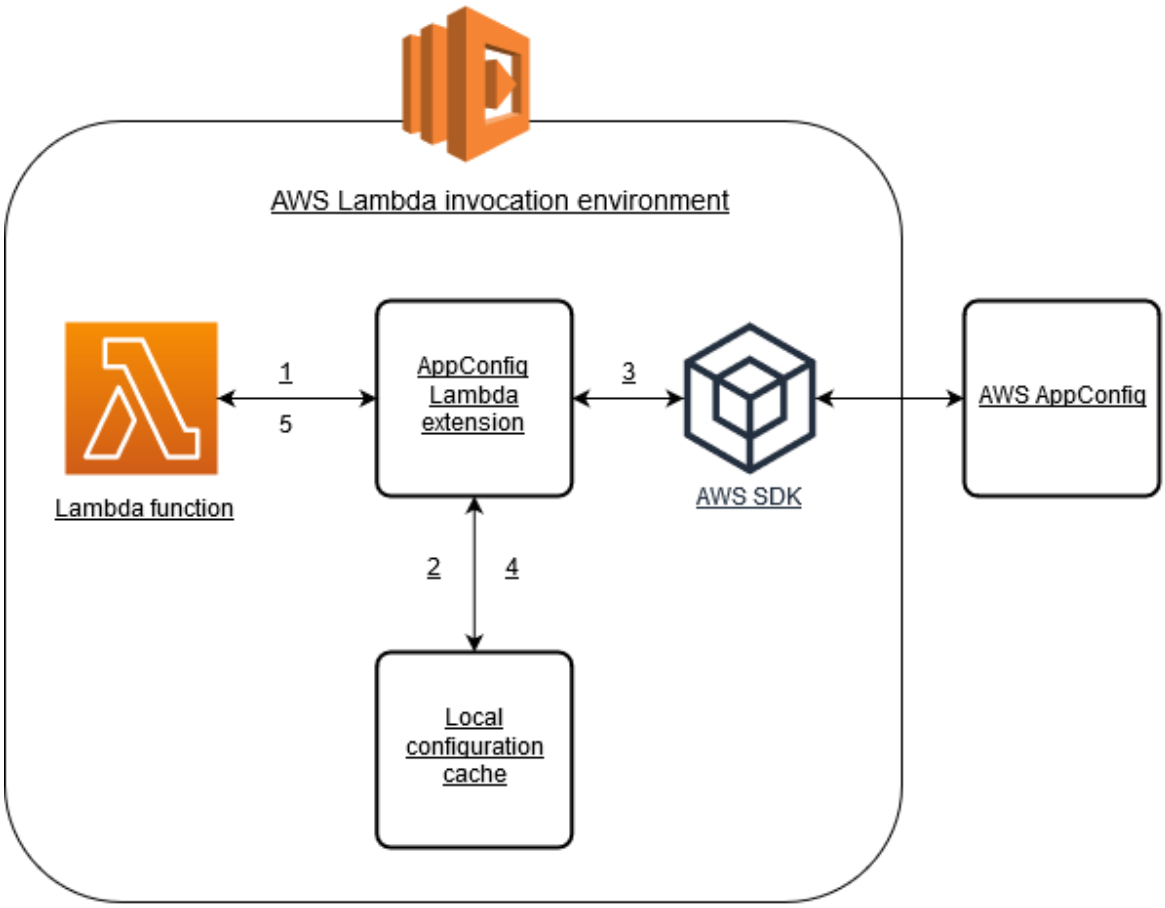
> Note
>
> AWS AppConfig 価格は、コンフィギュレーションが呼び出されて受信された回数に基づきます。Lambda が複数のコールドスタートを実行し、新しい構成データを頻繁に取得すると、コストが増加します。

このトピックには、AWS AppConfig Lambda 拡張機能の詳細、および Lambda 関数と連携できるように拡張機能を設定する方法の手順が含まれています。

仕組み

使用するものAWS AppConfigLambda 関数の設定を管理するにはなしLambda 拡張機能を使用する場合は、Lambda 関数と統合して、Lambda 関数が設定の更新を受け取るように設定する必要があります。設定セッションの開始そして最新設定を取得API アクション。

AWS AppConfig Lambda 拡張機能と Lambda 関数を統合することで、このプロセスが簡素化されます。拡張機能は、AWS AppConfigサービス、取得したデータのローカルキャッシュの管理、次のサービスコールに必要な設定トークンの追跡、バックグラウンドでの設定の更新の定期的なチェックを行います。次の図は、その仕組みを示しています。



1. AWS AppConfig Lambda 拡張機能を Lambda 関数のレイヤーとして設定します。
2. 設定データにアクセスするには、関数がAWS AppConfigで実行されている HTTP エンドポイントでの拡張`localhost:2772`。
3. 拡張機能は、設定データのローカルキャッシュを保持します。データがキャッシュにない場合、拡張機能は AWS AppConfig を呼び出して、設定データを取得します。
4. サービスから設定を受信すると、拡張機能はローカルキャッシュに設定を保存し、Lambda 関数に渡します。
5. AWS AppConfig Lambda 拡張機能は、設定データの更新をバックグラウンドで定期的にチェックします。Lambda 関数が呼び出されるたびに、拡張機能は、設定を取得してからの経過時間をチェックします。設定されたポーリング間隔よりも経過時間が長い場合、拡張機能は AWS AppConfig を呼び出して、新しくデプロイされたデータをチェックし、変更がある場合はローカルキャッシュを更新して、経過時間をリセットします。

