Guide de l’utilisateur

AWS AppConfig

AWS AppConfig: Guide de l’utilisateur

Table of Contents

Qu'est-ce que c'est AWS AppConfig ?

AWS AppConfig les indicateurs de fonctionnalités et les configurations dynamiques aident les concepteurs de logiciels à ajuster rapidement et en toute sécurité le comportement des applications dans les environnements de production sans déploiement de code complet. AWS AppConfig accélère la fréquence de publication des logiciels, améliore la résilience des applications et vous aide à résoudre les problèmes émergents plus rapidement. Grâce aux indicateurs de fonctionnalités, vous pouvez publier progressivement de nouvelles fonctionnalités pour les utilisateurs et mesurer l'impact de ces changements avant de les déployer complètement pour tous les utilisateurs. Avec les indicateurs opérationnels et les configurations dynamiques, vous pouvez mettre à jour les listes de blocage, les listes d'autorisation, les limites de limitation, la verbosité des journaux et effectuer d'autres réglages opérationnels pour répondre rapidement aux problèmes dans les environnements de production.

> **Note**
>
> AWS AppConfig est un outil dans AWS Systems Manager.

Améliorez l'efficacité et publiez les modifications plus rapidement

L'utilisation d'indicateurs de fonctionnalités dotés de nouvelles fonctionnalités accélère le processus de publication des modifications dans les environnements de production. Au lieu de vous fier à des branches de développement pérennes qui nécessitent des fusions complexes avant une publication, les indicateurs de fonctionnalité vous permettent d'écrire des logiciels à l'aide d'un développement basé sur des troncs. Les indicateurs de fonctionnalité vous permettent de déployer du code de pré-version en toute sécurité dans un pipeline CI/CD masqué aux utilisateurs. Lorsque vous êtes prêt à publier les modifications, vous pouvez mettre à jour l'indicateur de fonctionnalité sans déployer de nouveau code. Une fois le lancement terminé, l'indicateur peut toujours fonctionner comme un commutateur de bloc pour désactiver une nouvelle fonctionnalité ou capacité sans qu'il soit nécessaire d'annuler le déploiement du code.

Évitez les modifications ou défaillances involontaires grâce aux fonctions de sécurité intégrées

AWS AppConfig propose les fonctionnalités de sécurité suivantes pour vous aider à éviter d'activer les indicateurs de fonctionnalité ou de mettre à jour les données de configuration susceptibles de provoquer des défaillances d'applications.

- Validateurs : un validateur garantit que vos données de configuration sont syntaxiquement et sémantiquement correctes avant de déployer les modifications dans les environnements de production.
- Stratégies de déploiement : une stratégie de déploiement vous permet d'apporter progressivement des modifications aux environnements de production en quelques minutes ou heures.
- Surveillance et annulation automatique : s' AWS AppConfig intègre CloudWatch à Amazon pour surveiller les modifications apportées à vos applications. Si votre application devient défectueuse en raison d'une modification de configuration incorrecte et que cette modification déclenche une alarme CloudWatch, AWS AppConfig annulez automatiquement la modification afin de minimiser l'impact sur les utilisateurs de votre application.

Déploiements d'indicateurs de fonctionnalités sécurisés et évolutifs

AWS AppConfig s'intègre à AWS Identity and Access Management (IAM) pour fournir un accès précis et basé sur les rôles au service. AWS AppConfig s'intègre également à AWS Key Management Service (AWS KMS) pour le chiffrement et AWS CloudTrail l'audit. Avant d'être mis à la disposition des clients externes, tous les contrôles de AWS AppConfig sécurité ont été initialement développés et validés par des clients internes qui utilisent le service à grande échelle.

AWS AppConfig cas d'utilisation

Bien que le contenu de configuration des applications puisse varier considérablement d'une application à l'autre, elle AWS AppConfig prend en charge les cas d'utilisation suivants, qui couvrent un large éventail de besoins des clients :

- Ajoutez des drapeaux et des boutons : offrez de nouvelles fonctionnalités à vos clients en toute sécurité dans un environnement contrôlé. Annulez instantanément les modifications en cas de problème.
- Optimisation des applications : introduisez soigneusement les modifications apportées aux applications tout en testant l'impact de ces modifications auprès des utilisateurs dans les environnements de production.
- Autoriser la liste ou la liste bloquée : contrôlez l'accès aux fonctionnalités premium ou bloquez instantanément des utilisateurs spécifiques sans déployer de nouveau code.
- Stockage de configuration centralisé : veillez à ce que vos données de configuration soient organisées et cohérentes pour toutes vos charges de travail. Vous pouvez les utiliser AWS AppConfig pour déployer les données de configuration stockées dans le magasin de configuration

AWS AppConfig hébergé AWS Secrets Manager, le magasin de paramètres Systems Manager ou Amazon S3.

Avantages de l'utilisation AWS AppConfig

AWS AppConfig offre les avantages suivants à votre organisation :

- Réduisez les temps d'arrêt imprévus pour vos clients

  AWS AppConfig réduit le temps d'arrêt des applications en vous permettant de créer des règles pour valider votre configuration. Les configurations non valides ne peuvent pas être déployées. AWS AppConfig propose les deux options suivantes pour valider les configurations :

  - Pour la validation syntaxique, vous pouvez utiliser un schéma JSON. AWS AppConfig valide votre configuration en utilisant le schéma JSON pour garantir que les modifications de configuration sont conformes aux exigences de l'application.
  - Pour la validation sémantique, vous AWS AppConfig pouvez appeler une AWS Lambda fonction que vous possédez pour valider les données de votre configuration.
- Déployez rapidement les modifications sur un ensemble de cibles

  AWS AppConfig simplifie l'administration des applications à grande échelle en déployant les modifications de configuration à partir d'un emplacement central. AWS AppConfig prend en charge les configurations stockées dans le magasin de configuration AWS AppConfig hébergé, le magasin de paramètres de Systems Manager, les documents Systems Manager (SSM) et Amazon S3. Vous pouvez l'utiliser AWS AppConfig avec des applications hébergées sur des EC2 instances AWS Lambda, des conteneurs, des applications mobiles ou des appareils IoT.

  Les cibles n'ont pas besoin d'être configurées avec l'agent SSM de Systems Manager ou avec le profil d'instance IAM requis par les autres outils de Systems Manager. Cela signifie que cela AWS AppConfig fonctionne avec les instances non gérées.
- Mise à jour des applications sans interruption

  AWS AppConfig déploie les modifications de configuration sur vos cibles au moment de l'exécution sans avoir à recourir à un processus de génération fastidieux ou à la mise hors service de vos cibles.
- Contrôle du déploiement des modifications dans votre application

Lorsque vous déployez des modifications de configuration sur vos cibles, cela vous AWS AppConfig permet de minimiser les risques en utilisant une stratégie de déploiement. Les stratégies de déploiement vous permettent de déployer lentement les modifications de configuration de votre flotte. Si vous rencontrez un problème lors du déploiement, vous pouvez annuler la modification de configuration avant qu'elle n'atteigne la majorité de vos hôtes.

Comment AWS AppConfig fonctionne

Cette section fournit une description détaillée du AWS AppConfig fonctionnement et de la façon dont vous pouvez démarrer.

1. Identifiez les valeurs de configuration dans le code que vous souhaitez gérer dans le cloud

   Avant de commencer à créer AWS AppConfig des artefacts, nous vous recommandons d'identifier dans votre code les données de configuration que vous souhaitez gérer de manière dynamique AWS AppConfig. Parmi les bons exemples, citons les indicateurs ou les boutons de fonctionnalité, les listes d'autorisation et de blocage, la verbosité de la journalisation, les limites de service et les règles de limitation, pour n'en citer que quelques-uns.

   Si vos données de configuration existent déjà dans le cloud, vous pouvez tirer parti des fonctionnalités de AWS AppConfig validation, de déploiement et d'extension pour rationaliser davantage la gestion des données de configuration.
2. Création d'un espace de noms d'application

   Pour créer un espace de noms, vous devez créer un AWS AppConfig artefact appelé application. Une application est simplement une structure organisationnelle telle qu'un dossier.
3. Créer des environnements

   Pour chaque AWS AppConfig application, vous définissez un ou plusieurs environnements. Un environnement est un regroupement logique de cibles, telles que des applications dans un `Beta` `Production` environnement, des AWS Lambda fonctions ou des conteneurs. Vous pouvez également définir des environnements pour les sous-composants de l'application, tels que `WebMobile`, et`Back-end`.

   Vous pouvez configurer les CloudWatch alarmes Amazon pour chaque environnement. Le système surveille les alarmes lors d'un déploiement de configuration. Si une alarme est déclenchée, le système annule la configuration.

4. Création d'un profil de configuration

   Un profil de configuration inclut, entre autres, une URI qui permet de AWS AppConfig localiser vos données de configuration dans leur emplacement enregistré et un type de profil. AWS AppConfig prend en charge deux types de profils de configuration : les indicateurs de fonctionnalité et les configurations de forme libre. Les profils de configuration Feature Flag stockent leurs données dans le magasin de configuration AWS AppConfig hébergé et l'URI est simple`hosted`. Pour les profils de configuration libres, vous pouvez stocker vos données dans le magasin de configuration AWS AppConfig hébergé ou dans tout AWS service intégré AWS AppConfig, comme décrit dansCréation d'un profil de configuration sous forme libre dans AWS AppConfig.

   Un profil de configuration peut également inclure des validateurs facultatifs pour garantir l'exactitude syntaxique et sémantique de vos données de configuration. AWS AppConfig effectue une vérification à l'aide des validateurs lorsque vous démarrez un déploiement. Si des erreurs sont détectées, le déploiement revient aux données de configuration précédentes.

5. Déployer les données de configuration

   Lorsque vous créez un nouveau déploiement, vous spécifiez les éléments suivants :

   - Un identifiant d'application
   - Un ID de profil de configuration
   - Une version de configuration
   - Un ID d'environnement dans lequel vous souhaitez déployer les données de configuration
   - Un identifiant de stratégie de déploiement qui définit la rapidité avec laquelle vous souhaitez que les modifications prennent effet

   Lorsque vous appelez l'action StartDeploymentAPI, AWS AppConfig exécute les tâches suivantes :

   1. Récupère les données de configuration du magasin de données sous-jacent à l'aide de l'URI de localisation dans le profil de configuration.
   2. Vérifie que les données de configuration sont syntaxiquement et sémantiquement correctes en utilisant les validateurs que vous avez spécifiés lors de la création de votre profil de configuration.
   3. Met en cache une copie des données afin qu'elles soient prêtes à être récupérées par votre application. Cette copie mise en cache s'appelle les données déployées.

6. Récupérez la configuration

   Vous pouvez configurer AWS AppConfig l'agent en tant qu'hôte local et demander à l'agent de AWS AppConfig demander des mises à jour de configuration. L'agent appelle les actions StartConfigurationSessionet GetLatestConfigurationAPI et met en cache vos données de configuration localement. Pour récupérer les données, votre application lance un appel HTTP au serveur localhost. AWS AppConfig L'agent prend en charge plusieurs cas d'utilisation, comme décrit dansComment utiliser l' AWS AppConfig agent pour récupérer les données de configuration.

   Si AWS AppConfig l'agent n'est pas pris en charge pour votre cas d'utilisation, vous pouvez configurer votre application AWS AppConfig pour demander des mises à jour de configuration en appelant directement les actions StartConfigurationSessionet GetLatestConfigurationAPI.

Mise en route avec AWS AppConfig

Les ressources suivantes peuvent vous aider à travailler directement avec AWS AppConfig.

Visionnez d'autres AWS vidéos sur la YouTube chaîne Amazon Web Services.

Les blogs suivants peuvent vous aider à en savoir plus sur AWS AppConfig ses fonctionnalités :

- Utilisation d'indicateurs AWS AppConfig de fonctionnalité
- Meilleures pratiques pour valider les indicateurs de AWS AppConfig fonctionnalités et les données de configuration

SDKs

Pour plus d'informations sur les AWS AppConfig langues spécifiques SDKs, consultez les ressources suivantes :

- AWS Command Line Interface
- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go
- AWS SDK pour Java V2
- AWS SDK pour JavaScript

- AWS SDK pour PHP V3
- AWS SDK pour Python
- AWS SDK pour Ruby V3

Tarification pour AWS AppConfig

La tarification AWS AppConfig est pay-as-you-go basée sur les données de configuration et la récupération des indicateurs de fonctionnalités. Nous vous recommandons d'utiliser l' AWS AppConfig agent pour optimiser les coûts. Pour plus d'informations, consultez AWS Systems Manager Pricing (Tarification CTlong).

AWS AppConfig quotas

Les informations sur les AWS AppConfig points de terminaison et les quotas de service, ainsi que sur les autres quotas de Systems Manager, se trouvent dans le Référence générale d'Amazon Web Services.

> **Note**
>
> Pour plus d'informations sur les quotas pour les services qui stockent AWS AppConfig des configurations, consultezComprendre les quotas et les limites du magasin de configuration.

Con AWS AppConfig figuration

Si ce n'est pas déjà fait, inscrivez-vous Compte AWS et créez un utilisateur administratif.

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .

2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu'utilisateur doté d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d'un ensemble d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l'utilisateur AWS IAM Identity Center .

Octroi d'un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Comprendre le IPv6 support

Tous les AWS AppConfig APIs IPv6 appels IPv4 et le support sont complets.

Plan de contrôle APIs

Utilisez le point de terminaison suivant pour IPv4 les appels IPv6 à double pile vers le plan de contrôle :

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

Par exemple : appconfig.us-east-1.api.aws

Pour IPv4 uniquement, utilisez l'URL suivante :

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

Plan de données APIs

Pour les appels à double pile vers le plan de données, utilisez le point de terminaison suivant :

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

Par exemple : appconfig.us-east-1.api.aws

Pour IPv4 uniquement, utilisez l'URL suivante :

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

> **Note**
>
> Pour plus d'informations, consultez Points de terminaison et quotas AWS AppConfig dans le document Références générales AWS.

(Recommandé) Configurer les autorisations pour la restauration automatique

Vous pouvez configurer AWS AppConfig pour revenir à une version précédente d'une configuration en réponse à une ou plusieurs CloudWatch alarmes Amazon. Lorsque vous configurez un déploiement pour répondre aux CloudWatch alarmes, vous spécifiez un rôle AWS Identity and Access Management (IAM). AWS AppConfig nécessite ce rôle afin de pouvoir surveiller les CloudWatch alarmes. Cette procédure est facultative, mais vivement recommandée.

> **Note**
>
> Notez les informations suivantes.
>
> - Le rôle IAM doit appartenir au compte courant. Par défaut, seules les alarmes détenues par le compte courant AWS AppConfig peuvent être surveillées.

- Pour plus d'informations sur les mesures à surveiller et sur la manière AWS AppConfig de configurer la restauration automatique, consultezSurveillance des déploiements pour une annulation automatique.

Utilisez les procédures suivantes pour créer un rôle IAM qui permet de revenir en arrière AWS AppConfig en fonction CloudWatch des alarmes. Cette section comprend les procédures suivantes.

1. Étape 1 : créer la politique d'autorisation pour l'annulation en fonction des alarmes CloudWatch
2. Étape 2 : créer le rôle IAM pour la restauration en fonction des alarmes CloudWatch
3. Étape 3 : Ajout d'une relation d'approbation

Étape 1 : créer la politique d'autorisation pour l'annulation en fonction des alarmes CloudWatch

Utilisez la procédure suivante pour créer une politique IAM AWS AppConfig autorisant l'appel de l'action `DescribeAlarms` API.

Pour créer une politique d'autorisation IAM pour une annulation basée sur des alarmes CloudWatch

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Politiques, puis Créer une politique.
3. Sur la page Créer une politique, choisissez l'onglet JSON.
4. Remplacez le contenu par défaut de l'onglet JSON par la politique d'autorisation suivante, puis choisissez Next : Tags.

> **Note**
>
> Pour renvoyer des informations sur les alarmes CloudWatch composites, des * autorisations doivent être attribuées à l'opération DescribeAlarmsAPI, comme indiqué ici. Vous ne pouvez pas renvoyer d'informations sur les alarmes composites si leur champ `DescribeAlarms` d'application est plus restreint.

JSON

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

5. Entrez des balises pour ce rôle, puis choisissez Next: Review (Suivant : Vérification).
6. Sur la page Révision, entrez **`SSMCloudWatchAlarmDiscoveryPolicy`** dans le champ Nom.
7. Choisissez Create Policy (Créer une politique). Le système vous renvoie à la page Policies (Stratégies).

Étape 2 : créer le rôle IAM pour la restauration en fonction des alarmes CloudWatch

Utilisez la procédure suivante pour créer un rôle IAM et lui attribuer la politique que vous avez créée dans la procédure précédente.

Pour créer un rôle IAM à des fins d'annulation en fonction des alarmes CloudWatch

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Rôles, puis Créer un rôle.
3. Sous Select type of trusted entity, sélectionnez AWS service.
4. Immédiatement sous Choisissez le service qui utilisera ce rôle, choisissez EC2: Autorise les EC2 instances à appeler AWS des services en votre nom, puis choisissez Suivant : Autorisations.
5. Sur la page Politique d'autorisations attachée, recherchez SSMCloudWatchAlarmDiscoveryPolicy.
6. Choisissez cette stratégie, puis Next: Tags (Suivant : Balises).

7. Entrez des balises pour ce rôle, puis choisissez Next: Review (Suivant : Vérification).
8. Sur la page Créer un rôle, entrez **`SSMCloudWatchAlarmDiscoveryRole`** le champ Nom du rôle, puis choisissez Créer un rôle.
9. Sur la page Rôles, sélectionnez le rôle que vous venez de créer. La page Récapitulatif s'ouvre.