Note

- Lambda は、関数が必要とする同時実行レベルに対応する個別のインスタンスをインスタンス化します。各インスタンスは分離され、設定データの独自のローカルキャッシュが保持されます。Lambda インスタンスと同時実行の詳細については、「Lambda 関数の同時実行数の管理」を参照してください。
- AWS AppConfig から更新された設定をデプロイしてから Lambda 関数に設定の変更が表示されるまでにかかる時間は、デプロイに使用したデプロイ戦略、および拡張機能に設定したポーリング間隔によって異なります。

始める前に

AWS AppConfig Lambda 拡張機能を有効にする前に、以下を実行します。

- 設定を AWS AppConfig に外部化できるように、Lambda 関数での設定を整理します。
- 設定の更新を管理するように、AWS AppConfig を設定します。詳細については、「AWS AppConfigの使用 (p. 12)」を参照してください。
- を追加します。`appconfig:StartConfigurationSession`そして`appconfig:GetLatestConfiguration`にAWS Identity and Access ManagementLambda 関数の実行ロールによって使用される (IAM) ポリシー。詳細については、「AWS Lambda デベロッパーガイド」の「AWS Lambda 実行ロール」を参照してください。の詳細AWS AppConfig権限、「」を参照してください。アクション、リソース、条件キー:AWS AppConfigのサービス認証リファレンス。

ランタイムのサポート

-AWS AppConfigLambda 拡張機能は、次のランタイムをサポートします。

- Python 3.7, 3.8, 3.9
- Node.js 12.x、14.x
- Ruby 2.7
- Java 8、11 (Amazon Corretto)
- .NET Core 3.1
- カスタムランタイム (Amazon Linux および Amazon Linux 2)

対応する SDK、オペレーティングシステム、アーキテクチャなど、これらのランタイムの詳細については、を参照してください。Lambda ランタイムのAWS Lambdaデベロッパーガイド。

AWS AppConfig Lambda 拡張機能を追加する

AWS AppConfig Lambda 拡張機能を使用するには、拡張機能を Lambda に追加する必要があります。これには、AWS AppConfig Lambda 拡張機能を Lambda 関数にレイヤーとして追加するか、または Lambda 関数における拡張機能をコンテナイメージとして有効にします。

[AWS AppConfigレイヤーと ARN を使用した Lambda 拡張

AWS AppConfig Lambda 拡張機能を使用するには、拡張機能を Lambda 関数にレイヤーとして追加する必要があります。関数にレイヤーを追加する方法については、「AWS Lambdaデベロッパーガイド」の「拡張機能の設定」を参照してください。AWS Lambda コンソールでの拡張機能の名前は AWS-AppConfig-Extension です。また、拡張機能を Lambda にレイヤーとして追加する場合は、Amazon リソースネーム (ARN) を指定する必要があることにも注意してください。プラットフォームに対応する ARN を次のいずれかのリストから選択し、AWS リージョンLambda を作成した場所。

- x86-64 プラットフォーム (p. 56)
- ARM64 プラットフォーム (p. 58)

拡張機能を関数に追加する前にテストする場合は、次のコード例を使用して拡張機能が機能することを確認できます。

```
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
```

これをテストするには、Python 用の新しい Lambda 関数を作成し、拡張機能を追加して Lambda 関数を実行します。Lambda 関数を実行すると、指定した設定が AWS AppConfig Lambda 関数によって http://localhost:2772 パスに返されます。Lambda 関数の作成については、「AWS Lambda デベロッパーガイド」の「コンソールで Lambda の関数の作成」を参照してください。

AWS AppConfig Lambda 拡張機能をコンテナイメージとして作成する場合は、「コンテナイメージを使用して AWS AppConfig Lambda 拡張機能を追加する (p. 63)」を参照してください。

設定:AWS AppConfigLambda 拡張

拡張機能を設定するには、次の AWS Lambda 環境変数を変更します。詳細については、「AWS Lambda デベロッパーガイド」の「AWS Lambda 環境変数の使用」を参照してください。

設定データのプリフェッチ

環境変数`AWS_APPCONFIG_EXTENSION_PREFETCH_LIST`関数の起動時間を改善できます。AWS AppConfig Lambda 拡張機能が初期化されると、Lambda が関数を初期化してハンドラーを呼び出す前に、指定された設定が AWS AppConfig から取得されます。場合によっては、関数が要求する前に、設定データがローカルキャッシュで既に利用可能になっていることがあります。

プリフェッチ機能を使用するには、設定データに対応するパスに、環境変数の値を設定します。例えば、設定がそれぞれ「my_application」、「my_environment」、「my_configuration_data」という名前のアプリケーション、環境、および設定プロファイルに対応している場合、パスは `/applications/my_application/environments/my_environment/configurations/my_configuration_data` のようになります。設定項目をカンマ区切りのリストとして列挙することで、複数の設定項目を指定できます (カンマを含むリソース名がある場合は、リソースの名前ではなく ID 値を使用します)。

別のアカウントから設定データにアクセスする

-AWS AppConfigLambda 拡張機能は、付与する IAM ロールを指定することで、別のアカウントから設定データを取得できます。許可データに。これを設定するには、以下の手順に従います。