Étape 3 : Ajout d'une relation d'approbation

Utilisez la procédure suivante pour configurer le rôle que vous venez de créer pour approuver AWS AppConfig.

Pour ajouter une relation de confiance pour AWS AppConfig

1. Dans la page Récapitulatif du rôle que vous venez de créer, choisissez l'onglet Relations d'approbation, puis choisissez Modifier la relation d'approbation.
2. Modifiez la stratégie pour inclure uniquement « `appconfig.amazonaws.com` », comme indiqué dans l'exemple suivant :

   JSON

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

3. Choisissez Mettre à jour la politique d'approbation.

Création d'indicateurs de fonctionnalités et de données de configuration sous forme libre dans AWS AppConfig

Les rubriques de cette section vous aident à effectuer les tâches suivantes dans AWS AppConfig. Ces tâches créent des artefacts importants pour le déploiement des données de configuration.

1. Création d'un espace de noms d'application

   Pour créer un espace de noms d'application, vous devez créer un AWS AppConfig artefact appelé application. Une application est simplement une structure organisationnelle telle qu'un dossier.

2. Créez des environnements

   Pour chaque AWS AppConfig application, vous définissez un ou plusieurs environnements. Un environnement est un groupe de déploiement logique de AWS AppConfig cibles, telles que des applications dans un `Production` environnement `Beta` OR. Vous pouvez également définir des environnements pour les sous-composants de l'application, tels que `AWS Lambda functionsContainers`,`Web`,`Mobile`, et`Back-end`.

   Vous pouvez configurer les CloudWatch alarmes Amazon pour chaque environnement afin d'annuler automatiquement les modifications de configuration problématiques. Le système surveille les alarmes lors d'un déploiement de configuration. Si une alarme est déclenchée, le système annule la configuration.

3. Création d'un profil de configuration

   Les données de configuration sont un ensemble de paramètres qui influencent le comportement de votre application. Un profil de configuration inclut, entre autres, un URI qui permet de AWS AppConfig localiser vos données de configuration dans leur emplacement enregistré et un type de configuration. AWS AppConfig prend en charge les types de profils de configuration suivants :

   - Indicateurs de fonctionnalités : vous pouvez utiliser des indicateurs de fonctionnalité pour activer ou désactiver des fonctionnalités au sein de vos applications ou pour configurer différentes caractéristiques des fonctionnalités de vos applications à l'aide d'attributs d'indicateur. AWS AppConfig stocke les configurations d'indicateurs de fonctionnalités dans le magasin de configuration AWS AppConfig hébergé dans un format d'indicateur de fonctionnalité qui contient des données et des métadonnées relatives à vos indicateurs et aux attributs des indicateurs. L'URI pour les configurations d'indicateurs de fonctionnalité est simple`hosted`.
   - Configurations de forme libre : une configuration de forme libre peut stocker des données dans l'un des outils suivants Services AWS et dans les outils Systems Manager :

- AWS AppConfig magasin de configuration hébergé
- Amazon Simple Storage Service
- AWS CodePipeline
- AWS Secrets Manager
- AWS Systems Manager (SSM) Magasin de paramètres
- Boutique de documents SSM

> **Note**
>
> Dans la mesure du possible, nous vous recommandons d'héberger vos données de configuration dans le magasin de configuration AWS AppConfig hébergé, qui offre le plus de fonctionnalités et d'améliorations.

4. (Facultatif, mais recommandé) Créez des indicateurs de fonctionnalités à variantes multiples

   AWS AppConfig propose des indicateurs de fonctionnalités de base qui (s'ils sont activés) renvoient un ensemble spécifique de données de configuration par demande. Pour mieux prendre en charge les cas d'utilisation liés à la segmentation des utilisateurs et à la répartition du trafic, propose AWS AppConfig également des indicateurs de fonctionnalités à variantes multiples, qui vous permettent de définir un ensemble de valeurs d'indicateur possibles à renvoyer pour une demande. Vous pouvez également configurer différents statuts (activé ou désactivé) pour les indicateurs à variantes multiples. Lorsque vous demandez un indicateur configuré avec des variantes, votre application fournit un contexte qui est AWS AppConfig évalué par rapport à un ensemble de règles définies par l'utilisateur. En fonction du contexte spécifié dans la demande et des règles définies pour la variante, AWS AppConfig renvoie différentes valeurs d'indicateur à l'application.

Rubriques

- Comprendre le rôle IAM du profil de configuration
- Création d'un espace de noms pour votre application dans AWS AppConfig
- Création d'environnements pour votre application dans AWS AppConfig
- Création d'un profil de configuration dans AWS AppConfig

Comprendre le rôle IAM du profil de configuration

Vous pouvez créer le rôle IAM qui donne accès aux données de configuration à l'aide AWS AppConfig de. Vous pouvez également créer le rôle IAM vous-même. Si vous créez le rôle en utilisant AWS AppConfig, le système crée le rôle et spécifie l'une des politiques d'autorisation suivantes, en fonction du type de source de configuration que vous choisissez.

La source de configuration est un secret de Secrets Manager

JSON

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

La source de configuration est un paramètre Parameter Store

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
```

```
                "arn:aws:ssm:Région AWS:account_ID:parameter/parameter_name"
            ]
        }
    ]
    }
```

La source de configuration est un document SSM

JSON

```
{

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

Si vous créez le rôle en utilisant AWS AppConfig, le système crée également la relation de confiance suivante pour le rôle.

JSON

```
{

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

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

Création d'un espace de noms pour votre application dans AWS AppConfig

Les procédures décrites dans cette section vous aident à créer un AWS AppConfig artefact appelé application. Une application est simplement une structure organisationnelle telle qu'un dossier qui identifie l'espace de noms de votre application. Cette structure organisationnelle a une relation avec une unité de code exécutable. Par exemple, vous pouvez créer une application appelée MyMobileApp pour organiser et gérer les données de configuration d'une application mobile installée par vos utilisateurs. Vous devez créer ces artefacts avant de pouvoir les utiliser AWS AppConfig pour déployer et récupérer des indicateurs de fonctionnalités ou des données de configuration sous forme libre.

La procédure suivante vous permet d'associer une extension à un profil de configuration d'indicateur de fonctionnalité. Une extension augmente votre capacité à injecter de la logique ou du comportement à différents moments du AWS AppConfig flux de travail de création ou de déploiement d'une configuration. Pour de plus amples informations, veuillez consulter Comprendre les AWS AppConfig extensions.

> **Note**
>
> Vous pouvez les utiliser AWS CloudFormation pour créer des AWS AppConfig artefacts, notamment des applications, des environnements, des profils de configuration, des déploiements, des stratégies de déploiement et des versions de configuration hébergées. Pour plus d'informations, consultez Référence du type de ressource AWS AppConfig dans le Guide de l'utilisateur AWS CloudFormation .

Rubriques

- Création d'une AWS AppConfig application (console)
- Création d'une AWS AppConfig application (ligne de commande)

Création d'une AWS AppConfig application (console)

Pour créer une AWS AppConfig application à l'aide de la AWS Systems Manager console, procédez comme suit.

Pour créer une application

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications (Applications), puis Create a new application (Créer une nouvelle application).
3. Pour Name (Nom), entrez un nom pour l'application.
4. Pour Description, entrez les informations concernant l'application.
5. (Facultatif) Dans la section Extensions, choisissez une extension dans la liste. Pour de plus amples informations, veuillez consulter Comprendre les AWS AppConfig extensions.
6. (Facultatif) Dans la section Balises, entrez une clé et une valeur facultative. Vous pouvez spécifier un maximum de 50 balises par ressource.
7. Choisissez Créer une application.

AWS AppConfig crée l'application puis affiche l'onglet Environnements. Passez à Création d'environnements pour votre application dans AWS AppConfig.

Création d'une AWS AppConfig application (ligne de commande)

La procédure suivante décrit comment utiliser AWS CLI (sous Linux ou Windows) ou comment Outils AWS pour PowerShell créer une AWS AppConfig application.

Pour créer une application étape par étape

1. Ouvrez le AWS CLI.
2. Exécutez la commande suivante pour créer une application.

   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
```

Le système retourne des informations telles que les suivantes.

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
```

Création d'environnements pour votre application dans AWS AppConfig

Pour chaque AWS AppConfig application, vous définissez un ou plusieurs environnements. Un environnement est un groupe de déploiement logique de AppConfig cibles, telles que des applications dans un `Beta` `Production` environnement, des AWS Lambda fonctions ou des conteneurs. Vous pouvez également définir des environnements pour les sous-composants de l'application, tels que `WebMobile`, et`Back-end`. Vous pouvez configurer les CloudWatch alarmes Amazon pour chaque environnement. Le système surveille les alarmes lors d'un déploiement de configuration. Si une alarme est déclenchée, le système annule la configuration.

Avant de commencer

Si vous AWS AppConfig souhaitez activer l'annulation d'une configuration en réponse à une CloudWatch alarme, vous devez configurer un rôle AWS Identity and Access Management (IAM) avec des autorisations permettant de répondre AWS AppConfig aux CloudWatch alarmes. Vous choisissez ce rôle dans la procédure suivante. Pour de plus amples informations, veuillez consulter (Recommandé) Configurer les autorisations pour la restauration automatique.

Rubriques

- Création d'un AWS AppConfig environnement (console)
- Création d'un AWS AppConfig environnement (ligne de commande)

Création d'un AWS AppConfig environnement (console)

Pour créer un AWS AppConfig environnement à l'aide de la AWS Systems Manager console, procédez comme suit.

Pour créer un environnement .

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis le nom d'une application pour ouvrir la page de détails.

3. Choisissez l'onglet Environnements, puis sélectionnez Créer un environnement.
4. Pour Name (Nom), entrez un nom pour l'environnement.
5. Pour Description, entrez les informations concernant l'environnement.
6. (Facultatif) Dans la section Moniteurs, choisissez le champ Rôle IAM, puis choisissez un rôle IAM autorisé à faire appel `cloudwatch:DescribeAlarms` aux métriques que vous souhaitez surveiller pour détecter les alarmes.
7. Dans la liste des CloudWatch alarmes, entrez dans Amazon Resource Names (ARNs) une ou plusieurs métriques à surveiller. AWS AppConfig annule le déploiement de votre configuration si l'une de ces métriques passe à un `ALARM` état. Pour plus d'informations sur les mesures recommandées, voir Surveillance des déploiements pour une annulation automatique
8. (Facultatif) Dans la section Associer des extensions, choisissez une extension dans la liste. Pour de plus amples informations, veuillez consulter Comprendre les AWS AppConfig extensions.
9. (Facultatif) Dans la section Balises, entrez une clé et une valeur facultative. Vous pouvez spécifier un maximum de 50 balises par ressource.
10. Choisissez Create environment.

AWS AppConfig crée l'environnement, puis affiche la page des détails de l'environnement. Passez à Création d'un profil de configuration dans AWS AppConfig.

Création d'un AWS AppConfig environnement (ligne de commande)

La procédure suivante décrit comment utiliser AWS CLI (sous Linux ou Windows) ou comment Outils AWS pour PowerShell créer un AWS AppConfig environnement.

Pour créer un environnement étape par étape

1. Ouvrez le AWS CLI.
2. Exécutez la commande suivante pour créer un environnement.

   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
```

Le système retourne des informations telles que les suivantes.

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
```

Passez à Création d'un profil de configuration dans AWS AppConfig.

Création d'un profil de configuration dans AWS AppConfig

Les données de configuration sont un ensemble de paramètres qui influencent le comportement de votre application. Un profil de configuration inclut, entre autres, un URI qui permet de AWS AppConfig localiser vos données de configuration dans leur emplacement enregistré et un type de configuration. AWS AppConfig prend en charge les types de profils de configuration suivants :

- Indicateurs de fonctionnalités : vous pouvez utiliser des indicateurs de fonctionnalité pour activer ou désactiver des fonctionnalités au sein de vos applications ou pour configurer différentes caractéristiques des fonctionnalités de vos applications à l'aide d'attributs d'indicateur. AWS AppConfig stocke les configurations d'indicateurs de fonctionnalités dans le magasin de configuration AWS AppConfig hébergé dans un format d'indicateur de fonctionnalité qui contient des données et des métadonnées relatives à vos indicateurs et aux attributs des indicateurs. L'URI pour les configurations d'indicateurs de fonctionnalité est simple`hosted`.
- Configurations de forme libre : une configuration de forme libre peut stocker des données dans l'un des outils suivants Services AWS et dans les outils Systems Manager :
  - AWS AppConfig magasin de configuration hébergé
  - Amazon Simple Storage Service
  - AWS CodePipeline
  - AWS Secrets Manager
  - AWS Systems Manager (SSM) Magasin de paramètres
  - Boutique de documents SSM

> **Note**
>
> Dans la mesure du possible, nous vous recommandons d'héberger vos données de configuration dans le magasin de configuration AWS AppConfig hébergé, qui offre le plus de fonctionnalités et d'améliorations.

Voici quelques exemples de données de configuration pour vous aider à mieux comprendre les différents types de données de configuration et la manière dont elles peuvent être utilisées dans un indicateur de fonctionnalité ou sans profil de configuration.

Données de configuration des indicateurs de fonctionnalité

Les fonctionnalités suivantes indiquent que les données de configuration activent ou désactivent les paiements mobiles et les paiements par défaut pour chaque région.

JSON

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

YAML

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

Données de configuration opérationnelle

Les données de configuration libres suivantes imposent des limites quant à la manière dont une application traite les demandes.

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
```

Données de configuration de la liste de contrôle d'accès

Les données de configuration libres de la liste de contrôle d'accès ci-dessous indiquent quels utilisateurs ou groupes peuvent accéder à une application.