1. アカウント内のどこAWS AppConfigは、設定データを管理し、Lambda 関数を実行しているアカウントに`appconfig:StartConfigurationSession`そして`appconfig:GetLatestConfiguration`アクション、および対応する部分的または完全な ARNAWS AppConfig設定リソース。
2. Lambda 関数を実行しているアカウントで、`AWS_APPCONFIG_EXTENSION_ROLE_ARN`Lambda 関数の環境変数を、ステップ 1 で作成したロールの ARN を使用します。
3. (オプション) 必要に応じて、外部 IDを使用して指定できる。`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`環境変数。同様に、セッション名は`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`環境変数。

   Note

   以下の情報に注意してください。

   - -AWS AppConfigLambda 拡張機能は 1 つのアカウントからのみデータを取得できます。IAM ロールを指定した場合、エクステンションは Lambda 関数が実行されているアカウントから設定データを取得できません。
   - AWS Lambdaについての情報をログに記録します。AWS AppConfigAmazon を使用した Lambda 拡張機能と Lambda 関数 CloudWatch ログ。

環境変数	詳細	デフォルト値
AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS	この環境変数は、更新された設定の AWS AppConfig に拡張機能がポーリングする頻度を秒単位で制御します。	45
AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS	この環境変数は、キャッシュ内のデータを更新するときに AWS AppConfig からの応答を待機する最大時間 (ミリ秒単位) を制御します。指定された時間内に AWS AppConfig が応答しない場合、拡張機能はこのポーリング間隔をスキップし、以前に更新されたキャッシュデータを返します。	3000
AWS_APPCONFIG_EXTENSION_HTTP_PORT	この環境変数は、拡張機能をホストするローカル HTTP サーバーが実行されるポートを指定します。	2772
AWS_APPCONFIG_EXTENSION_PREFETCH_LIST	この環境変数は、関数が初期化されてハンドラーが実行される前に、拡張機能が取得を開始する設定データを指定します。これにより、機能のコールドスタート時間を大幅に短縮できます。	[None] (なし)
AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS	この環境変数は、拡張機能が AWS AppConfig から設定を取得するために使用する最大接続数を設定します。	3
AWS_APPCONFIG_EXTENSION_LOG_LEVEL	この環境変数は、どれを指定します。AWS AppConfigAmazon に拡張機能固有のログが送信されます CloudWatch 関数のログを記録します。大文字と小文字を区別しない有効な値は debug、info、warn、error、および none です。デバッグには、タイミング情報など、拡張機能に関する詳細情報が含まれます。	info
AWS_APPCONFIG_EXTENSION_ROLE_ARN	この環境変数は、で引き受けるべきロールに対応する IAM ロール ARN を指定します。AWS AppConfig設定を取得するための拡張機能。	[None] (なし)
AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID	この環境変数は、引き受けたロール ARN と組み合わせて使用する外部 ID を指定します。	[None] (なし)

環境変数	詳細	デフォルト値
AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME	この環境変数は、引き受ける IAM ロールの認証情報に関連付けるセッション名を指定します。	[None] (なし)
AWS_APPCONFIG_EXTENSION_SERVICE_ENDPOINT	この環境変数は、拡張機能が AWS AppConfigサービス。未定義の場合、拡張機能は現在のリージョンのエンドポイントを使用します。	[None] (なし)

機能フラグ設定から 1 つ以上のフラグを取得する

フィーチャフラグコンフィギュレーション (タイプのコンフィギュレーション) の場合AWS.AppConfig.FeatureFlags) の場合、Lambda 拡張機能を使用すると、設定内の単一のフラグまたはフラグのサブセットを取得できます。1 つまたは 2 つのフラグを取得すると、Lambda が設定プロファイルからいくつかのフラグのみを使用する必要がある場合に便利です。以下の例では Python を使用しています。

> **Note**
>
> コンフィギュレーション内の 1 つの機能フラグまたはフラグのサブセットを呼び出す機能は、AWS AppConfigLambda 拡張バージョン 2.0.45 以上。

取得できますAWS AppConfigローカル HTTP エンドポイントからの設定データ。特定のフラグまたはフラグのリストにアクセスするには、`?flag=`*`flag_name`*のクエリパラメータAWS AppConfig設定プロファイル。

1 つのフラグとその属性にアクセスするには

```
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
```

複数のフラグとその属性にアクセスするには

```
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
```

の使用可能なバージョンAWS AppConfigLambda 拡張

このトピックの一覧を示します。AWS AppConfigプロセッサによる Lambda 拡張バージョン。-AWS AppConfigLambda 拡張機能は、x86-64 および ARM64 (Graviton2) プラットフォーム用に開発された Lambda 関数をサポートしています。各テーブルには、それぞれで使用する Amazon リソースネーム (ARN) が含まれますAWS リージョン。

トピック

- x86-64 プラットフォーム (p. 56)
- ARM64 プラットフォーム (p. 58)
- 古い拡張機能バージョン (p. 59)

x86-64 プラットフォーム

拡張機能を Lambda にレイヤーとして追加する場合、ARN を指定する必要があります。次のテーブルから、に対応する ARN を選択します。AWS リージョンLambda を作成した場所。これらの ARN は x86-64 プラットフォーム用に開発された Lambda 関数用です。