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
```

Rubriques

- Création d'un profil de configuration d'indicateur de fonctionnalité dans AWS AppConfig
- Création d'un profil de configuration sous forme libre dans AWS AppConfig
- Création d'un profil de configuration pour les sources de données non natives

Création d'un profil de configuration d'indicateur de fonctionnalité dans AWS AppConfig

Vous pouvez utiliser des indicateurs de fonctionnalité pour activer ou désactiver des fonctionnalités au sein de vos applications ou pour configurer différentes caractéristiques des fonctionnalités de vos applications à l'aide d'attributs d'indicateur. AWS AppConfig stocke les configurations d'indicateurs de fonctionnalités dans le magasin de configuration AWS AppConfig hébergé dans un format d'indicateur de fonctionnalité qui contient des données et des métadonnées relatives à vos indicateurs et aux attributs des indicateurs.

> **Note**
>
> Lorsque vous créez un profil de configuration d'indicateur de fonctionnalité, vous pouvez créer un indicateur de fonctionnalité de base dans le cadre du flux de travail du profil de configuration. AWS AppConfig prend également en charge les drapeaux de fonctionnalités à variantes multiples. Les indicateurs de fonctionnalités à variantes multiples vous permettent de définir un ensemble de valeurs d'indicateurs possibles à renvoyer pour une demande.

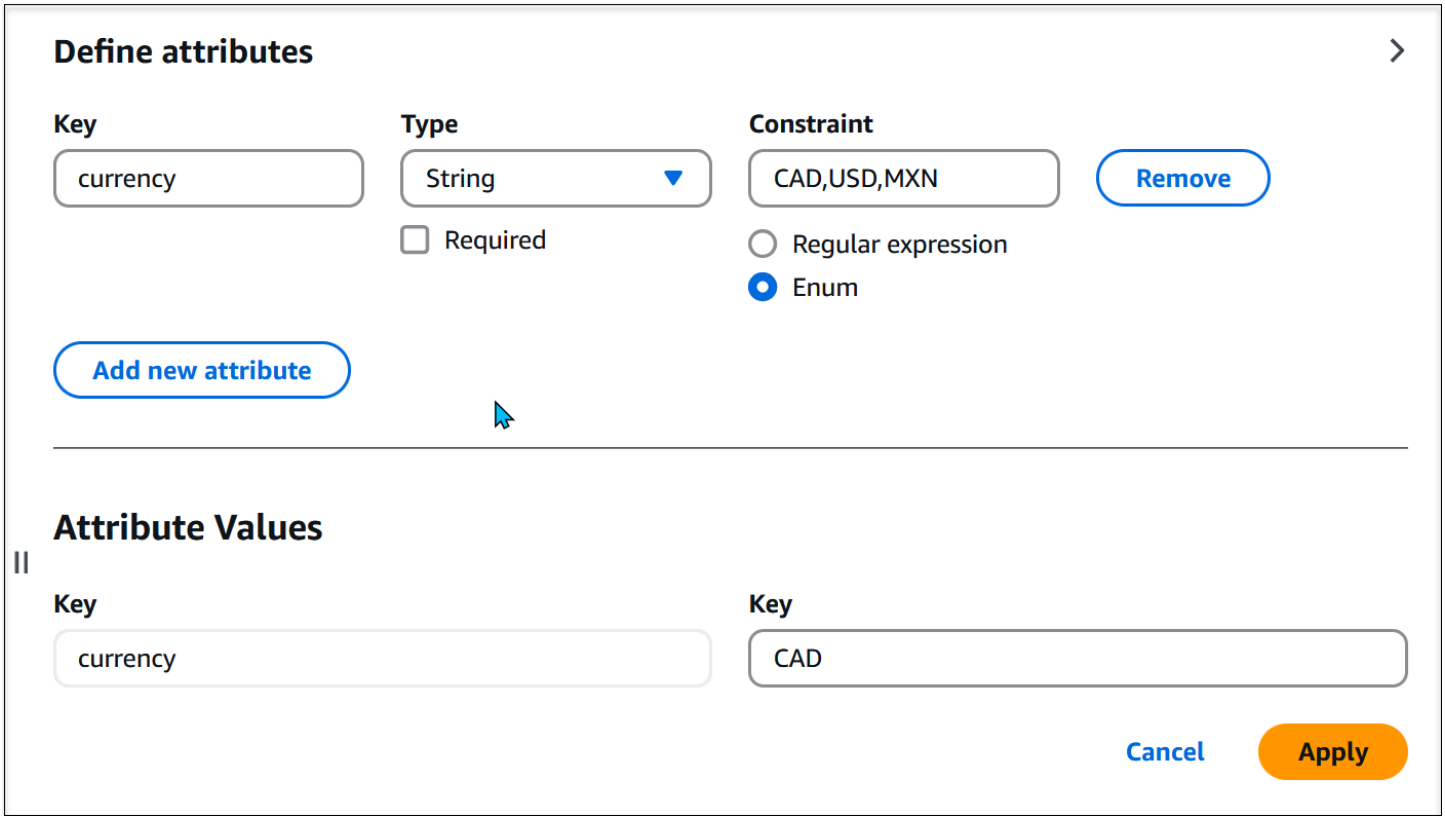
Lorsque vous demandez un indicateur configuré avec des variantes, votre application fournit un contexte qui est AWS AppConfig évalué par rapport à un ensemble de règles définies par l'utilisateur. En fonction du contexte spécifié dans la demande et des règles définies pour la variante, AWS AppConfig renvoie différentes valeurs d'indicateur à l'application.
Pour créer des indicateurs de fonctionnalités à variantes multiples, créez d'abord un profil de configuration, puis modifiez les indicateurs du profil de configuration pour ajouter des variantes. Pour de plus amples informations, veuillez consulter Création d'indicateurs de fonctionnalités à variantes multiples.

Rubriques

Comprendre les attributs des indicateurs de fonctionnalité

Lorsque vous créez un profil de configuration d'indicateur de fonctionnalité, ou que vous créez un nouvel indicateur dans un profil de configuration existant, vous pouvez spécifier les attributs et les contraintes correspondantes pour l'indicateur. Un attribut est un champ que vous associez à votre indicateur d'entité pour exprimer les propriétés associées à votre indicateur d'entité. Les attributs sont fournis à votre application avec votre clé de drapeau et la `disable` valeur `enable` ou du drapeau.

Les contraintes garantissent que les valeurs d'attribut inattendues ne sont pas déployées dans votre application. L'image suivante illustre cet affichage :



> **Note**
>
> Notez les informations suivantes concernant les attributs des drapeaux.
>
> - Pour les noms d'attributs, le mot « activé » est réservé. Vous ne pouvez pas créer un attribut d'indicateur de fonctionnalité appelé « activé ». Il n'y a pas d'autres mots réservés.
> - Les attributs d'un indicateur de fonctionnalité ne sont inclus dans la `GetLatestConfiguration` réponse que si cet indicateur est activé.
> - Les clés d'attribut d'un drapeau donné doivent être uniques.

AWS AppConfig prend en charge les types d'attributs de drapeau suivants et les contraintes correspondantes.

Type	Contrainte	Description
String	Expression régulière	Modèle Regex pour la chaîne

Type	Contrainte	Description
	Enum	Liste des valeurs acceptables pour la chaîne
Nombre	Minimum	Valeur numérique minimale pour l'attribut
	Maximum	Valeur numérique maximale pour l'attribut
Booléen	Aucun	Aucun
Tableau de chaînes	Expression régulière	Modèle Regex pour les éléments du tableau
	Enum	Liste des valeurs acceptables pour les éléments du tableau
Tableau de numéros	Minimum	Valeur numérique minimale pour les éléments du tableau
	Maximum	Valeur numérique maximale pour les éléments du tableau

Création d'un profil de configuration d'indicateur de fonctionnalité (console)

Utilisez la procédure suivante pour créer un profil de configuration d'indicateur de AWS AppConfig fonctionnalité à l'aide de la AWS AppConfig console. Au moment de créer le profil de configuration, vous pouvez également créer un indicateur de fonctionnalité de base.

Pour créer un profil de configuration

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis choisissez une application que vous avez créée dansCréation d'un espace de noms pour votre application dans AWS AppConfig.

3. Dans l'onglet Profils de configuration et indicateurs de fonctionnalités, choisissez Créer une configuration.
4. Dans la section Options de configuration, choisissez Feature flag.
5. Dans la section Profil de configuration, pour Nom du profil de configuration, entrez un nom.
6. (Facultatif) Développez la description et entrez une description.
7. (Facultatif) Développez les options supplémentaires et effectuez les opérations suivantes, si nécessaire.

    a. Dans la liste Chiffrement, choisissez une clé AWS Key Management Service (AWS KMS) dans la liste. Cette clé gérée par le client vous permet de chiffrer les nouvelles versions des données de configuration dans le magasin de configuration AWS AppConfig hébergé. Pour plus d'informations sur cette clé, consultez la section AWS AppConfig Supporte les touches du gestionnaire de clientèleSécurité dans AWS AppConfig.

    b. Dans la section Balises, choisissez Ajouter une nouvelle balise, puis spécifiez une clé et une valeur facultative.
8. Choisissez Suivant.
9. Dans la section Définition du drapeau de fonctionnalité, pour Nom du drapeau, entrez un nom.
10. Pour la touche Drapeau, entrez un identifiant de drapeau pour distinguer les indicateurs au sein d'un même profil de configuration. Les drapeaux d'un même profil de configuration ne peuvent pas avoir la même clé. Une fois le drapeau créé, vous pouvez modifier le nom du drapeau, mais pas la clé du drapeau.
11. (Facultatif) Développez la description et entrez les informations relatives à cet indicateur.
12. Sélectionnez Ceci est un drapeau à court terme et choisissez éventuellement une date à laquelle le drapeau doit être désactivé ou supprimé. AWS AppConfig ne désactive pas le drapeau à la date d'obsolescence.
13. (Facultatif) Dans la section Attributs du drapeau de fonction, choisissez Définir l'attribut. Les attributs vous permettent de fournir des valeurs supplémentaires dans votre drapeau. Pour plus d'informations sur les attributs et les contraintes, consultezComprendre les attributs des indicateurs de fonctionnalité.

    a. Pour Clé, spécifiez une touche indicateur et choisissez son type dans la liste Type. Pour plus d'informations sur les options prises en charge pour les champs Valeur et Contraintes, consultez la section précédemment référencée sur les attributs.

    b. Sélectionnez Valeur requise pour indiquer si une valeur d'attribut est requise.

    c. Choisissez Définir un attribut pour ajouter des attributs supplémentaires.

14. Dans la section Valeur de l'indicateur de fonctionnalité, choisissez Activé pour activer l'indicateur. Utilisez cette même option pour désactiver un indicateur lorsqu'il atteint une date de dépréciation spécifiée, le cas échéant.
15. Choisissez Suivant.
16. Sur la page Réviser et enregistrer, vérifiez les détails de l'indicateur, puis enregistrez et poursuivez le déploiement.

Passez à Déploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig.

Création d'un profil de configuration d'indicateur de fonctionnalité (ligne de commande)

La procédure suivante décrit comment utiliser le AWS Command Line Interface (sous Linux ou Windows) ou les outils pour Windows PowerShell pour créer un profil de configuration avec indicateur de AWS AppConfig fonctionnalité. Au moment de créer le profil de configuration, vous pouvez également créer un indicateur de fonctionnalité de base.

Pour créer une configuration d'indicateur de fonctionnalité

1. Ouvrez le AWS CLI.
2. Créez un profil de configuration d'indicateur de fonctionnalité en spécifiant son type commeAWS.AppConfig.FeatureFlags. Le profil de configuration doit être utilisé `hosted` comme URI de localisation.

   Linux

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

   Windows

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

PowerShell

```
New-APPCConfigurationProfile `
  -Name CONFIGURATION_PROFILE_NAME `
  -ApplicationId APPLICATION_ID `
  -LocationUri hosted `
  -Type AWS.AppConfig.FeatureFlags
```

3. Créez les données de configuration de votre indicateur de fonctionnalité. Vos données doivent être au format JSON et conformes au schéma `AWS.AppConfig.FeatureFlags` JSON. Pour plus d'informations sur le schéma, consultezComprendre la référence de type pour AWS.AppConfig.FeatureFlags.
4. Utilisez l'`CreateHostedConfigurationVersion`API pour enregistrer les données de configuration de votre indicateur de fonctionnalité dans AWS AppConfig.

   Linux

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

   Windows

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

   PowerShell

   ```
   New-APPCHostedConfigurationVersion `
     -ApplicationId APPLICATION_ID `
   ```

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

La commande charge le contenu spécifié pour le `Content` paramètre à partir du disque. Le contenu doit être similaire à celui de l'exemple suivant.

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

Le système retourne des informations telles que les suivantes.

Linux

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

Windows

```
{
   "ApplicationId"          : "ui_refresh",
   "ConfigurationProfileId" : "UI Refresh",
   "VersionNumber"          : "1",
```

```
   "ContentType"            : "application/json"
}
```

PowerShell

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

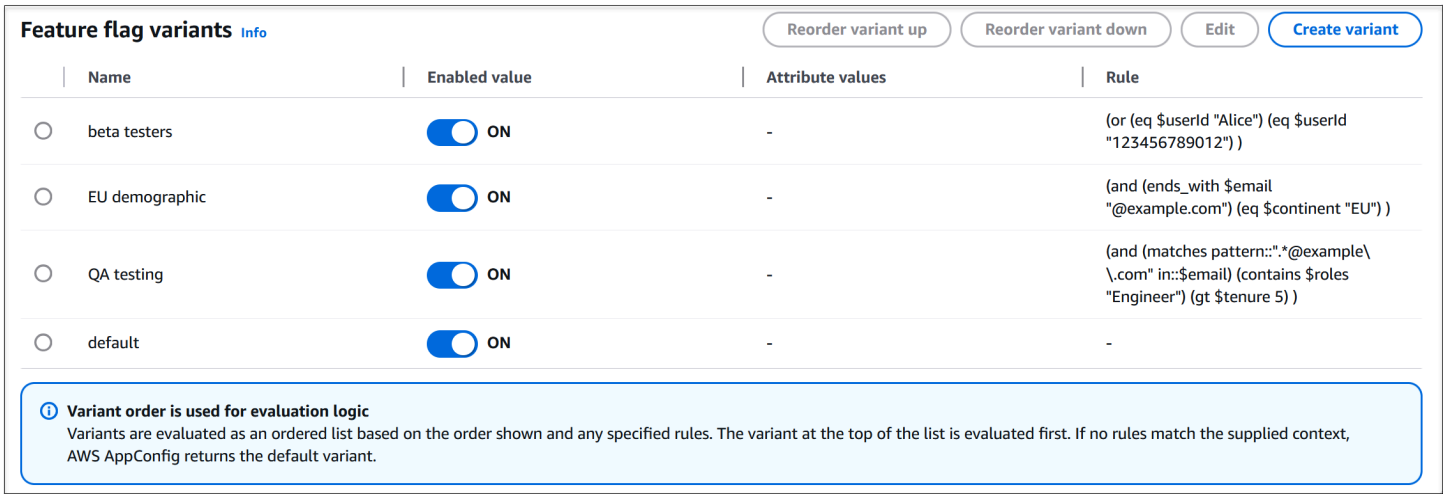
Le `service_returned_content_file` contient vos données de configuration, y compris certaines métadonnées AWS AppConfig générées.

> **Note**
>
> Lorsque vous créez la version de configuration hébergée AWS AppConfig , vérifiez que vos données sont conformes au schéma `AWS.AppConfig.FeatureFlags` JSON. AWS AppConfig confirme également que chaque attribut d'indicateur d'entité de vos données répond aux contraintes que vous avez définies pour ces attributs.

Création d'indicateurs de fonctionnalités à variantes multiples

Les variantes d'indicateurs de fonctionnalité vous permettent de définir un ensemble de valeurs d'indicateur possibles à renvoyer pour une demande. Vous pouvez également configurer différents statuts (activé ou désactivé) pour les indicateurs à variantes multiples. Lorsque vous demandez un indicateur configuré avec des variantes, votre application fournit un contexte qui est AWS AppConfig évalué par rapport à un ensemble de règles définies par l'utilisateur. En fonction du contexte spécifié dans la demande et des règles définies pour la variante, AWS AppConfig renvoie différentes valeurs d'indicateur à l'application.

La capture d'écran suivante montre un exemple d'indicateur de fonctionnalité avec trois variantes définies par l'utilisateur et la variante par défaut.



Rubriques

- Comprendre les concepts d'indicateurs de fonctionnalités à variantes multiples et les cas d'utilisation courants
- Comprendre les règles relatives aux indicateurs de fonctionnalités à variantes multiples
- Création d'un indicateur de fonctionnalité à variantes multiples

Comprendre les concepts d'indicateurs de fonctionnalités à variantes multiples et les cas d'utilisation courants

Pour vous aider à mieux comprendre les variantes d'indicateurs de fonctionnalités, cette section explique les concepts des variantes d'indicateur et les cas d'utilisation courants.

Concepts

- Indicateur de fonctionnalité : type de AWS AppConfig configuration utilisé pour contrôler le comportement d'une fonctionnalité dans une application. Un indicateur possède un statut (activé ou désactivé) et un ensemble facultatif d'attributs contenant des valeurs de chaîne, numériques, booléennes ou matricielles arbitraires.
- Variante d'indicateur d'entité : combinaison spécifique de valeurs d'état et d'attribut appartenant à un indicateur d'entité. Un indicateur de fonctionnalité peut avoir plusieurs variantes.
- Règle de variante : expression définie par l'utilisateur utilisée pour sélectionner une variante d'indicateur de fonctionnalité. Chaque variante possède sa propre règle qui AWS AppConfig évalue si elle doit être renvoyée ou non.
- Variante par défaut : variante spéciale renvoyée lorsqu'aucune autre variante n'est sélectionnée. Tous les indicateurs de fonctionnalités à variantes multiples ont une variante par défaut.

Notez que la variante par défaut doit être la dernière de votre commande de variantes et qu'aucune règle ne peut lui être associée. S'il n'est pas défini en dernier, AWS AppConfig renvoie un `BadRequestException` lorsque vous essayez de créer l'indicateur multivariant.

- Contexte : clés et valeurs définies par l'utilisateur transmises au AWS AppConfig moment de la récupération de la configuration. Les valeurs de contexte sont utilisées lors de l'évaluation des règles pour sélectionner la variante d'indicateur de fonctionnalité à renvoyer.

> **Note**
>
> AWS AppConfig l'agent évalue les règles de variante et détermine quelle règle s'applique à la demande en fonction du contexte fourni. Pour plus d'informations sur la récupération d'indicateurs de fonctionnalités à plusieurs variantes, consultez. Récupération des indicateurs de fonctionnalités de base et multivariantes

Cas d'utilisation courants

Cette section décrit deux cas d'utilisation courants pour les variantes d'indicateurs de fonctionnalité.

Segmentation des utilisateurs

La segmentation des utilisateurs est le processus qui consiste à diviser les utilisateurs en fonction de certains attributs. Par exemple, vous pouvez utiliser des variantes de drapeau pour présenter une fonctionnalité à certains utilisateurs, mais pas à d'autres en fonction de leur nom d'utilisateur, de leur situation géographique, de leur type d'appareil ou de leur fréquence d'achat.

À l'aide de l'exemple de la fréquence des achats, supposons que votre application de commerce propose une fonctionnalité visant à fidéliser les clients. Vous pouvez utiliser des variantes de drapeau pour configurer différents types d'incitations à afficher à un utilisateur en fonction de la date de son dernier achat. Un nouvel utilisateur peut bénéficier d'une petite réduction pour l'encourager à devenir client, tandis qu'un client régulier peut bénéficier d'une réduction plus importante s'il achète quelque chose d'une nouvelle catégorie.

Répartition du trafic

Le fractionnement du trafic consiste à sélectionner une variante d'indicateur aléatoire mais cohérente en fonction d'une valeur de contexte que vous définissez. Par exemple, vous souhaiterez peut-être

réaliser un test dans le cadre duquel un petit pourcentage de vos utilisateurs (identifiés par leur nom d'utilisateur) voit une variante particulière. Vous pouvez également exécuter un déploiement progressif des fonctionnalités dans le cadre duquel une fonctionnalité est d'abord exposée à 5 % de vos utilisateurs, puis à 15 %, puis à 40 %, puis à 100 %, tout en maintenant une expérience utilisateur cohérente tout au long du déploiement.

À l'aide de l'exemple d'expérimentation, vous pouvez utiliser des variantes de drapeau pour tester un nouveau style de bouton pour l'action principale sur la page d'accueil de votre application afin de déterminer s'il génère plus de clics. Pour votre expérience, vous pouvez créer une variante de drapeau avec une règle de répartition du trafic qui sélectionne 5 % des utilisateurs pour voir le nouveau style, tandis que la variante par défaut indique les utilisateurs qui devraient continuer à voir le style existant. Si l'expérience est réussie, vous pouvez augmenter la valeur en pourcentage ou même faire de cette variante la variante par défaut.

Comprendre les règles relatives aux indicateurs de fonctionnalités à variantes multiples

Lorsque vous créez une variante d'indicateur de fonctionnalité, vous spécifiez une règle pour celle-ci. Les règles sont des expressions qui prennent des valeurs de contexte en entrée et produisent un résultat booléen en sortie. Par exemple, vous pouvez définir une règle pour sélectionner une variante d'indicateur pour les utilisateurs bêta, identifiés par leur identifiant de compte, afin de tester une actualisation de l'interface utilisateur. Pour ce scénario, vous devez effectuer les opérations suivantes :

1. Créez un nouveau profil de configuration d'indicateur de fonctionnalité appelé UI Refresh.
2. Créez un nouvel indicateur de fonctionnalité appelé ui_refresh.
3. Modifiez l'indicateur de fonctionnalité après l'avoir créé pour ajouter des variantes.
4. Créez et activez une nouvelle variante appelée BetaUsers.
5. Définissez une règle BetaUsersqui sélectionne la variante si l'identifiant du compte indiqué dans le contexte de la demande figure dans une liste de comptes IDs approuvés pour accéder à la nouvelle expérience bêta.
6. Vérifiez que le statut de la variante par défaut est défini sur Désactivé.

> **Note**
>
> Les variantes sont évaluées sous forme de liste ordonnée en fonction de l'ordre dans lequel elles sont définies dans la console. La variante en haut de la liste est évaluée en premier.

Si aucune règle ne correspond au contexte fourni, AWS AppConfig renvoie la variante par défaut.

Lors du AWS AppConfig traitement de la demande d'indicateur de fonctionnalité, il compare le contexte fourni, qui inclut d'abord l'AccountID (pour cet exemple), à BetaUsers la variante. Si le contexte correspond à la règle de BetaUsers, AWS AppConfig renvoie les données de configuration pour l'expérience bêta. Si le contexte n'inclut pas d'identifiant de compte ou si l'identifiant de compte se termine par une valeur autre que 123, AWS AppConfig renvoie les données de configuration pour la règle par défaut, ce qui signifie que l'utilisateur visualise l'expérience actuelle en production.

Note

Pour plus d'informations sur la récupération d'indicateurs de fonctionnalités à variantes multiples, consultez. Récupération des indicateurs de fonctionnalités de base et multivariantes

Définition de règles pour les indicateurs de fonctionnalités à variantes multiples

Une règle de variante est une expression composée d'un ou de plusieurs opérandes et d'un opérateur. Un opérande est une valeur spécifique utilisée lors de l'évaluation d'une règle. Les valeurs des opérandes peuvent être statiques, telles qu'un nombre littéral ou une chaîne, ou variables, telles que la valeur trouvée dans un contexte ou le résultat d'une autre expression. Un opérateur, tel que « supérieur à », est un test ou une action appliqué à ses opérandes qui produit une valeur. Une expression de règle variante doit produire un « vrai » ou un « faux » pour être valide.

Opérandes

Type	Description	Exemple
Chaîne	Séquence de caractères UTF-8, entre guillemets.	`"apple", "###ë# ##š##"`
Entier	Une valeur entière de 64 bits.	`-7, 42`
Float	Une valeur à virgule flottante IEEE-754 64 bits.	`3.14, 1.234e-5`

Type	Description	Exemple
Horodatage	Un moment précis tel que décrit dans la note du W3C sur les formats de date et d'heure.	`2012-03-04T05:06:07-08:00, 2024-01`
Booléen	Une valeur vraie ou fausse.	`true, false`
Valeur contextuelle	Une valeur paramétrée sous la forme de $ *key* qui est extraite du contexte lors de l'évaluation des règles.	`$country, $userId`

Opérateurs de comparaison

Opérateur	Description	Exemple
eq	Détermine si une valeur de contexte est égale à une valeur donnée.	`(eq $state "Virginia")`
gt	Détermine si une valeur de contexte est supérieure à une valeur donnée.	`(gt $age 65)`
gte	Détermine si une valeur de contexte est supérieure ou égale à une valeur donnée.	`(gte $age 65)`
lt	Détermine si une valeur de contexte est inférieure à une valeur donnée.	`(lt $age 65)`
lte	Détermine si une valeur de contexte est inférieure ou égale à une valeur donnée.	`(lte $age 65)`

Opérateurs logiques

Opérateur	Description	Exemple
and	Détermine si les deux opérandes sont vrais.	`(and (eq $state "Virginia") (gt $age 65))`
or	Détermine si au moins un des opérandes est vrai.	`(or (eq $state "Virginia") (gt $age 65))`
not	Inverse la valeur d'une expression.	`(not (eq $state "Virginia"))`

Opérateurs personnalisés

Opérateur	Description	Exemple
commence_par	Détermine si une valeur de contexte commence par un préfixe donné.	`(begins_with $state "A")`
se termine_par	Détermine si une valeur de contexte se termine par un préfixe donné.	`(ends_with $email "amazon.com")`
contient	Détermine si une valeur de contexte contient une sous-chaîne donnée.	`(contains $promoCode "WIN")`

Opérateur	Description	Exemple
dans	Détermine si une valeur de contexte est contenue dans une liste de constantes.	`(in $userId ["123", "456"])`
allumettes	Détermine si une valeur de contexte correspond à un modèle d'expression régulière donné.	`(matches in::$greeting pattern::"h.*y")`
exists	Détermine si une valeur a été fournie pour une clé de contexte.	`(exists key::"country")`

Opérateur	Description	Exemple
split	Est évalué à `true` pour un pourcentage donné du trafic sur la base d'un hachage cohérent des valeurs contextuelles fournies. Pour une explication détaillée de son `split` fonctionnement, reportez-vous à la section suivante de cette rubrique, Comprendre l'opérateur de division. Notez qu'il `seed` s'agit d'une propriété facultative. Si vous ne le spécifiez pas`seed`, le hachage est cohérent localement, ce qui signifie que le trafic sera réparti de manière cohérente pour cet indicateur, mais les autres indicateurs recevant la même valeur de contexte peuvent répartir le trafic différemm ent. Si elle `seed` est fournie, chaque valeur unique est garantie pour répartir le trafic de manière cohérente entre les indicateurs de fonctionn alités, les profils de configura tion et Comptes AWS.	`(split pct::10 by::$user Id seed::"abc")`

Comprendre l'opérateur de division

La section suivante décrit le comportement de l'`split`opérateur lorsqu'il est utilisé dans différents scénarios. Pour rappel, est `split` évalué à `true` pour un pourcentage donné du trafic sur la base d'un hachage cohérent de la valeur de contexte fournie. Pour mieux comprendre cela, considérez le scénario de référence suivant qui utilise le fractionnement avec deux variantes :

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

Comme prévu, le fait de fournir un ensemble aléatoire de `uniqueId` valeurs produit une distribution approximativement égale à :

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

Si vous ajoutez une troisième variante, mais que vous utilisez le même pourcentage de partage, comme suit :

```


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

Vous vous retrouvez avec la distribution suivante :

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

Cette distribution potentiellement inattendue se produit parce que chaque règle de variante est évaluée dans l'ordre et que la première correspondance détermine la variante renvoyée. Lorsque la règle A est évaluée, 20 % des `uniqueId` valeurs y correspondent, de sorte que la première variante est renvoyée. Ensuite, la règle B est évaluée. Cependant, toutes les `uniqueId` valeurs qui auraient correspondu à la deuxième instruction fractionnée correspondaient déjà à la règle de variante A, donc aucune valeur ne correspond à B. La variante par défaut est renvoyée à la place.

Prenons maintenant un troisième exemple.

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

Comme dans l'exemple précédent, les 20 % premiers des `uniqueId` valeurs correspondent à la règle A. Pour la variante de la règle B, 25 % de toutes les `uniqueId` valeurs correspondraient, mais la plupart des valeurs précédemment correspondantes correspondent à la règle A. Cela laisse 5 % du total pour la variante B, le reste recevant la variante C. La distribution serait la suivante :

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

Utilisation de la **`seed`** propriété

Vous pouvez utiliser cette `seed` propriété pour vous assurer que le trafic est réparti de manière cohérente pour une valeur de contexte donnée, quel que soit l'endroit où l'opérateur de division est utilisé. Si vous ne le spécifiez pas`seed`, le hachage est cohérent localement, ce qui signifie que le trafic sera réparti de manière cohérente pour cet indicateur, mais les autres indicateurs recevant la même valeur de contexte peuvent répartir le trafic différemment. Si elle `seed` est fournie, chaque valeur unique est garantie pour répartir le trafic de manière cohérente entre les indicateurs de fonctionnalités, les profils de configuration et Comptes AWS.

Généralement, les clients utilisent la même `seed` valeur pour toutes les variantes d'un drapeau lorsqu'ils répartissent le trafic sur la même propriété de contexte. Cependant, il peut parfois être judicieux d'utiliser une valeur de départ différente. Voici un exemple qui utilise des valeurs de départ différentes pour les règles A et B :

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

Comme auparavant, 20 % des `uniqueId` valeurs correspondantes correspondent à la règle A. Cela signifie que 80 % des valeurs sont rejetées et testées par rapport à la règle de variante B. Comme le point de départ est différent, il n'y a aucune corrélation entre les valeurs correspondant à A et les valeurs correspondantes B. Il n'y a cependant que 80 % de `uniqueId` valeurs en moins à diviser, 25 % de ce nombre correspondant à la règle B et 75 % non. Cela correspond à la distribution suivante :

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

Création d'un indicateur de fonctionnalité à variantes multiples

Utilisez les procédures décrites dans cette section pour créer des variantes d'un indicateur de fonctionnalité.

Avant de commencer

Notez les informations importantes suivantes.

- Vous pouvez créer des variantes d'indicateurs de fonctionnalités existants en les modifiant. Vous ne pouvez pas créer de variantes d'un nouvel indicateur de fonctionnalité lorsque vous créez un nouveau profil de configuration. Vous devez d'abord terminer le processus de création du nouveau profil de configuration. Après avoir créé le profil de configuration, vous pouvez ajouter des variantes à n'importe quel indicateur du profil de configuration. Pour plus d'informations sur la création d'un nouveau profil de configuration, consultezCréation d'un profil de configuration d'indicateur de fonctionnalité dans AWS AppConfig.
- Pour récupérer les données des variantes des indicateurs de fonctionnalité pour les plateformes de calcul Amazon EC2, Amazon ECS et Amazon EKS, vous devez utiliser la version 2.0.4416 ou ultérieure de l' AWS AppConfig agent.
- Pour des raisons de performances, AWS CLI et pour des raisons de performance, le SDK appelle à AWS AppConfig ne pas récupérer les données des variantes. Pour plus d'informations sur AWS AppConfig l'agent, consultezComment utiliser l' AWS AppConfig agent pour récupérer les données de configuration.
- Lorsque vous créez une variante d'indicateur de fonctionnalité, vous spécifiez une règle pour celle-ci. Les règles sont des expressions qui prennent le contexte de la demande en entrée et produisent un résultat booléen en sortie. Avant de créer des variantes, vérifiez les opérandes et les opérateurs pris en charge pour connaître les règles relatives aux variantes de marquage. Vous pouvez créer des règles avant de créer des variantes. Pour de plus amples informations, veuillez consulter Comprendre les règles relatives aux indicateurs de fonctionnalités à variantes multiples.

Rubriques

- Création d'un indicateur de fonctionnalité à plusieurs variantes (console)
- Création d'un indicateur de fonctionnalité à plusieurs variantes (ligne de commande)

Création d'un indicateur de fonctionnalité à plusieurs variantes (console)

La procédure suivante décrit comment créer un indicateur de fonctionnalité multivariant pour un profil de configuration existant à l'aide de la AWS AppConfig console. Vous pouvez également modifier les indicateurs de fonction existants pour créer des variantes.

Pour créer un indicateur de fonctionnalité à plusieurs variantes

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis choisissez une application.
3. Dans l'onglet Profils de configuration et indicateurs de fonctionnalités, choisissez un profil de configuration d'indicateur de fonctionnalité existant.
4. Dans la section Drapeaux, choisissez Ajouter un nouveau drapeau.
5. Dans la section Définition de l'indicateur de fonctionnalité, pour Nom du drapeau, entrez un nom.
6. Pour la touche Drapeau, entrez un identifiant de drapeau pour distinguer les indicateurs au sein d'un même profil de configuration. Les drapeaux d'un même profil de configuration ne peuvent pas avoir la même clé. Une fois le drapeau créé, vous pouvez modifier le nom du drapeau, mais pas la clé du drapeau.
7. (Facultatif) Dans le champ Description, entrez les informations relatives à cet indicateur.
8. Dans la section Variantes, choisissez Indicateur multivariant.
9. (Facultatif) Dans la section Attributs du drapeau de fonction, choisissez Définir l'attribut. Les attributs vous permettent de fournir des valeurs supplémentaires dans votre drapeau. Pour plus d'informations sur les attributs et les contraintes, consultezComprendre les attributs des indicateurs de fonctionnalité.

    a. Pour Clé, spécifiez une touche indicateur et choisissez son type dans la liste Type. Pour plus d'informations sur les options prises en charge pour les champs Valeur et Contraintes, consultez la section précédemment référencée sur les attributs.
    b. Sélectionnez Valeur requise pour indiquer si une valeur d'attribut est requise.
    c. Choisissez Définir un attribut pour ajouter des attributs supplémentaires.
    d. Choisissez Appliquer pour enregistrer les modifications d'attributs.
10. Dans la section Variantes du drapeau de fonctionnalité, choisissez Créer une variante.

    a. Dans Nom de la variante, entrez un nom.
    b. Utilisez le bouton Valeur activée pour activer la variante.

c. Dans la zone de texte Règle, entrez une règle.

d. Utilisez les options Créer une variante > Créer une variante ci-dessus ou Créer une variante ci-dessous pour créer des variantes supplémentaires pour cet indicateur.

e. Dans la section Variante par défaut, utilisez le bouton Valeur activée pour activer la variante par défaut. Vous pouvez éventuellement fournir des valeurs pour les attributs définis à l'étape 10.

f. Choisissez Appliquer.

11. Vérifiez les détails du drapeau et de ses variantes, puis choisissez Créer un drapeau.

Pour plus d'informations sur le déploiement de votre nouvel indicateur de fonctionnalité avec des variantes, consultezDéploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig.

Création d'un indicateur de fonctionnalité à plusieurs variantes (ligne de commande)

La procédure suivante décrit comment utiliser le AWS Command Line Interface (sous Linux ou Windows) ou les Outils pour Windows PowerShell afin de créer un indicateur de fonctionnalité à plusieurs variantes pour un profil de configuration existant. Vous pouvez également modifier les indicateurs de fonction existants pour créer des variantes.

Avant de commencer

Effectuez les tâches suivantes avant de créer un indicateur de fonctionnalité à variantes multiples à l'aide du AWS CLI.

- Créez un profil de configuration d'indicateur de fonctionnalité. Pour de plus amples informations, veuillez consulter Création d'un profil de configuration d'indicateur de fonctionnalité dans AWS AppConfig.
- Effectuez une mise à jour vers la dernière version du AWS CLI. Pour plus d'informations, voir Installer ou mettre à jour la dernière version du AWS CLI dans le guide de AWS Command Line Interface l'utilisateur.

Pour créer un indicateur de fonctionnalité à plusieurs variantes

1. Créez un fichier de configuration sur votre machine locale qui spécifie les détails de l'indicateur multivariant que vous souhaitez créer. Enregistrez le fichier avec une extension de `.json` fichier.

Le fichier doit respecter le schéma AWS.AppConfig.FeatureFlagsJSON. Le contenu du schéma de votre fichier de configuration sera similaire à ce qui suit.

```
{
  "flags": {
    "FLAG_NAME": {
      "attributes": {
          "ATTRIBUTE_NAME": {
          "constraints": {
            "type": "CONSTRAINT_TYPE"
          }
        }
      },
      "description": "FLAG_DESCRIPTION",
      "name": "VARIANT_NAME"
    }
  },
  "values": {
    "VARIANT_VALUE_NAME": {
      "_variants": [
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
        },
        {
          "attributeValues": {
            "ATTRIBUTE_NAME": BOOLEAN
          },
          "enabled": BOOLEAN,
          "name": "VARIANT_NAME",
          "rule": "VARIANT_RULE"
```

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

Voici un exemple avec trois variantes et la variante par défaut.

```
{
  "flags": {
    "ui_refresh": {
      "attributes": {
        "dark_mode_support": {
          "constraints": {
            "type": "boolean"
          }
        }
      },
      "description": "A release flag used to release a new UI",
      "name": "UI Refresh"
    }
  },
  "values": {
    "ui_refresh": {
      "_variants": [
        {
          "attributeValues": {
            "dark_mode_support": true
          },
          "enabled": true,
          "name": "QA",
          "rule": "(ends_with $email \"qa-testers.mycompany.com\")"
        },
        {
```

```
          "attributeValues": {
            "dark_mode_support": true
          },
          "enabled": true,
          "name": "Beta Testers",
          "rule": "(exists key::\"opted_in_to_beta\")"
        },
        {
          "attributeValues": {
            "dark_mode_support": false
          },
          "enabled": true,
          "name": "Sample Population",
          "rule": "(split pct::10 by::$email)"
        },
        {
          "attributeValues": {
            "dark_mode_support": false
          },
          "enabled": false,
          "name": "Default Variant"
        }
      ]
    }
  },
  "version": "1"
}
```

2. Utilisez l'`CreateHostedConfigurationVersion`API pour enregistrer les données de configuration de votre indicateur de fonctionnalité dans AWS AppConfig.

   Linux

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

Windows

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

PowerShell

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

Le `service_returned_content_file` contient vos données de configuration, y compris certaines métadonnées AWS AppConfig générées.