バージョン2.0.58

リージョン	ARN
米国東部 (バージニア北部)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`
米国東部 (オハイオ)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50`
米国西部 (北カリフォルニア)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78`
米国西部 (オレゴン)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101`
カナダ(中部)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50`
欧州(フランクフルト)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59`
ヨーロッパ (アイルランド)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69`
欧州 (ロンドン)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50`
欧州 (パリ)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51`
欧州 (ストックホルム)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98`
ヨーロッパ (ミラノ)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47`

リージョン	ARN
中国 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46
中国 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46
アジアパシフィック (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47
アジアパシフィック (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49
アジアパシフィック (ソウル)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59
アジアパシフィック (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46
アジアパシフィック (シンガポール)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51
アジアパシフィック (シドニー)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59
アジアパシフィック (ジャカルタ)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24
アジアパシフィック (ムンバイ)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60
南米 (サンパウロ)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69
アフリカ (ケープタウン)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47
中東 (バーレーン)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47
AWS GovCloud (米国東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23

リージョン	ARN
AWS GovCloud (米国西部)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23`

ARM64 プラットフォーム

拡張機能を Lambda にレイヤーとして追加する場合、ARN を指定する必要があります。次のテーブルから、に対応する ARN を選択します。AWS リージョンLambda を作成した場所。これらの ARN は、ARM64 プラットフォーム用に開発された Lambda 関数用です。

バージョン2.0.58

リージョン	ARN
米国東部 (バージニア北部)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2`
米国東部 (オハイオ)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2`
米国西部 (オレゴン)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3`
欧州(フランクフルト)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2`
ヨーロッパ (アイルランド)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7`
欧州 (ロンドン)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2`
アジアパシフィック (東京)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2`
アジアパシフィック (シンガポール)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3`
アジアパシフィック (シドニー)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2`
アジアパシフィック (ムンバイ)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2`

古い拡張機能バージョン

このセクションでは、ARN およびAWS リージョン古いバージョンのAWS AppConfigLambda 拡張機能。このリストには、以前のバージョンのすべての情報が記載されているわけではありません。AWS AppConfigLambda 拡張機能。ただし、新しいバージョンがリリースされると更新されます。

古い拡張バージョン (x86-64 プラットフォーム)

次の表は ARN とAWS リージョン古いバージョンのAWS AppConfigx86-64 プラットフォーム用に開発された Lambda 拡張機能。

新しい拡張子で置き換えられた日付: 2022 年 4 月 21 日

バージョン2.0.45

リージョン	ARN
米国東部 (バージニア北部)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68`
米国東部 (オハイオ)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49`
米国西部 (北カリフォルニア)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77`
米国西部 (オレゴン)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100`
カナダ(中部)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49`
欧州(フランクフルト)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58`
ヨーロッパ (アイルランド)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68`
欧州 (ロンドン)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49`
欧州 (パリ)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50`
欧州 (ストックホルム)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97`
ヨーロッパ (ミラノ)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46`

リージョン	ARN
中国 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45
中国 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45
アジアパシフィック (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46
アジアパシフィック (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48
アジアパシフィック (ソウル)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58
アジアパシフィック (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45
アジアパシフィック (シンガポール)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50
アジアパシフィック (シドニー)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58
アジアパシフィック (ジャカルタ)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23
アジアパシフィック (ムンバイ)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59
南米 (サンパウロ)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68
アフリカ (ケープタウン)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46
中東 (バーレーン)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46
AWS GovCloud (米国東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22

リージョン	ARN
AWS GovCloud (米国西部)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22`

新しい拡張子で置き換えられた日付: 2022年03月15日

バージョン2.0.30

リージョン	ARN
米国東部 (バージニア北部)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61`
米国東部 (オハイオ)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47`
米国西部 (北カリフォルニア)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61`
米国西部 (オレゴン)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89`
カナダ(中部)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47`
欧州(フランクフルト)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54`
ヨーロッパ (アイルランド)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59`
欧州 (ロンドン)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47`
欧州 (パリ)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48`
欧州 (ストックホルム)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86`
ヨーロッパ (ミラノ)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44`
中国 (北京)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43`

リージョン	ARN
中国 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43
アジアパシフィック (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44
アジアパシフィック (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45
アジアパシフィック (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42
アジアパシフィック (ソウル)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54
アジアパシフィック (シンガポール)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45
アジアパシフィック (シドニー)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54
アジアパシフィック (ジャカルタ)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13
アジアパシフィック (ムンバイ)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55
南米 (サンパウロ)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61
アフリカ (ケープタウン)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44
中東 (バーレーン)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44
AWS GovCloud (米国東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20
AWS GovCloud (米国西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20

古い拡張バージョン (ARM64 プラットフォーム)

次の表は ARN とAWS リージョン古いバージョンのAWS AppConfigARM64 プラットフォーム用に開発された Lambda 拡張機能。

新しい拡張子で置き換えられた日付: 2022 年 4 月 21 日

バージョン2.0.45

リージョン	ARN
米国東部 (バージニア北部)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1`
米国東部 (オハイオ)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1`
米国西部 (オレゴン)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2`
欧州(フランクフルト)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1`
ヨーロッパ (アイルランド)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6`
欧州 (ロンドン)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1`
アジアパシフィック (東京)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1`
アジアパシフィック (シンガポール)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2`
アジアパシフィック (シドニー)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1`
アジアパシフィック (ムンバイ)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1`