> **Note**
>
> Lorsque vous créez la version de configuration hébergée AWS AppConfig , vérifiez que vos données sont conformes au schéma `AWS.AppConfig.FeatureFlags`JSON. AWS AppConfig confirme également que chaque attribut d'indicateur d'entité de vos données répond aux contraintes que vous avez définies pour ces attributs.

Comprendre la référence de type pour AWS.AppConfig.FeatureFlags

Utilisez le schéma `AWS.AppConfig.FeatureFlags` JSON comme référence pour créer les données de configuration de vos indicateurs de fonctionnalité.

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

```
      "type": "object",
      "properties": {
        "version": {
          "$ref": "#/definitions/flagSchemaVersions"
        },
        "flags": {
          "$ref": "#/definitions/flagDefinitions"
        },
        "values": {
          "$ref": "#/definitions/flagValues"
        }
      },
      "required": ["version"],
      "additionalProperties": false
    },
    "flagDefinitions": {
      "type": "object",
      "patternProperties": {
        "^[a-z][a-zA-Z\\d_-]{0,63}$": {
          "$ref": "#/definitions/flagDefinition"
        }
      },
      "additionalProperties": false
    },
    "flagDefinition": {
      "type": "object",
      "properties": {
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "description": {
          "$ref": "#/definitions/customerDefinedDescription"
        },
        "_createdAt": {
          "type": "string"
        },
        "_updatedAt": {
          "type": "string"
        },
        "_deprecation": {
          "type": "object",
          "properties": {
            "status": {
              "type": "string",
```

```
            "enum": ["planned"]
          },
          "date": {
            "type": "string",
            "format": "date"
          }
        },
       "additionalProperties": false
      },
      "attributes": {
        "$ref": "#/definitions/attributeDefinitions"
      }
    },
    "additionalProperties": false
  },
  "attributeDefinitions": {
    "type": "object",
    "patternProperties": {
      "^[a-z][a-zA-Z\\d_-]{0,63}$": {
        "$ref": "#/definitions/attributeDefinition"
      }
    },
    "maxProperties": 25,
    "additionalProperties": false
  },
  "attributeDefinition": {
    "type": "object",
    "properties": {
      "description": {
        "$ref": "#/definitions/customerDefinedDescription"
      },
      "constraints": {
        "oneOf": [
          { "$ref": "#/definitions/numberConstraints" },
          { "$ref": "#/definitions/stringConstraints" },
          { "$ref": "#/definitions/arrayConstraints" },
          { "$ref": "#/definitions/boolConstraints" }
        ]
      }
    },
    "additionalProperties": false
  },
  "flagValues": {
    "type": "object",
```

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

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

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

```
    "boolConstraints": {
      "type": "object",
      "properties": {
        "type": {
          "type": "string",
          "enum": ["boolean"]
        },
        "required": {
          "type": "boolean"
        }
      },
      "required": ["type"],
      "additionalProperties": false
    },
    "elementConstraints": {
      "oneOf": [
        { "$ref": "#/definitions/numberConstraints" },
        { "$ref": "#/definitions/stringConstraints" }
      ]
    },
    "variant": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
        },
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "rule": {
          "type": "string",
          "maxLength": 16384
        },
        "attributeValues": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d_-]{0,63}$": {
              "$ref": "#/definitions/attributeValue"
            }
          },
          "maxProperties": 25,
          "additionalProperties": false
        }
      },
```

```
      "required": ["name", "enabled"],
      "additionalProperties": false
    },
    "customerDefinedName": {
      "type": "string",
      "pattern": "^[^\\n]{1,64}$"
    },
    "customerDefinedDescription": {
      "type": "string",
      "maxLength": 1024
    },
    "flagSchemaVersions": {
      "type": "string",
      "enum": ["1"]
    }
  },
  "type": "object",
  "$ref": "#/definitions/flagSetDefinition",
  "additionalProperties": false
}
```

> **Important**
>
> Pour récupérer les données de configuration des indicateurs de fonctionnalité, votre application doit appeler l'`GetLatestConfiguration`API. Vous ne pouvez pas récupérer les données de configuration des indicateurs de fonctionnalité en appelant`GetConfiguration`, ce qui est obsolète. Pour plus d'informations, consultez GetLatestConfiguration dans la Référence d'API AWS AppConfig .

Lorsque votre application appelle GetLatestConfigurationet reçoit une configuration nouvellement déployée, les informations qui définissent vos indicateurs et attributs de fonctionnalités sont supprimées. Le JSON simplifié contient une carte de clés correspondant à chacune des clés d'indicateur que vous avez spécifiées. Le JSON simplifié contient également des valeurs mappées de `true` ou `false` pour l'`enabled`attribut. Si un drapeau est `enabled` défini sur`true`, tous les attributs du drapeau seront également présents. Le schéma JSON suivant décrit le format de la sortie JSON.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "patternProperties": {
```

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

```
                {
                  "type": "number"
                }
              ]
            }
          }
        ]
      }
    ],
    "additionalProperties": false
  }
 }
}
```

Enregistrer une version précédente d'un indicateur de fonctionnalité dans une nouvelle version

Lorsque vous mettez à jour un indicateur de fonctionnalité, vos modifications AWS AppConfig sont automatiquement enregistrées dans une nouvelle version. Si vous souhaitez utiliser une version précédente de l'indicateur de fonctionnalité, vous devez la copier dans une version brouillon, puis l'enregistrer. Vous ne pouvez pas modifier et enregistrer les modifications apportées à une version précédente d'un drapeau sans l'enregistrer dans une nouvelle version.

Pour modifier une version précédente d'un indicateur de fonctionnalité et l'enregistrer dans une nouvelle version

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis choisissez l'application avec l'indicateur de fonctionnalité que vous souhaitez modifier et enregistrer dans une nouvelle version.
3. Dans l'onglet Profils de configuration et indicateurs de fonctionnalité, choisissez le profil de configuration avec l'indicateur de fonctionnalité que vous souhaitez modifier et enregistrer dans une nouvelle version.
4. Dans l'onglet Feature flags, utilisez la liste des versions pour choisir la version que vous souhaitez modifier et enregistrer dans une nouvelle version.
5. Choisissez Copier vers une version brouillon.
6. Dans le champ Libellé de version, entrez un nouveau libellé (facultatif, mais recommandé).

7. Dans le champ Description de la version, entrez une nouvelle description (facultatif, mais recommandé).
8. Choisissez Enregistrer la version.
9. Choisissez Démarrer le déploiement pour déployer la nouvelle version.

Création d'un profil de configuration sous forme libre dans AWS AppConfig

Les données de configuration sont un ensemble de paramètres qui influencent le comportement de votre application. Un profil de configuration inclut, entre autres, un URI qui permet de AWS AppConfig localiser vos données de configuration dans leur emplacement enregistré et un type de configuration. Avec les profils de configuration libres, vous pouvez stocker vos données dans le magasin de configuration AWS AppConfig hébergé ou dans l'un des outils suivants Services AWS ou dans les outils Systems Manager :

Emplacement	Types de fichier pris en charge
AWS AppConfig magasin de configuration hébergé	YAML, JSON et texte s'ils sont ajoutés à l'aide du AWS Management Console. Tout type de fichier s'il est ajouté à l'aide de AWS AppConfig CreateHostedConfigurationVersionl'action API.
Amazon Simple Storage Service (Amazon S3)	N'importe quel compte
AWS CodePipeline	Pipeline (tel que défini par le service)
AWS Secrets Manager	Secret (tel que défini par le service)
AWS Systems Manager Magasin de paramètres	Paramètres de chaîne standard et sécurisés (tels que définis par Parameter Store)
AWS Systems Manager magasin de documents (documents SSM)	YAML, JSON, texte

Un profil de configuration peut également inclure des validateurs facultatifs pour garantir l'exactitude syntaxique et sémantique de vos données de configuration. AWS AppConfig effectue une vérification à l'aide des validateurs lorsque vous démarrez un déploiement. Si des erreurs sont détectées, le déploiement s'arrête avant d'apporter des modifications aux cibles de la configuration.

> **Note**
>
> Dans la mesure du possible, nous vous recommandons d'héberger vos données de configuration dans le magasin de configuration AWS AppConfig hébergé, qui offre le plus de fonctionnalités et d'améliorations.

Pour les configurations libres stockées dans le magasin de configuration AWS AppConfig hébergé ou dans les documents SSM, vous pouvez créer la configuration libre en utilisant la console Systems Manager au moment de créer un profil de configuration. Le processus est décrit plus loin dans cette rubrique.

Pour les configurations libres stockées dans Parameter Store, Secrets Manager ou Amazon S3, vous devez d'abord créer le paramètre, le secret ou l'objet et le stocker dans le magasin de configuration approprié. Après avoir enregistré les données de configuration, utilisez la procédure décrite dans cette rubrique pour créer le profil de configuration.

Rubriques

Comprendre les validateurs

Lorsque vous créez un profil de configuration, vous avez la possibilité de spécifier jusqu'à deux validateurs. Un validateur garantit que vos données de configuration sont syntaxiquement et sémantiquement correctes. Si vous envisagez d'utiliser un validateur, vous devez le créer avant de créer le profil de configuration. AWS AppConfig prend en charge les types de validateurs suivants :

- AWS Lambda fonctions : prise en charge pour les indicateurs de fonctionnalités et les configurations de forme libre.
- Schéma JSON : pris en charge pour les configurations de formulaire libre. (valide AWS AppConfig automatiquement les indicateurs de fonctionnalité par rapport à un schéma JSON.)

Rubriques

- AWS Lambda validateurs de fonctions
- Validateurs de schéma JSON

AWS Lambda validateurs de fonctions

Les validateurs de fonctions Lambda doivent être configurés avec le schéma d'événements suivant. AWS AppConfig utilise ce schéma pour appeler la fonction Lambda. Le contenu est une chaîne codée en base64, et l'URI est une chaîne.

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

AWS AppConfig vérifie que l'en-tête `X-Amz-Function-Error` Lambda est défini dans la réponse. Lambda définit cet en-tête si la fonction génère une exception. Pour plus d'informations`X-Amz-Function-Error`, consultez la section Gestion des erreurs et tentatives automatiques AWS Lambda dans le Guide du AWS Lambda développeur.

Voici un exemple simple de code de réponse Lambda pour une validation réussie.

```
import json

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

Voici un exemple simple de code de réponse Lambda pour une validation infructueuse.

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

Voici un autre exemple qui n'est valide que si le paramètre de configuration est un nombre premier.

```
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 appelle votre Lambda de validation lorsque vous appelez les opérations `StartDeployment` et `ValidateConfigurationActivity` API. Vous devez fournir des `appconfig.amazonaws.com` autorisations pour appeler votre Lambda. Pour plus d'informations, consultez la section Autorisation de l'accès aux fonctions aux AWS services. AWS AppConfig limite la durée d'exécution Lambda de validation à 15 secondes, y compris la latence de démarrage.

Validateurs de schéma JSON

Si vous créez une configuration dans un document SSM, vous devez spécifier ou créer un schéma JSON pour cette configuration. Un schéma JSON définit les propriétés autorisées pour chaque paramètre de configuration d'application. Ce schéma JSON fonctionne comme un ensemble de règles visant à garantir que les nouveaux paramètres de configuration ou les paramètres de configuration mis à jour sont conformes aux bonnes pratiques requises par votre application. Voici un exemple.

```
    {
      "$schema": "http://json-schema.org/draft-04/schema#",
```

```
    "title": "$id$",
    "description": "BasicFeatureToggle-1",
    "type": "object",
    "additionalProperties": false,
    "patternProperties": {
        "[^\\s]+$": {
            "type": "boolean"
        }
    },
    "minProperties": 1
}
```

Lorsque vous créez une configuration à partir d'un document SSM, le système vérifie automatiquement que la configuration est conforme aux exigences du schéma. Si tel n'est pas le cas, AWS AppConfig renvoie une erreur de validation.

Important

Notez les informations importantes suivantes concernant les validateurs de schéma JSON :

- Les données de configuration stockées dans les documents SSM doivent être validées par rapport à un schéma JSON associé avant de pouvoir ajouter la configuration au système. Les paramètres SSM ne nécessitent pas de méthode de validation, mais nous vous recommandons de créer un contrôle de validation pour les configurations de paramètres SSM nouvelles ou mises à jour en utilisant. AWS Lambda
- Une configuration dans un document SSM utilise le type de `ApplicationConfiguration` document. Le schéma JSON correspondant utilise le type de `ApplicationConfigurationSchema` document.
- AWS AppConfig prend en charge le schéma JSON version 4.X pour le schéma en ligne. Si la configuration de votre application nécessite une version différente du schéma JSON, vous devez créer un validateur Lambda.

Comprendre les quotas et les limites du magasin de configuration

Les magasins de configuration pris en charge par AWS AppConfig sont soumis aux quotas et limites suivants.

	AWS AppConfig magasin de configuration hébergé	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Boutique de documents Systems Manager	AWS CodePipeline
Limite de taille de la configuration	2 Mo par défaut, 4 Mo maximum	2 Mo Appliqué par AWS AppConfig, et non par S3	4 ko (offre gratuite) / 8 k (paramètres avancés)	64 Ko	64 Ko	2 Mo Appliqué par AWS AppConfig, et non CodePipeline
Limite de stockage des ressources	1 Go	Illimité	10 000 paramètres (offre gratuite) / 10(paramètres (paramètres avancés)	500 000	500 documer	Limité par le nombre de profils de configuration par application (100 profils par application)
Chiffrement côté serveur	Oui	SSE-S3, SSE-KMS	Oui	Oui	Non	Oui
AWS CloudFormation soutien	Oui	Ne permet pas de créer ou de mettre	Oui	Oui	Non	Oui

	AWS AppConfig magasin de configuration hébergé	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Boutique de documents Systems Manager	AWS CodePipeline
		à jour des données				
Tarification	Free	Voir les tarifs d'Amazon S3	Voir les AWS Systems Manager tarifs	Voir les AWS Secrets Manager tarifs	Free	Voir les AWS CodePipeline tarifs

Comprendre le magasin de configuration AWS AppConfig hébergé

AWS AppConfig inclut un magasin de configuration interne ou hébergé. Les configurations doivent être inférieures ou égales à 2 Mo. Le magasin de configuration AWS AppConfig hébergé offre les avantages suivants par rapport aux autres options du magasin de configuration.

- Vous n'avez pas besoin de configurer d'autres services tels qu'Amazon Simple Storage Service (Amazon S3) ou Parameter Store.
- Vous n'avez pas besoin de configurer les autorisations AWS Identity and Access Management (IAM) pour utiliser le magasin de configuration.
- Vous pouvez stocker des configurations dans YAML, JSON ou sous forme de documents texte.
- L'utilisation du magasin est gratuite.
- Vous pouvez créer une configuration et l'ajouter au magasin lorsque vous créez un profil de configuration.

Comprendre les configurations stockées dans Amazon S3

Vous pouvez stocker les configurations dans un bucket Amazon Simple Storage Service (Amazon S3). Lorsque vous créez le profil de configuration, vous spécifiez l'URI pour un seul objet S3 dans un

compartiment. Vous spécifiez également le nom de ressource Amazon (ARN) d'un rôle AWS Identity and Access Management (IAM) qui AWS AppConfig autorise l'obtention de l'objet. Avant de créer un profil de configuration pour un objet Amazon S3, tenez compte des restrictions suivantes.

Restriction	Détails
Size	La taille maximale des configurations stockées en tant qu'objets S3 est de 1 Mo.
Chiffrement de l'objet	Un profil de configuration peut cibler des objets chiffrés SSE-S3 et SSE-KMS.
Classes de stockage	AWS AppConfig prend en charge les classes de stockage S3 suivantes : `STANDARDINTELLIGENT_TIERIN G` ,`REDUCED_REDUNDANCY` ,`STANDARD_ IA` , et`ONEZONE_IA` . Les classes suivantes ne sont pas prises en charge : toutes les classes S3 Glacier (`GLACIER` et `DEEP_ARCH IVE`).
Gestion des versions	AWS AppConfig nécessite que l'objet S3 utilise le versionnement.

Configuration des autorisations pour une configuration stockée en tant qu'objet Amazon S3

Lorsque vous créez un profil de configuration pour une configuration stockée en tant qu'objet S3, vous devez spécifier un ARN pour un rôle IAM qui AWS AppConfig autorise l'accès à l'objet. Votre rôle doit inclure les autorisations suivantes :

Autorisations pour accéder à l'objet S3

- s3 : GetObject
- s3 : GetObjectVersion

Autorisations pour répertorier les compartiments S3

s3 : ListAllMyBuckets

Autorisations pour accéder au compartiment S3 où l'objet est stocké

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

Complétez la procédure suivante pour créer un rôle qui permet d' AWS AppConfig obtenir une configuration stockée dans un objet S3.

Création de la politique IAM pour accéder à un objet S3

Utilisez la procédure suivante pour créer une politique IAM qui permet d' AWS AppConfig obtenir une configuration stockée dans un objet S3.

Pour créer une politique IAM pour accéder à un objet S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Politiques, puis Créer une politique.
3. Sur la page Créer une politique, choisissez l'onglet JSON.
4. Mettez à jour l'exemple de stratégie suivant avec les informations sur votre compartiment S3 et votre objet de configuration. Collez ensuite la stratégie dans le champ de texte de l'onglet JSON . Remplacez *placeholder values* par vos propres informations.

   JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/my-configurations/my-
configuration.json"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

5. Choisissez Examiner une politique.
6. Sur la page Review policy (Examiner une stratégie), saisissez un nom dans la zone Name (Nom), puis saisissez une description.
7. Choisissez Create Policy (Créer une politique). Le système vous renvoie à la page Rôles.

Création du rôle IAM pour accéder à un objet S3

Utilisez la procédure suivante pour créer un rôle IAM qui permet d' AWS AppConfig obtenir une configuration stockée dans un objet S3.

Pour créer un rôle IAM pour accéder à un objet Amazon S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Rôles, puis Créer un rôle.
3. Dans la section Sélectionner le type d'entité de confiance, sélectionnez AWS service.
4. Dans la section Choisir un cas d'utilisation, sous Cas d'utilisation courants, sélectionnez EC2, puis cliquez sur Suivant : Autorisations.
5. Dans la page Attach permissions policy (Attacher des stratégies d'autorisations) dans la zone de recherche, saisissez le nom de la stratégie que vous avez créée lors de la procédure précédente.

6. Choisissez la stratégie, puis sélectionnez Next: Tags (Suivant : Balises).
7. Sur la page Ajouter des balises (facultatif), entrez une clé et une valeur facultative, puis choisissez Suivant : Révision.
8. Sur la page Review (Vérification), saisissez un nom dans le champ Role name (Nom du rôle), puis saisissez une description.
9. Sélectionnez Créer un rôle. Le système vous renvoie à la page Rôles.
10. Sur la page Rôles, sélectionnez le rôle que vous venez de créer pour ouvrir la page Récapitulatif. Notez le Nom du rôle et l'ARN de rôle. Vous spécifierez l'ARN de rôle lors de la création du profil de configuration plus loin dans cette rubrique.

Création d'une relation d'approbation

Utilisez la procédure suivante pour configurer le rôle que vous venez de créer pour approuver AWS AppConfig.

Pour ajouter une relation d'approbation

1. Dans la page Récapitulatif du rôle que vous venez de créer, choisissez l'onglet Relations d'approbation, puis choisissez Modifier la relation d'approbation.
2. Supprimez `"ec2.amazonaws.com"` et ajoutez `"appconfig.amazonaws.com"` comme le montre l'exemple suivant.

   JSON

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

3. Choisissez Mettre à jour la politique d'approbation.

Création d'un profil AWS AppConfig de configuration libre (console)

Utilisez la procédure suivante pour créer un profil de configuration AWS AppConfig libre et (éventuellement) une configuration libre à l'aide de la console. AWS Systems Manager

Pour créer un profil de configuration libre

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis choisissez une application que vous avez créée dansCréation d'un espace de noms pour votre application dans AWS AppConfig.
3. Choisissez l'onglet Profils de configuration et indicateurs de fonctionnalités, puis choisissez Créer une configuration.
4. Dans la section Options de configuration, choisissez Configuration Freeform.
5. Dans Nom du profil de configuration, entrez un nom pour le profil de configuration.
6. (Facultatif) Développez la description et entrez une description.
7. (Facultatif) Développez les options supplémentaires et effectuez les opérations suivantes, si nécessaire.
   a. Dans la section Associer des extensions, choisissez une extension dans la liste.
   b. Dans la section Balises, choisissez Ajouter une nouvelle balise, puis spécifiez une clé et une valeur facultative.
8. Choisissez Suivant.
9. Sur la page Spécifier les données de configuration, dans la section Définition de la configuration, choisissez une option.
10. Renseignez les champs correspondant à l'option que vous avez sélectionnée, comme décrit dans le tableau suivant.

Option sélectionnée	Détails
AWS AppConfig configuration hébergée	Choisissez Text, JSON ou YAML, puis entrez votre configuration dans le champ. Passez à l'étape 12 de cette procédure.

Option sélectionnée	Détails
Objet Amazon S3	Entrez l'URI de l'objet dans le champ source de l'objet S3 et passez à l'étape 11 de cette procédure.
AWS CodePipeline	Choisissez Next et passez à l'étape 12 de cette procédure.
Secret du Gestionnaire de Secrets	Choisissez le secret dans la liste, passez à l'étape 11 de cette procédure.
AWS Systems Manager paramètre	Choisissez le paramètre dans la liste et passez à l'étape 11 de cette procédure.
AWS Systems Manager document	1. Choisissez un document dans la liste ou choisissez Créer un nouveau document. 2. Si vous choisissez Créer un nouveau document, entrez un nom dans le champ Nom du document. Vous pouvez éventuellement développer le nom de la version et saisir un nom pour la version du document. 3. Pour le schéma de configuration de l'application, choisissez le schéma JSON dans la liste ou choisissez Créer un schéma. Si vous choisissez Create schema, Systems Manager ouvre la page Create schema. Entrez les détails du schéma, puis choisissez Créer un schéma de configuration d'application. 4. Dans la section Contenu, choisissez YAML ou JSON, puis entrez les données de configuration dans le champ.

11. Dans la section Rôle de service, choisissez Nouveau rôle de service pour AWS AppConfig créer le rôle IAM qui donne accès aux données de configuration. AWS AppConfig remplit automatiquement le champ Nom du rôle en fonction du nom que vous avez saisi précédemment.

Vous pouvez également choisir Rôle de service existant. Choisissez le rôle dans la liste Role ARN (ARN du rôle).

12. Sur la page Ajouter des validateurs, vous pouvez éventuellement sélectionner Schéma JSON ou AWS Lambda. Si vous choisissez JSON Schema (Schéma JSON), saisissez le schéma JSON dans le champ. Si vous choisissez AWS Lambda, choisissez également la fonction Amazon Resource Name (ARN) et la version dans la liste.

    Important

    Les données de configuration stockées dans les documents SSM doivent être validées par rapport à un schéma JSON associé avant de pouvoir ajouter la configuration au système. Les paramètres SSM ne nécessitent pas de méthode de validation, mais nous vous recommandons de créer un contrôle de validation pour les configurations de paramètres SSM nouvelles ou mises à jour en utilisant. AWS Lambda

13. Choisissez Suivant.
14. Sur la page Réviser et enregistrer, choisissez Enregistrer et poursuivez le déploiement.

Important

Si vous avez créé un profil de configuration pour AWS CodePipeline, vous devez créer un pipeline dans CodePipeline lequel vous AWS AppConfig spécifiez le fournisseur de déploiement. Vous n'avez pas besoin de jouerDéploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig. Toutefois, vous devez configurer un client pour recevoir les mises à jour de configuration des applications, comme décrit dansRécupération des données de configuration sans AWS AppConfig agent. Pour plus d'informations sur la création d'un pipeline spécifié AWS AppConfig comme fournisseur de déploiement, voir Tutoriel : Création d'un pipeline utilisé en AWS AppConfig tant que fournisseur de déploiement dans le guide de AWS CodePipeline l'utilisateur.

Passez à Déploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig.

Création d'un profil de configuration AWS AppConfig libre (ligne de commande)

La procédure suivante décrit comment utiliser AWS CLI (sous Linux ou Windows) ou comment Outils AWS pour PowerShell créer un profil de configuration AWS AppConfig libre. Si vous préférez, vous pouvez AWS CloudShell exécuter les commandes répertoriées ci-dessous. Pour plus d'informations, consultez Présentation d' AWS CloudShell dans le Guide de l'utilisateur AWS CloudShell .

> **Note**
>
> Pour les configurations de forme libre hébergées dans le magasin de configuration AWS AppConfig hébergé, vous devez spécifier `hosted` l'URI de localisation.

Pour créer un profil de configuration à l'aide du AWS CLI

1. Ouvrez le AWS CLI.
2. Exécutez la commande suivante pour créer un profil de configuration libre.

   Linux

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

   Windows

   ```
   aws appconfig create-configuration-profile ^
     --application-id APPLICATION_ID ^
     --name NAME ^
     --description CONFIGURATION_PROFILE_DESCRIPTION ^
     --location-uri CONFIGURATION_URI or hosted  ^
     --retrieval-role-arn IAM_ROLE_ARN ^
     --tags TAGS ^
   ```

```
  --validators "Content=SCHEMA_CONTENT or LAMBDA_FUNCTION_ARN,Type=JSON_SCHEMA
 or LAMBDA"
```

PowerShell

```
New-APPCConfigurationProfile `
  -Name NAME `
  -ApplicationId APPLICATION_ID `
  -Description CONFIGURATION_PROFILE_DESCRIPTION `
  -LocationUri CONFIGURATION_URI or hosted `
  -RetrievalRoleArn IAM_ROLE_ARN `
  -Tag TAGS `
  -Validators "Content=SCHEMA_CONTENT or LAMBDA_FUNCTION_ARN,Type=JSON_SCHEMA
 or LAMBDA"
```

Important

Notez les informations importantes suivantes.

- Si vous avez créé un profil de configuration pour AWS CodePipeline, vous devez créer un pipeline dans CodePipeline lequel vous AWS AppConfig spécifiez le fournisseur de déploiement. Vous n'avez pas besoin de jouerDéploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig. Toutefois, vous devez configurer un client pour recevoir les mises à jour de configuration des applications, comme décrit dansRécupération des données de configuration sans AWS AppConfig agent. Pour plus d'informations sur la création d'un pipeline spécifié AWS AppConfig comme fournisseur de déploiement, voir Tutoriel : Création d'un pipeline utilisé en AWS AppConfig tant que fournisseur de déploiement dans le guide de AWS CodePipeline l'utilisateur.
- Si vous avez créé une configuration dans le magasin de configuration AWS AppConfig hébergé, vous pouvez créer de nouvelles versions de la configuration à l'aide des opérations CreateHostedConfigurationVersiond'API. Pour consulter AWS CLI les détails et les exemples de commandes relatifs à cette opération d'API, reportez-vous create-hosted-configuration-versionà la référence des AWS CLI commandes.

Passez à Déploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig.

Création d'un profil de configuration pour les sources de données non natives

AWS AppConfig prend en charge le déploiement des données de configuration à partir de la plupart des magasins de données. AWS AppConfig Supporte nativement le déploiement des données de configuration stockées dans les services suivants :

- Le magasin de configuration AWS AppConfig hébergé
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager Magasin de paramètres
- Boutique de documents Systems Manager
- AWS CodePipeline

Si vos données de configuration sont stockées dans un emplacement qui n'est pas pris en charge de manière native AWS AppConfig, vous pouvez créer une AWS AppConfig extension pour récupérer vos données depuis leur source. Par exemple, à l'aide d'une AWS AppConfig extension, vous pouvez récupérer les données de configuration stockées dans Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB (DynamoDB) ou dans un dépôt local GitHub GitLab, pour n'en citer que quelques-unes. En implémentant une extension, vous pouvez tirer parti de la AWS AppConfig sécurité et des DevOps améliorations apportées à vos applications et à votre environnement informatique. Vous pouvez également utiliser cette méthode lorsque vous migrez les données de configuration des anciens systèmes vers AWS AppConfig.

La création d'un profil de configuration pour les sources de données non prises en charge de manière native AWS AppConfig implique les processus ou actions suivants :

1. Créez une AWS Lambda fonction qui extrait les données de votre source de données. Tant qu'une fonction Lambda peut accéder à la source de données, votre AWS AppConfig extension pourra récupérer les données.
2. Créez une AWS AppConfig extension personnalisée qui invoque votre fonction Lambda. Pour de plus amples informations, veuillez consulter Procédure pas à pas : création d'extensions personnalisées AWS AppConfig.
3. Créez un profil de configuration AWS AppConfig de forme libre. Plus précisément, créez un profil de configuration qui utilise la définition de configuration AWS AppConfig hébergée. Le profil de configuration fonctionne comme un magasin de données temporaire une fois que votre fonction

Lambda a récupéré votre configuration à partir de votre source. Votre application récupérera les données de configuration depuis le magasin de configuration AWS AppConfig hébergé. Pour de plus amples informations, veuillez consulter Création d'un profil de configuration sous forme libre dans AWS AppConfig.

4. Créez une association d'extension qui se déclenche à l'aide du point `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` d'action. Pour de plus amples informations, veuillez consulter Étape 4 : Création d'une association d'extension pour une AWS AppConfig extension personnalisée.

Une fois configurée, lorsque votre application demande une nouvelle version des données de configuration, le Lambda récupère vos données de configuration et les extrait dans le profil de configuration. AWS AppConfig enregistre ensuite le profil de configuration et vos données tierces.

Lorsque vous êtes prêt, vous pouvez déployer le profil de configuration dans vos applications, comme pour tout autre type de données de configuration.

> **Note**
>
> Vous pouvez choisir d'insérer des données tierces conformément aux données de configuration existantes ou de faire en sorte que l'intégralité du contenu des données de configuration contienne uniquement les données tierces. Si vous souhaitez que les données soient conformes à d'autres données existantes, cette logique doit faire partie de la fonction Lambda qui importe les données depuis la source tierce.

Migration vers des AWS AppConfig services de configuration existants et locaux

Si vous avez commencé à utiliser des données de configuration ou des indicateurs de fonctionnalité existants dans un autre système AWS AppConfig et que vous disposez toujours de ces données, vous pouvez utiliser le processus décrit plus haut dans cette rubrique pour effectuer une migration depuis votre ancien système vers AWS AppConfig. Vous pouvez créer une extension qui extrait les données de votre ancien système et les déploie. AWS AppConfig Cette utilisation AWS AppConfig vous permet de bénéficier de tous les contrôles de sécurité et de tous les avantages tout en continuant à utiliser vos anciens magasins de données.

Déploiement d'indicateurs de fonctionnalités et de données de configuration dans AWS AppConfig

Après avoir créé les artefacts nécessaires pour utiliser les indicateurs de fonctionnalités et les données de configuration en format libre, vous pouvez créer un nouveau déploiement. Lorsque vous créez un nouveau déploiement, vous spécifiez les informations suivantes :

- Un identifiant d'application
- Un ID de profil de configuration
- Une version de configuration
- Un ID d'environnement dans lequel vous souhaitez déployer les données de configuration
- Un identifiant de stratégie de déploiement qui définit la rapidité avec laquelle vous souhaitez que les modifications prennent effet
- Un identifiant de clé AWS Key Management Service (AWS KMS) pour chiffrer les données à l'aide d'une clé gérée par le client.

Lorsque vous appelez l'action StartDeploymentAPI, AWS AppConfig exécute les tâches suivantes :

1. Récupère les données de configuration du magasin de données sous-jacent à l'aide de l'URI de localisation dans le profil de configuration.
2. Vérifie que les données de configuration sont syntaxiquement et sémantiquement correctes en utilisant les validateurs que vous avez spécifiés lors de la création de votre profil de configuration.
3. Met en cache une copie des données afin qu'elles soient prêtes à être récupérées par votre application. Cette copie mise en cache s'appelle les données déployées.

Vous pouvez atténuer les situations dans lesquelles le déploiement des données de configuration entraîne des erreurs dans votre application en combinant des stratégies de AWS AppConfig déploiement et des annulations automatiques basées sur les CloudWatch alarmes Amazon. Une stratégie de déploiement vous permet d'apporter lentement les modifications aux environnements de production en quelques minutes ou heures. Une fois configurée, si une ou plusieurs CloudWatch alarmes passent à l'état d'alarme pendant un déploiement, vos données de configuration sont AWS AppConfig automatiquement rétablies à la version précédente. Pour plus d'informations sur les stratégies de déploiement, consultezTravailler avec des stratégies de déploiement. Pour plus

d'informations sur les annulations automatiques, consultezSurveillance des déploiements pour une annulation automatique.

Rubriques

- Travailler avec des stratégies de déploiement
- Déploiement d'une configuration
- Déploiement de AWS AppConfig configurations en utilisant CodePipeline
- Rétablir une configuration

Travailler avec des stratégies de déploiement

Une stratégie de déploiement vous permet d'apporter lentement les modifications aux environnements de production en quelques minutes ou heures. Une stratégie de AWS AppConfig déploiement définit les aspects importants suivants d'un déploiement de configuration.

Paramètre	Description
Type de déploiement	Le type de déploiement définit le mode de déploiement ou de déploiement de la configura tion. AWS AppConfig prend en charge les types de déploiement linéaire et exponentiel. • Linéaire : pour ce type, AWS AppConfig traite le déploiement par incréments du facteur de croissance répartis uniformément sur le déploiement. Voici un exemple de calendrie r pour un déploiement de 10 heures utilisant une croissance linéaire de 20 % :

Temps écoulé	Progression du déploiement
0 heure	0 %
2 heures	20 %

Paramètre	Description

	<table><tr><th>Temps écoulé</th><th>Progression du déploiement</th></tr><tr><td>4 heures</td><td>40 %</td></tr><tr><td>6 heures</td><td>60 %</td></tr><tr><td>8 heures</td><td>80 %</td></tr><tr><td>10 heures</td><td>100 %</td></tr></table>
	• Exponentiel : pour ce type, AWS AppConfig traite le déploiement de manière exponenti elle à l'aide de la formule suivante : `G*(2^N)`. Dans cette formule, `G` est le pourcentage d'étape spécifié par l'utilisa teur et `N` est le nombre d'étapes jusqu'à ce que la configuration soit déployée sur toutes les cibles. Par exemple, si vous spécifiez un facteur de croissance de 2, le système déploie la configuration comme suit :
	`2*(2^0)`
	`2*(2^1)`
	`2*(2^2)`
	Exprimé numériquement, le déploiement se déroule comme suit : 2 % des cibles, 4 % des cibles, 8 % des cibles, et cela se poursuit jusqu'à ce que la configuration ait été déployée sur toutes les cibles.

Paramètre	Description
Pourcentage d'étape (facteur de croissance)	Ce paramètre spécifie le pourcentage de mandataires à cibler à chaque étape du déploiement. **Note** Dans le SDK et la Référence d'API AWS AppConfig, `step percentage` est appelé `growth factor`.
Temps de déploiement	Ce paramètre indique la durée pendant laquelle les AWS AppConfig déploiements sur les hôtes sont effectués. Il ne s'agit pas d'une valeur de délai d'attente. Il s'agit d'une fenêtre horaire au cours de laquelle le déploiement est géré à intervalles réguliers.
Temps d'attente	Ce paramètre indique la durée pendant laquelle Amazon AWS AppConfig surveille les CloudWatch alarmes après le déploiement de la configuration sur 100 % de ses cibles, avant de considérer que le déploiement est terminé. Si une alarme est déclenchée pendant cette période, AWS AppConfig annule le déploieme nt. Vous devez configurer les autorisations pour AWS AppConfig revenir en arrière en fonction des CloudWatch alarmes. Pour de plus amples informations, veuillez consulter (Recommandé) Configurer les autorisations pour la restauration automatique.

Vous pouvez choisir une stratégie prédéfinie incluse AWS AppConfig ou créer la vôtre.

Rubriques

Utilisation de stratégies de déploiement prédéfinies

AWS AppConfig inclut des stratégies de déploiement prédéfinies pour vous aider à déployer rapidement une configuration. Au lieu de créer vos propres stratégies, vous pouvez choisir l'une des options suivantes lorsque vous déployez une configuration.

Stratégie de déploiement	Description
AppConfig. Linéaire 20 PercentEvery à 6 minutes	AWS recommandé : Cette stratégie déploie la configuration sur 20 % de toutes les cibles toutes les six minutes pour un déploiement de 30 minutes. Le système surveille les CloudWatch alarmes Amazon pendant 30 minutes. Si aucune alarme n'est reçue pendant cette période, le déploieme nt est terminé. Si une alarme est déclenchée pendant cette période, AWS AppConfig annule le déploiement. Nous recommandons d'utiliser cette stratégie pour les déploiements de production, car elle est conforme aux AWS meilleures pratiques et met davantage l'accent sur la sécurité des déploiements en raison de sa longue durée et de son temps de cuisson.
AppConfig. Canary 10 % 20 minutes	AWS recommandé : Cette stratégie traite le déploiement de manière exponentielle en utilisant un facteur de croissance de 10 % sur 20 minutes. Le système surveille les CloudWatch alarmes pendant 10 minutes. Si aucune alarme n'est reçue pendant cette période, le déploieme

Stratégie de déploiement	Description
	nt est terminé. Si une alarme est déclenchée pendant cette période, AWS AppConfig annule le déploiement. Nous recommandons d'utiliser cette stratégie pour les déploiements de production car elle est conforme aux AWS meilleures pratiques en matière de déploiements de configuration.
AppConfig.AllAtOnce	Quick : Cette stratégie déploie immédiatement la configuration sur toutes les cibles. Le système surveille les CloudWatch alarmes pendant 10 minutes. Si aucune alarme n'est reçue pendant cette période, le déploiement est terminé. Si une alarme est déclenchée pendant cette période, AWS AppConfig annule le déploieme nt.
AppConfig. Linéaire 50 30 secondes PercentEv ery	Tests et démonstration : Cette stratégie déploie la configuration sur la moitié de toutes les cibles toutes les 30 secondes pour un déploiement d'une minute. Le système surveille les CloudWatch alarmes Amazon pendant 1 minute. Si aucune alarme n'est reçue pendant cette période, le déploieme nt est terminé. Si une alarme est déclenchée pendant cette période, AWS AppConfig annule le déploiement. Nous vous recommandons d'utiliser cette stratégie uniquement à des fins de test ou de démonstration, car elle a une courte durée et un temps d'attente.

Création d'une stratégie de déploiement

Si vous ne souhaitez pas utiliser l'une des stratégies de déploiement prédéfinies, vous pouvez créer la vôtre. Vous pouvez créer un maximum de 20 stratégies de déploiement. Lorsque vous déployez une configuration, vous pouvez choisir la stratégie de déploiement qui convient le mieux à l'application et à l'environnement.

Création d'une stratégie de AWS AppConfig déploiement (console)

Utilisez la procédure suivante pour créer une stratégie de AWS AppConfig déploiement à l'aide de la AWS Systems Manager console.

Pour créer une stratégie de déploiement

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Stratégies de déploiement, puis sélectionnez Créer une stratégie de déploiement.
3. Pour Name (Nom), entrez un nom pour la stratégie de déploiement.
4. Pour Description, entrez des informations sur la stratégie de déploiement.
5. Pour Type de déploiement, choisissez un type.
6. Pour Step percentage (Pourcentage d'étape), choisissez le pourcentage de mandataires à cibler à chaque étape du déploiement.
7. Dans la zone Deployment time (Temps de déploiement), entrez la durée totale du déploiement en minutes ou en heures.
8. Pour le temps de cuisson, entrez le temps total, en minutes ou en heures, nécessaire pour surveiller les CloudWatch alarmes Amazon avant de passer à l'étape suivante d'un déploiement ou avant de considérer le déploiement comme terminé.
9. Dans la section Tags (Balises) entrez une clé et une valeur facultative. Vous pouvez spécifier un maximum de 50 balises par ressource.
10. Choisissez Create deployment strategy (Créer une stratégie de déploiement).

> **Important**
>
> Si vous avez créé un profil de configuration pour AWS CodePipeline, vous devez créer un pipeline dans CodePipeline lequel vous AWS AppConfig spécifiez le fournisseur de

déploiement. Vous n'avez pas besoin de jouerDéploiement d'une configuration. Toutefois, vous devez configurer un client pour recevoir les mises à jour de configuration des applications, comme décrit dansRécupération des données de configuration sans AWS AppConfig agent. Pour plus d'informations sur la création d'un pipeline AWS AppConfig spécifié comme fournisseur de déploiement, voir Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement dans le Guide de AWS CodePipeline l'utilisateur.

Passez à Déploiement d'une configuration.

Création d'une stratégie de AWS AppConfig déploiement (ligne de commande)

La procédure suivante décrit comment utiliser AWS CLI (sous Linux ou Windows) ou comment Outils AWS pour PowerShell créer une stratégie de AWS AppConfig déploiement.

Pour créer une stratégie de déploiement étape par étape

1. Ouvrez le AWS CLI.
2. Exécutez la commande suivante pour créer une stratégie de déploiement.

   Linux

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

Windows

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

PowerShell

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

Le système retourne des informations telles que les suivantes.

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
```