コンテナイメージを使用して AWS AppConfig Lambda 拡張機能を追加する

AWS AppConfig Lambda 拡張機能をコンテナイメージとしてパッケージして、Amazon Elastic Container Registry (Amazon ECR) でホストされているコンテナレジストリにアップロードすることができます。

AWS AppConfig Lambda 拡張機能を Lambda コンテナイメージとして追加するには

1. 以下に示すように、AWS リージョン および Amazon リソースネーム (ARN) を AWS Command Line Interface (AWS CLI) に入力します。リージョンと ARN の値をご使用のリージョンと一致する ARN に置き換えて、Lambda レイヤーのコピーを取得します。

```
aws lambda get-layer-version-by-arn \
  --region us-east-1 \
  --arn arn:aws:lambda::layer:AWS-AppConfig-Extension:00
```

以下は、レスポンスです。

```
{
  "LayerVersionArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension:00",
  "Description": "AWS AppConfig extension: Use dynamic configurations deployed via AWS
 AppConfig for your AWS Lambda functions",
  "CreatedDate": "2021-04-01T02:37:55.339+0000",
  "LayerArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension",

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

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

2. 上記の応答では、`Location` に対して返された値は、Lambda 拡張機能を含む Amazon Simple Storage Service (Amazon S3) バケットの URL です。ウェブブラウザに URL を貼り付けて、Lambda 拡張機能の.zip ファイルをダウンロードします。

   Note

   Amazon S3 バケット URL は 10 分間のみ有効です。

   (オプション) または、次の `curl` コマンドを使用して、Lambda 拡張機能をダウンロードすることもできます。

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

(オプション) または、ステップ 1 とステップ 2 を組み合わせて、ARN の取得および拡張機能の .zip ファイルのダウンロードを一度に済ませることもできます。

```
aws lambda get-layer-version-by-arn \
  --arn arn:aws:lambda::layer:AWS-AppConfig-Extension:00 \
  | jq -r '.Content.Location' \
  | xargs curl -o extension.zip
```

3. 次の行を `Dockerfile` に追加して、コンテナイメージに拡張機能を追加します。

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

```
  && rm -f extension.zip
```

4. Lambda 関数の実行ロールには必ず、appconfig:GetConfiguration アクセス許可が設定されているようにしてください。

例

このセクションでは、コンテナイメージベースの Python Lambda 関数で AWS AppConfig Lambda 拡張機能を有効にする例を示します。

1. 次のような `Dockerfile` を作成します。

```
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. 拡張機能レイヤーを、`Dockerfile` と同じディレクトリにダウンロードします。

```
aws lambda get-layer-version-by-arn \
  --arn arn:aws:lambda::layer:AWS-AppConfig-Extension:00 \
  | jq -r '.Content.Location' \
  | xargs curl -o extension.zip
```

3. `index.py` という名前の Python ファイルを、`Dockerfile` と同じディレクトリに作成します。

```
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. 次のステップを実行して `docker` イメージを構築し、それを Amazon ECR にアップロードします。

```
// 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. 先ほど作成した Amazon ECR イメージを使用して、Lambda 関数を作成します。コンテナとしての Lambda 関数の詳細については、「コンテナイメージとして定義された Lambda 関数を作成する」を参照してください。
6. Lambda 関数の実行ロールには必ず、appconfig:GetConfiguration アクセス許可が設定されているようにしてください。

OpenAPI と統合する

OpenAPI の次の YAML 仕様を使用して、OpenAPI Generator のようなツールを使って SDK を作成できます。この仕様を更新して、アプリケーション、環境、または設定用のハードコーディングされた値を含めることができます。また、パスを追加して (複数の設定タイプがある場合)、設定スキーマを含めて、SDK クライアント用の設定固有の型付けされたモデルを生成することもできます。OpenAPI (Swagger とも呼ばれる) の詳細については、「OpenAPI の仕様」を参照してください。

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

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

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

AWS AppConfigアトラシアン Jira との統合

アトラシアン Jira との統合によりAWS AppConfigに変更を加えるたびに Atlassian コンソールで課題を作成して更新するには機能フラグのAWS アカウント指定されたAWS リージョン。各 Jira 課題には、フラグ名、アプリケーション ID、設定プロファイル ID、フラグ値が含まれます。フラグの変更を更新、保存、デプロイすると、Jira は変更の詳細で既存の課題を更新します。

Note

Jira は、機能フラグを作成または更新するたびに課題を更新します。Jira は、親レベルのフラグから子レベルのフラグ属性を削除すると、課題も更新されます。親レベルのフラグを削除しても、Jira は情報を記録しません。

統合を設定するには、以下の手順を実行する必要があります。

- アクセス許可の設定

- 統合の設定

のアクセス権限の設定AWS AppConfigJira 統合

構成するときAWS AppConfigJira との統合では、AWS Identity and Access Management(IAM) ユーザー。具体的には、ユーザーのアクセスキー ID とシークレットキーをAWS AppConfigJira アプリケーション用。このユーザーは Jira との通信権限を付与しますAWS AppConfig。AWS AppConfigは、これらのクレデンシャルを 1 回使用して、間の関連付けを確立します。AWS AppConfigJira。認証情報は保存されません。関連付けを削除するには、AWS AppConfigJira アプリケーション用。