Déploiement d'une configuration

Après avoir créé les artefacts nécessaires pour utiliser les indicateurs de fonctionnalités et les données de configuration en format libre, vous pouvez créer un nouveau déploiement à l'aide du SDK AWS Management Console AWS CLI, du ou du SDK. Le démarrage d'un déploiement dans AWS AppConfig appelle l'opération StartDeploymentAPI. Cet appel inclut IDs l' AWS AppConfig application, l'environnement, le profil de configuration et (éventuellement) la version des données de configuration à déployer. L'appel inclut également l'ID de la stratégie de déploiement à utiliser, qui détermine le mode de déploiement des données de configuration.

Si vous déployez des secrets stockés dans AWS Secrets Manager des objets Amazon Simple Storage Service (Amazon S3) chiffrés avec une clé gérée par le client ou des paramètres de chaîne sécurisés stockés AWS Systems Manager dans Parameter Store chiffrés avec une clé gérée par le client, vous devez spécifier une valeur pour `KmsKeyIdentifier` le paramètre. Si votre configuration n'est pas chiffrée ou qu'elle est chiffrée avec un Clé gérée par AWS, il n'est pas nécessaire de spécifier une valeur pour le `KmsKeyIdentifier` paramètre.

> **Note**
>
> La valeur que vous spécifiez `KmsKeyIdentifier` doit être une clé gérée par le client. Il n'est pas nécessaire que ce soit la même clé que celle que vous avez utilisée pour chiffrer votre configuration.
> Lorsque vous démarrez un déploiement avec un`KmsKeyIdentifier`, la politique d'autorisation attachée à votre principal AWS Identity and Access Management (IAM) doit autoriser l'`kms:GenerateDataKey`opération.

AWS AppConfig surveille la distribution à tous les hôtes et indique le statut. Si une distribution échoue AWS AppConfig , la configuration est annulée.

> **Note**
>
> Vous ne pouvez déployer qu'une seule configuration à la fois dans un environnement. Cependant, vous pouvez déployer une configuration dans chaque environnement en même temps.

Déployer une configuration (console)

Utilisez la procédure suivante pour déployer une AWS AppConfig configuration à l'aide de la AWS Systems Manager console.

Pour déployer une configuration à l'aide de la console

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le volet de navigation, choisissez Applications, puis choisissez une application que vous avez créée dansCréation d'un espace de noms pour votre application dans AWS AppConfig.
3. Dans l'onglet Environnements, cliquez sur le bouton radio correspondant à un environnement, puis choisissez Afficher les détails.
4. Choisissez Démarrer le déploiement.
5. Dans Configuration, choisissez une configuration dans la liste.
6. En fonction de la source de votre configuration, utilisez la liste des versions pour choisir la version que vous souhaitez déployer.
7. Pour Deployment strategy (Stratégie de déploiement), choisissez une stratégie dans la liste.
8. (Facultatif) Pour Description du déploiement, saisissez une description.
9. Pour Options de chiffrement supplémentaires, choisissez une AWS Key Management Service clé dans la liste.
10. (Facultatif) Dans la section Balises, choisissez Ajouter une nouvelle balise et entrez une clé et une valeur facultative. Vous pouvez spécifier un maximum de 50 balises par ressource.
11. Choisissez Démarrer le déploiement.

Déployer une configuration (ligne de commande)

La procédure suivante décrit comment utiliser AWS CLI (sous Linux ou Windows) ou comment Outils AWS pour PowerShell déployer une AWS AppConfig configuration.

Pour déployer une configuration étape par étape

1. Ouvrez le AWS CLI.
2. Exécutez la commande suivante pour déployer une configuration.

   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
```

Le système retourne des informations telles que les suivantes.

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
```

Déploiement de AWS AppConfig configurations en utilisant CodePipeline

AWS AppConfig est une action de déploiement intégrée pour AWS CodePipeline (CodePipeline). CodePipeline est un service de livraison continue entièrement géré qui vous aide à automatiser vos pipelines de publication pour des mises à jour rapides et fiables des applications et de l'infrastructure. CodePipeline automatise les phases de création, de test et de déploiement de votre processus de publication chaque fois qu'un changement de code est effectué, en fonction du modèle de version que vous définissez. Pour plus d'informations, consultez Qu'est-ce qu' AWS CodePipeline ?

L'intégration de AWS AppConfig avec CodePipeline offre les avantages suivants :

- Les clients qui géraient l' CodePipeline orchestration disposent désormais d'un moyen léger de déployer des modifications de configuration dans leurs applications sans avoir à déployer l'intégralité de leur base de code.
- Les clients qui souhaitent gérer des déploiements AWS AppConfig de configuration mais qui sont limités parce que le code ou AWS AppConfig le magasin de configuration ne sont pas compatibles avec leur code actuel ou leur magasin de configuration disposent désormais d'options supplémentaires. CodePipeline prend en charge AWS CodeCommit GitHub, et BitBucket (pour n'en nommer que quelques-uns).

> **Note**
>
> AWS AppConfig l'intégration avec n' CodePipeline est prise en charge que Régions AWS là où CodePipeline elle est disponible.

Comment fonctionne l'intégration

Vous commencez par le configurer et le configurer CodePipeline. Cela inclut l'ajout de votre configuration à un magasin de code CodePipeline pris en charge. Vous devez ensuite configurer votre AWS AppConfig environnement en effectuant les tâches suivantes :

- Création d'un espace de noms et d'un profil de configuration
- Choisissez une stratégie de déploiement prédéfinie ou créez la vôtre

Une fois ces tâches terminées, vous créez un pipeline dans CodePipeline lequel vous spécifiez AWS AppConfig le fournisseur de déploiement. Vous pouvez ensuite modifier votre configuration et la télécharger dans votre magasin de CodePipeline codes. Le téléchargement de la nouvelle configuration lance automatiquement un nouveau déploiement dans CodePipeline. Une fois le déploiement terminé, vous pouvez vérifier vos modifications. Pour plus d'informations sur la création d'un pipeline AWS AppConfig spécifié comme fournisseur de déploiement, voir Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement dans le Guide de AWS CodePipeline l'utilisateur.

Rétablir une configuration

Au cours d'un déploiement, vous pouvez atténuer les situations dans lesquelles des données de configuration mal formées ou incorrectes provoquent des erreurs dans votre application en utilisant des annulations automatiques (si une alarme se déclenche pendant un déploiement) ou en rétablissant les données de configuration à la version précédente (si le déploiement s'est terminé avec succès).

Pour les annulations automatiques, vous pouvez utiliser une combinaison de stratégies de AWS AppConfig déploiement et d' CloudWatch alarmes Amazon. Une fois configurée, si une ou plusieurs CloudWatch alarmes se déclenchent `ALARM` pendant un déploiement, vos données de configuration AWS AppConfig sont automatiquement rétablies à la version précédente, évitant ainsi les pannes ou les erreurs des applications. Consultez (Recommandé) Configurer les autorisations pour la restauration automatique pour démarrer.

> **Note**
>
> Vous pouvez également annuler une configuration en appelant l'opération StopDeploymentAPI alors qu'un déploiement est toujours en cours.

Pour les déploiements réussis, il est AWS AppConfig également possible de rétablir les données de configuration à une version précédente en utilisant le `AllowRevert` paramètre avec l'opération d'StopDeploymentAPI. Pour certains clients, le retour à une configuration précédente après un déploiement réussi garantit que les données seront les mêmes qu'avant le déploiement. Le retour en arrière ignore également les moniteurs d'alarme, ce qui peut empêcher la progression d'une application en cas d'urgence.

> **Important**
>
> Si vous appelez `StopDeployment` avec le `AllowRevert` paramètre activé, le déploiement n' AWS AppConfig annulera le déploiement que s'il a réussi au cours des dernières 72 heures. Après 72 heures, le déploiement ne peut plus être annulé. Vous devez créer un nouveau déploiement.

Voici un aperçu des `StopDeployment` fonctionnalités en fonction des différentes situations.

1. S'il `StopDeployment` est appelé dans le cadre d'un déploiement en cours, l'état de déploiement résultant sera`ROLLED_BACK`.
2. Si `StopDeployment` (with`AllowRevert`) est appelé lors d'un déploiement en cours, l'état de déploiement résultant sera`ROLLED_BACK`.
3. S'`StopDeployment`il est appelé après un déploiement terminé, un `BadRequestException` sera lancé.
4. Si `StopDeployment` (with`AllowRevert`) est appelé après un déploiement terminé, l'état de déploiement résultant sera`REVERTED`.
5. Si `StopDeployment` (with`AllowRevert`) est appelé après un déploiement terminé après 72 heures, un `BadRequestException` sera lancé.

Vous pouvez utiliser le AWS CLI pour appeler l'StopDeploymentopération avec le `AllowRevert` paramètre. Voici un exemple de AWS CLI commande qui inclut le `AllowRevert` paramètre.

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

Récupération des indicateurs de fonctionnalités et des données de configuration dans AWS AppConfig

Votre application récupère les indicateurs de fonctionnalités et les données de configuration sous forme libre en établissant une session de configuration à l'aide du service AWS AppConfig Data. Nous vous recommandons d'utiliser l' AWS AppConfig agent pour récupérer les données de configuration. L'agent (ou l'extension AWS AppConfig Agent Lambda pour les environnements de calcul Lambda) gère une série d'appels d'API et de jetons de session en votre nom. À partir d'un niveau élevé, le processus fonctionne comme suit :

1. Vous configurez AWS AppConfig l'agent en tant qu'hôte local et demandez à l'agent de AWS AppConfig demander des mises à jour de configuration.
2. L'agent appelle les actions StartConfigurationSessionet GetLatestConfigurationAPI et met en cache vos données de configuration localement.
3. Pour récupérer les données, votre application lance un appel HTTP au serveur localhost. AWS AppConfig L'agent prend en charge plusieurs cas d'utilisation, comme décrit dansComment utiliser l' AWS AppConfig agent pour récupérer les données de configuration.

Si vous préférez, vous pouvez appeler manuellement ces actions d'API pour récupérer une configuration. Le processus de l'API fonctionne comme suit :

1. Votre application établit une session de configuration à l'aide de l'action `StartConfigurationSession` API. Le client de votre session passe ensuite des appels périodiques `GetLatestConfiguration` pour vérifier et récupérer les dernières données disponibles.
2. Lorsque vous appelez`StartConfigurationSession`, votre code envoie les identifiants (ID ou nom) d'une AWS AppConfig application, d'un environnement et d'un profil de configuration que la session suit.
3. En réponse, AWS AppConfig fournit un `InitialConfigurationToken` à donner au client de la session et à utiliser la première fois qu'il appelle `GetLatestConfiguration` cette session.
4. Lorsque vous appelez`GetLatestConfiguration`, votre code client envoie la `ConfigurationToken` valeur la plus récente qu'il possède et reçoit en réponse :
   - `NextPollConfigurationToken`: `ConfigurationToken` valeur à utiliser lors du prochain appel à`GetLatestConfiguration`.

- La configuration : les dernières données destinées à la session. Ce champ peut être vide si le client dispose déjà de la dernière version de la configuration.

> **Note**
>
> La récupération des données de configuration depuis un autre Compte AWS n'est pas prise en charge.

Table des matières

Qu'est-ce que AWS AppConfig l'agent ?