IAM ユーザーアカウントには、以下のアクションを含むアクセス権限ポリシーが必要です。

- `appconfig:AssociateExtension`
- `appconfig:GetConfigurationProfile`
- `appconfig:ListApplications`
- `appconfig:ListConfigurationProfiles`
- `appconfig:UpdateConfigurationProfile`
- `sts:GetCallerIdentity`

次のタスクを実行して、の IAM アクセス権限ポリシーと IAM ユーザーを作成します。AWS AppConfigそして Jira 統合:

[Tasks] (タスク)

- タスク 1: の IAM アクセス権限ポリシーを作成するAWS AppConfigJira 統合機能 (p. 68)
- タスク 2: の IAM ユーザーを作成するAWS AppConfigJira 統合機能 (p. 69)

タスク 1: の IAM アクセス権限ポリシーを作成するAWS AppConfigJira 統合機能

Atlassian Jira がとの通信を許可する IAM アクセス許可ポリシーを作成するには、以下の手順を使用します。AWS AppConfig。新しいポリシーを作成し、このポリシーを新しい IAM ロールにアタッチすることをお勧めします。既存の IAM ポリシーおよびロールに必要なアクセス権限を追加することは、最小権限の原則に反するため、推奨されません。

の IAM ポリシーを作成するにはAWS AppConfigJira 統合機能

1. https://console.aws.amazon.com/iam/ で IAM コンソール を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択してから [Create policy] (ポリシーの作成) を選択します。
3. リポジトリの []ポリシーの作成[] ページで、JSONタブをクリックし、デフォルトコンテンツを以下のポリシーに置き換えます。次のポリシーで、*#####*,*ACCOUNT_ID*,*application_ID*, および*configuration_profile_ID*からの情報でAWS AppConfig機能フラグ環境。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                        "appconfig:AssociateExtension",
                        "appconfig:GetConfigurationProfile",
                        "appconfig:UpdateConfigurationProfile"

             ],
            "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. [Next: (次へ:)] を選択します タグ
5. (オプション) 1 つ以上のタグキーと値のペアを追加して、このポリシーのアクセスを整理、追跡、または制御します。次へ: 確認.
6. [Review policy] (ポリシーの確認) ページの [Name] (名前) ボックスに **`AppConfigJiraPolicy`** などの名前を入力し、説明を入力します。
7. [Create policy] (ポリシーの作成) を選択します。

タスク 2: の IAM ユーザーを作成するAWS AppConfigJira 統合機能

以下の手順に従って IAM ユーザーを作成します。AWS AppConfigアトラシアン Jira との統合。ユーザーを作成した後、アクセスキー ID とシークレットキーをコピーできます。このシークレットキーは、統合の完了時に指定します。

の IAM ユーザーを作成するにはAWS AppConfigJira 統合機能

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、[] を選択します。Users[] を選択してから、ユーザーの追加。
3. 左[User name]フィールドに、次のような名前を入力します。**`AppConfigJiraUser`**。
4. を使用する場合SelectAWS認証情報のタイプで、アクセスキー-プログラムによるアクセス。
5. [Next: (次へ:)] を選択します アクセス許可.
6. []許可を設定[] ページで、[]既存のポリシーを直接アタッチする。で作成したポリシーを検索し、選択します。タスク 1: の IAM アクセス権限ポリシーを作成するAWS AppConfigJira 統合機能 (p. 68)[] を選択してから、次へ: タグ
7. リポジトリの []タグの追加 (オプション)ページで、1 つ以上のタグキーと値のペアを追加して、このユーザーのアクセスを整理、追跡、または制御します。[Next: (次へ:)] を選択します 確認.
8. リポジトリの []確認ページで、ユーザーの詳細を確認します。
9. [Create user] (ユーザーの作成) を選択します。システムは、ユーザーのアクセスキー ID とシークレットキーを表示します。.csv ファイルをダウンロードするか、これらの認証情報を別の場所にコピーします。統合を設定するときに、これらの認証情報を指定します。

設定:AWS AppConfigJira 統合アプリケーション

で必要なオプションを設定するには、以下の手順に従います。AWS AppConfigJira アプリケーション用。この手順を完了すると、Jira は、の機能フラグごとに新しい課題を作成します。AWS アカウント指定されたAWS リージョン。でフィーチャフラグを変更した場合AWS AppConfigの場合、Jira は既存の課題の詳細を記録します。

> Note
>
> AnAWS AppConfigフィーチャフラグには、複数の子レベルのフラグ属性を含めることができます。Jira は親レベルの機能フラグごとに 1 つの課題を作成します。子レベルのフラグ属性を変更すると、親レベルフラグの Jira 課題でその変更の詳細を表示できます。

統合を設定するには

1. にログインします。アトラシアンMarketplace。
2. タイプ**`AWS AppConfig`**検索フィールドで、[] を押します。Enter。
3. アプリケーションを Jira インスタンスにインストールします。
4. Atlassian コンソールで、[] を選択します。アプリの管理[] を選択してから、AWS AppConfigJira。
5. [Configure] (設定) を選択します。
6. []設定の詳細で、Jira プロジェクト次に、に関連付けるプロジェクトを選択します。AWS AppConfig機能フラグ。
7. 選択AWS リージョンを選択し、[Region (地域)] を選択します。AWS AppConfig機能フラグが配置されています。
8. 左アプリケーション IDフィールドに、名前を入力します。AWS AppConfig機能フラグを含むアプリケーション。
9. 左設定プロファイル IDフィールドに、名前を入力します。AWS AppConfig機能フラグの設定プロファイル。
10. 左アクセスキー IDそしてシークレットキーフィールドに、コピーした認証情報を入力します。タスク 2: の IAM ユーザーを作成するAWS AppConfigJira 統合機能 (p. 69)。オプションで、セッショントークンを指定することもできます。
11. [Submit] (送信) を選択します。
12. Atlassian コンソールで、[] を選択します。プロジェクトを選択し、選択したプロジェクトを選択します。AWS AppConfigの統合。-問題点ページには、指定された機能フラグごとに課題が表示されます。AWS アカウントそしてAWS リージョン。

の削除AWS AppConfigJira アプリケーションとデータ用

で Jira 統合を使用する必要がなくなった場合AWS AppConfig機能フラグ、削除できますAWS AppConfigアトラシアンコンソールの Jira アプリケーションの場合。統合アプリケーションを削除すると、以下の操作が実行されます。

- Jira インスタンスとの関連付けを削除します。AWS AppConfig。
- Jira インスタンスの詳細をから削除します。AWS AppConfig。

を削除するにはAWS AppConfigJira アプリケーション用

1. Atlassian コンソールで、[] を選択します。アプリの管理。
2. 選択AWS AppConfigJira。
3. [Uninstall] (アンインストール) を選択します。

AWS AppConfig と CodePipeline の統合

AWS AppConfigは、の統合デプロイアクションですかAWS CodePipeline(CodePipeline)。 CodePipeline はフルマネージド型の継続的デリバリーサービスで、アプリケーションとインフラストラクチャの更新を迅速かつ高い信頼性で行うために、パイプラインのリリースを自動化します。 CodePipeline は、お客様が定義したリリースモデルに基づいて、コードチェンジがあった場合のフェーズの構築、テスト、およびデプロイを自動化します。詳細については、「AWS CodePipeline とは?」を参照してください。

の統合AWS AppConfigと CodePipeline には次のような利点があります。

- ご利用のお客様 CodePipeline オーケストレーションを管理するために、コードベース全体をデプロイすることなく、アプリケーションに設定変更をデプロイする軽量な手段が利用できるようになりました。
- 利用したいお客様AWS AppConfig構成展開を管理するが、制限があるために制限されるAWS AppConfigは現在のコードまたは設定ストアをサポートしていません。オプションが追加されました。CodePipeline サポートするAWS CodeCommit、GitHub、 BitBucket (いくつか例を挙げると)。

> **Note**
>
> AWS AppConfigと の統合 CodePipeline はでのみサポートされていますAWS リージョンどこCodePipeline です使用可能です。

統合の仕組み

まず、CodePipeline をセットアップして設定します。これには、CodePipeline がサポートするコードストアに設定を追加することが含まれます。次に、以下のタスクを実行して、AWS AppConfig 環境をセットアップします。

- アプリケーションの作成
- 環境の作成
- AWS CodePipeline 設定プロファイルの作成
- 定義済みのデプロイ戦略の選択または独自のデプロイ戦略の作成

これらのタスクの完了後、でパイプラインを作成します。 CodePipeline 指定するAWS AppConfigとしてプロバイダをデプロイする。その後、設定を変更して、にアップロードします。 CodePipeline コード

ストア。新しい設定をアップロードすると、CodePipeline で新しいデプロイが自動的に開始されます。デプロイが完了したら、変更を確認できます。を指定するパイプラインを作成する方法については、AWS AppConfigデプロイプロバイダーとして、を参照してください。チュートリアル: を使用するパイプラインを作成するAWS AppConfigデプロイプロバイダーとしてのAWS CodePipelineユーザーガイド。

AWS AppConfigでのセキュリティ

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。責任共有モデル、、は、これをクラウドのセキュリティとクラウド内のセキュリティと説明しています:

- クラウドのセキュリティ - AWS は、AWS で AWS クラウド サービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。AWSコンプライアンスプログラム[g11]AWSコンプライアンスプログラム[/g11][g10]AWSコンプライアンスプログラム[/g10]の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。AWS Systems Manager に適用するコンプライアンスプログラムの詳細については、コンプライアンスプログラムによる対象範囲内の AWS のサービスをご参照ください。
- クラウド内のセキュリティ–お客様の責任は、使用するAWSのサービスに応じて判断されます。また、お客様は、データの機密性、お客様の会社の要件、および適用可能な法律および規制など、その他の要因についても責任を担います。

AWS Systems Manager は AWS AppConfig の一機能です。を使用する際に共有責任モデルを適用する方法を理解するにはAWS AppConfig「」を参照してください。セキュリティAWS Systems Manager。セキュリティおよびコンプライアンス目標を達成するために Systems Manager を設定する方法を説明しています。AWS AppConfig。

AWS AppConfig のモニタリング

モニタリングは、の信頼性、可用性、パフォーマンスを維持する上で重要な部分です。AWS解決策。AWS AppConfigの一機能ですAWS Systems Manager。Systems Manager の機能のモニタリングについては、「」を参照してください。MonitoringAWS Systems ManagerのAWS Systems Managerユーザーガイド。

AWS AppConfig ユーザーガイドドキュメントの履歴

次の表に、AWS AppConfig の前回のリリース以後に行われた、文書の重要な変更を示します。

現行 API バージョン: 2019-10-09

update-history-change	update-history-description	update-history-date
[] の新バージョンAWS AppConfigLambda 拡張 (p. 75)	のバージョン 2.0.58AWS AppConfigLambda 拡張機能が利用可能になりました。新しい拡張機能は、異なる Amazon Resource Name (ARN) を使用します。詳細については、次を参照してください。[] の利用可能なバージョンAWS AppConfigLambda 拡張。	2022 年 5 月 3 日
AWS AppConfigアトラシアン Jira との統合 (p. 75)	アトラシアン Jira との統合により、AWS AppConfigに変更を加えるたびに Atlassian コンソールで課題を作成して更新するには機能フラグ側AWS アカウント指定されたAWS リージョン。各 Jira 課題には、フラグ名、アプリケーション ID、設定プロファイル ID、フラグ値が含まれます。フラグの変更を更新、保存、デプロイすると、Jira は変更の詳細で既存の課題を更新します。詳細については、次を参照してください。AWS AppConfigアトラシアン Jira との統合。	2022 年 4 月 7 日
ARM64 (Graviton2) プロセッサの機能フラグと Lambda 拡張サポートの一般的な可用性 (p. 75)	とAWS AppConfig機能フラグを使用すると、新しい機能を開発し、その機能をユーザーから非表示にしながら、本番環境にデプロイできます。まず、フラグをに追加することから始めます。AWS AppConfigなどの他のツールを使用したり、機能をリリースする準備ができたら、コードをデプロイせずにフラグ設定データを更新できます。この機能により、機能をリリースするために新しいコードをデプロイする必要がないため、dev-ops 環境の安全性が向上します。詳細については、次を参照してください。機能フラグ設定プロファイルの作成。	2022 年 3 月 15 日

	の機能フラグの一般的な可用性AWS AppConfigには以下の機能強化が含まれています。 • コンソールには、フラグを短期フラグ。短期フラグのフラグのリストをフィルタリングしたり、ソートしたりできます。 • で機能フラグを使用しているお客様向けAWS Lambdaでは、新しい Lambda 拡張機能では、HTTP エンドポイントを使用して個々の機能フラグを呼び出すことができます。詳細については、次を参照してください。機能フラグ設定から 1 つ以上のフラグを取得する。 この更新プログラムでは、AWS LambdaARM64 (Graviton2) プロセッサ用に開発された拡張機能。詳細については、次を参照してください。[] の利用可能なバージョンAWS AppConfigLambda拡張。	
- GetConfiguration API アクションは廃止されました (p. 75)	-`GetConfiguration`API アクションは廃止されました。設定データを受信する呼び出しでは、`StartConfigurationSession`そして`GetLatestConfiguration`代わりに、API を使用してください。これらの API の詳細と使用方法については、」設定の取得。	2022 年 1 月 28 日
AWS AppConfig Lambda 拡張機能の新しいリージョン ARN (p. 75)	AWS AppConfig Lambda 拡張機能が、新しいアジアパシフィック (大阪) リージョンでご利用いただけるようになりました。Lambda をリージョンに作成するには、Amazon リソースネーム (ARN) が必要です。アジアパシフィック (大阪) リージョン ARN の詳細については、「AWS AppConfig Lambda 拡張機能の追加」を参照してください。	2021 年 3 月 4 日

AWS AppConfig Lambda 拡張機能 (p. 75)	AWS AppConfig を使用して Lambda 関数の設定を管理する場合、AWS AppConfig Lambda 拡張機能を追加することをお勧めします。この拡張機能には、コストを削減しながら、AWS AppConfig の使用を簡素化するベストプラクティスが含まれています。AWS AppConfig サービスへの API コールの減少によるコストの削減と、それとは別に、Lambda 関数の処理時間の短縮によるコスト削減が可能になります。詳細については、「AWS AppConfig と Lambda 拡張機能の統合」を参照してください。	2020 年 10 月 8 日
新規セクション (p. 75)	AWS AppConfig のセットアップの手順を提供する新しいセクションが追加されました。詳細については、「AWS AppConfig の設定」を参照してください。	2020 年 9 月 30 日
コマンドラインプロシージャの追加 (p. 75)	このユーザーガイドの手順には、AWS Command Line Interface (AWS CLI) のコマンドラインステップと Tools for Windows PowerShell が含まれています。詳細については、「AWS AppConfig の使用」を参照してください。	2020 年 9 月 30 日
AWS AppConfig ユーザーガイドの起動 (p. 75)	AWS AppConfig、AWS Systems Manager の機能を使用して、アプリケーション設定を作成、管理、迅速にデプロイします。AWS AppConfig はあらゆる規模のアプリケーションへの管理型デプロイをサポートし、組み込みの検証チェックおよびモニタリングを含みます。AWS AppConfig は、EC2 インスタンス、AWS Lambda、コンテナ、モバイルアプリケーション、または IoT デバイスでホストされているアプリケーションで使用できます。	2020 年 7 月 31 日

AWS 用語集

最新の AWS の用語については、AWS 全般のリファレンスの AWS 用語集を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。