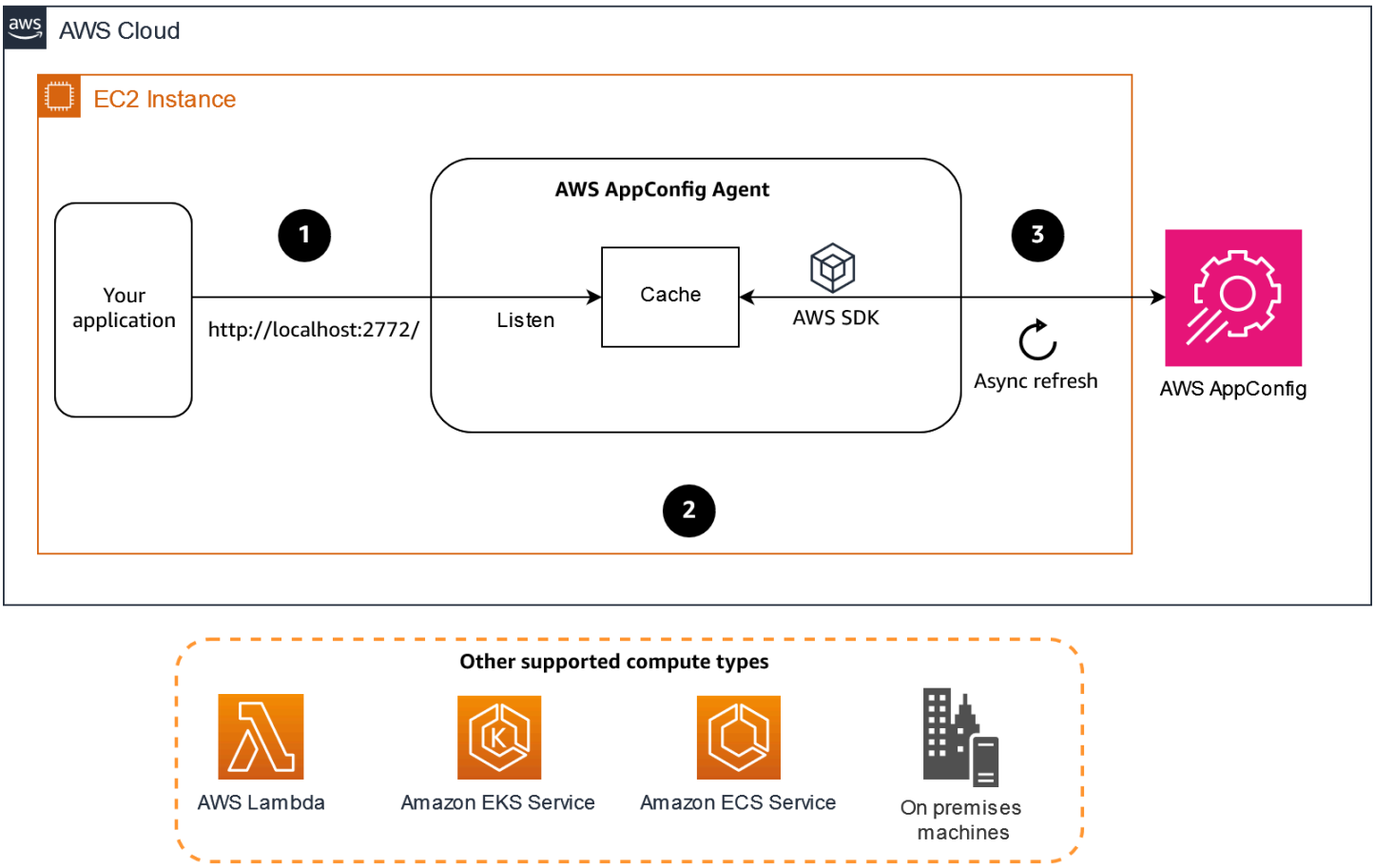
AWS AppConfig L'agent est un processus développé et géré par Amazon pour récupérer les données de configuration à partir de. AWS AppConfig Avec l'agent, vous pouvez mettre en cache les données de configuration localement et interroger le service de plan de AWS AppConfig données de manière asynchrone pour obtenir des mises à jour. Ce caching/polling processus garantit que vos données de configuration sont toujours disponibles pour votre application tout en minimisant le temps de latence et les coûts. L'agent n'est pas le seul moyen de récupérer les données de configuration AWS AppConfig, mais c'est le moyen recommandé. L'agent améliore le traitement et la gestion des demandes de la manière suivante :

- L'agent appelle AWS AppConfig en votre nom en utilisant un principal AWS Identity and Access Management (IAM) et en gérant un cache local de données de configuration. En récupérant les données de configuration depuis le cache local, votre application nécessite moins de mises à jour de code pour gérer les données de configuration, récupère les données de configuration en quelques millisecondes et n'est pas affectée par les problèmes réseau susceptibles de perturber les appels pour ces données.
- L'agent propose une expérience native pour récupérer et résoudre les indicateurs de AWS AppConfig fonctionnalités.

- Prêt à l'emploi, l'agent fournit les meilleures pratiques en matière de stratégies de mise en cache, d'intervalles d'interrogation et de disponibilité des données de configuration locales, tout en suivant les jetons de configuration nécessaires pour les appels de service suivants.
- Lorsqu'il s'exécute en arrière-plan, l'agent interroge régulièrement le service de plan de AWS AppConfig données pour obtenir des mises à jour des données de configuration. Votre application peut récupérer les données en se connectant à localhost sur le port 2772 (une valeur de port par défaut personnalisable) et en appelant HTTP GET pour récupérer les données.

> **Note**
>
> AWS AppConfig L'agent met en cache les données la première fois que le service récupère vos données de configuration. Pour cette raison, le premier appel pour récupérer les données est plus lent que les appels suivants.

Le schéma suivant montre le fonctionnement de AWS AppConfig l'agent.



1. Votre application demande des données de configuration à l'agent.
2. L'agent renvoie les données d'un cache en mémoire.
3. L'agent interroge le AWS AppConfig service de manière asynchrone pour obtenir les dernières données de configuration selon une cadence prédéfinie. Les dernières données de configuration sont toujours stockées dans un cache en mémoire.

Comment utiliser l' AWS AppConfig agent pour récupérer les données de configuration

L' AWS AppConfig agent est la méthode recommandée pour récupérer les indicateurs de AWS AppConfig fonctionnalités ou les données de configuration sous forme libre. L'agent est pris en charge sur toutes les formes de AWS calcul, notamment Amazon EC2, Amazon ECS, Amazon EKS et Lambda. Une fois que vous avez terminé la configuration initiale de l'agent, il est plus simple d'utiliser l'agent pour récupérer les données de configuration que d'appeler directement AWS

AppConfig APIs. L'agent met automatiquement en œuvre les meilleures pratiques et peut réduire vos coûts d'utilisation AWS AppConfig en réduisant le nombre d'appels d'API pour récupérer les configurations.

> **Note**
>
> La récupération des données de configuration depuis un autre Compte AWS n'est pas prise en charge.

Rubriques

Utilisation de AWS AppConfig l'agent avec AWS Lambda

Une AWS Lambda extension est un processus complémentaire qui augmente les capacités d'une fonction Lambda. Une extension peut démarrer avant qu'une fonction ne soit invoquée, s'exécuter en parallèle avec une fonction et continuer à s'exécuter après le traitement de l'appel de fonction. En substance, une extension Lambda est comme un client qui s'exécute en parallèle à un appel Lambda. Ce client parallèle peut s'interfacer avec votre fonction à tout moment au cours de son cycle de vie.

Si vous utilisez des indicateurs de AWS AppConfig fonctionnalité ou d'autres données de configuration dynamiques dans une fonction Lambda, nous vous recommandons d'ajouter l'extension Agent AWS AppConfig Lambda en tant que couche à votre fonction Lambda. Cela simplifie les indicateurs de fonctionnalité d'appel, et l'extension elle-même inclut les meilleures pratiques qui simplifient l'utilisation AWS AppConfig tout en réduisant les coûts. La réduction des coûts résulte de

la diminution du nombre d'appels d'API au AWS AppConfig service et de la réduction des temps de traitement des fonctions Lambda. Pour plus d'informations sur les extensions Lambda, consultez la section Extensions Lambda dans le Guide du développeur.AWS Lambda

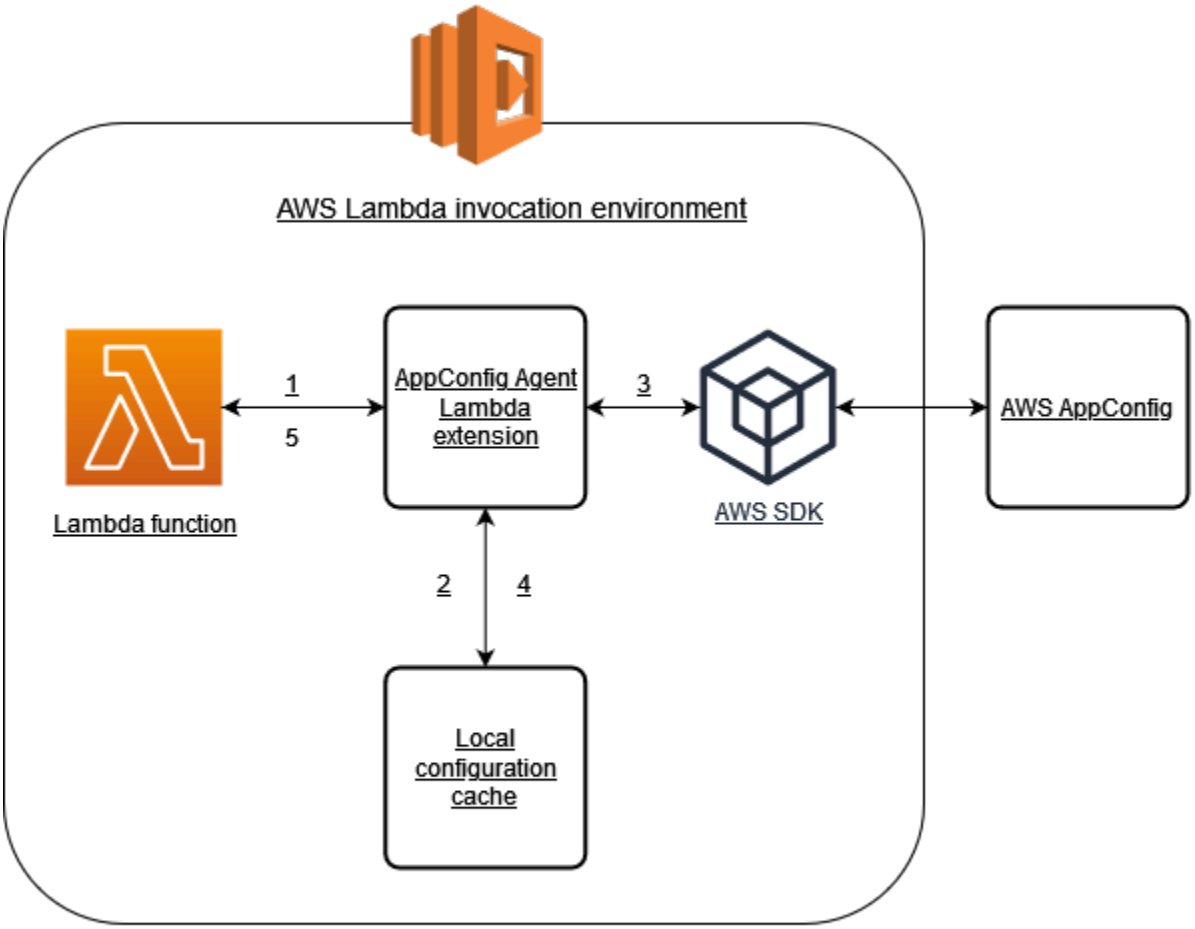
> **Note**
>
> AWS AppConfig est une capacité de AWS Systems Manager. AWS AppConfig la tarification est basée sur le nombre de fois qu'une configuration est appelée et reçue. Vos coûts augmentent si votre Lambda effectue plusieurs démarrages à froid et récupère fréquemment de nouvelles données de configuration.

Rubriques

Comprendre le fonctionnement de l' AWS AppConfig extension Agent Lambda

Si vous gérez les configurations d'une fonction Lambda sans extensions Lambda, vous devez configurer votre fonction Lambda pour recevoir des mises à jour de configuration en intégrant les actions et API. AWS AppConfig StartConfigurationSessionGetLatestConfiguration

L'intégration de l'extension AWS AppConfig Agent Lambda à votre fonction Lambda simplifie ce processus. L'extension se charge d'appeler le AWS AppConfig service, de gérer un cache local de données récupérées, de suivre les jetons de configuration nécessaires pour les prochains appels de service et de vérifier périodiquement les mises à jour de configuration en arrière-plan. Le schéma suivant montre comment cela fonctionne.



1. Vous configurez l'extension AWS AppConfig Agent Lambda en tant que couche de votre fonction Lambda.
2. Pour accéder à ses données de configuration, votre fonction appelle l' AWS AppConfig extension sur un point de terminaison HTTP exécuté sur`localhost:2772`.
3. L'extension gère un cache local des données de configuration. Si les données ne se trouvent pas dans le cache, l'extension appelle AWS AppConfig pour obtenir les données de configuration.
4. Dès réception de la configuration par le service, l'extension la stocke dans le cache local et la transmet à la fonction Lambda.
5. AWS AppConfig L'extension Agent Lambda vérifie régulièrement les mises à jour de vos données de configuration en arrière-plan. Chaque fois que votre fonction Lambda est invoquée, l'extension vérifie le temps écoulé depuis qu'elle a récupéré une configuration. Si le temps écoulé est supérieur à l'intervalle d'interrogation configuré, l'extension appelle AWS AppConfig pour vérifier

les données récemment déployées, met à jour le cache local en cas de modification et réinitialise le temps écoulé.

> **Note**
>
> - Lambda instancie des instances distinctes correspondant au niveau de simultanéité requis par votre fonction. Chaque instance est isolée et conserve son propre cache local de vos données de configuration. Pour plus d'informations sur les instances Lambda et la simultanéité, consultez la section Gestion de la simultanéité pour une fonction Lambda.
> - Le temps nécessaire pour qu'une modification de configuration apparaisse dans une fonction Lambda, après le déploiement d'une configuration mise à jour depuis AWS AppConfig, dépend de la stratégie de déploiement que vous avez utilisée pour le déploiement et de l'intervalle d'interrogation que vous avez configuré pour l'extension.

Ajout de l'extension AWS AppConfig Agent Lambda

Pour utiliser l'extension AWS AppConfig Agent Lambda, vous devez l'ajouter à votre Lambda. Cela peut être fait en ajoutant l'extension AWS AppConfig Agent Lambda à votre fonction Lambda sous forme de couche ou en activant l'extension sur une fonction Lambda en tant qu'image de conteneur.

> **Note**
>
> L' AWS AppConfig extension est indépendante de l'environnement d'exécution et prend en charge tous les environnements d'exécution.

Avant de commencer

Avant d'activer l'extension AWS AppConfig Agent Lambda, procédez comme suit :

- Organisez les configurations dans votre fonction Lambda afin de pouvoir les externaliser dans. AWS AppConfig
- Créez des AWS AppConfig artefacts et des données de configuration, notamment des indicateurs de fonctionnalités ou des données de configuration sous forme libre. Pour de plus amples informations, veuillez consulter Création d'indicateurs de fonctionnalités et de données de configuration sous forme libre dans AWS AppConfig.

- Ajoutez `appconfig:StartConfigurationSession` et `appconfig:GetLatestConfiguration` à la politique AWS Identity and Access Management (IAM) utilisée par le rôle d'exécution de la fonction Lambda. Pour plus d'informations, veuillez consulter Rôle d'exécution AWS Lambda dans le Guide du développeur AWS Lambda . Pour plus d'informations sur AWS AppConfig les autorisations, consultez la section Actions, ressources et clés de condition AWS AppConfig dans la référence d'autorisation de service.

Ajouter l'extension AWS AppConfig Agent Lambda à l'aide d'une couche et d'un ARN

Pour utiliser l'extension AWS AppConfig Agent Lambda, vous devez l'ajouter à votre fonction Lambda sous forme de couche. Pour plus d'informations sur la façon d'ajouter une couche à votre fonction, consultez la section Configuration des extensions dans le Guide du AWS Lambda développeur. Le nom de l'extension dans la AWS Lambda console est AWS- AppConfig -Extension. Notez également que lorsque vous ajoutez l'extension en tant que couche à votre Lambda, vous devez spécifier un Amazon Resource Name (ARN). Dans l'une des listes suivantes, choisissez un ARN correspondant à la plate-forme et à l' Région AWS endroit où vous avez créé le Lambda.

- plate-forme x86-64
- ARM64plateforme

Si vous souhaitez tester l'extension avant de l'ajouter à votre fonction, vous pouvez vérifier qu'elle fonctionne à l'aide de l'exemple de code suivant.

```
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
```

Pour le tester, créez une nouvelle fonction Lambda pour Python, ajoutez l'extension, puis exécutez la fonction Lambda. Après avoir exécuté la fonction Lambda, celle-ci renvoie la configuration que vous avez spécifiée pour le chemin http://localhost:2772. AWS AppConfig Pour plus d'informations sur la création d'une fonction Lambda, voir Création d'une fonction Lambda avec la console dans le Guide du développeur.AWS Lambda

Important

Vous pouvez consulter les données des journaux de l'extension AWS AppConfig Agent Lambda dans les AWS Lambda journaux. Les entrées du journal sont précédées `appconfig agent` de. Voici un exemple :

```
[appconfig agent] 2024/05/07 04:19:01 ERROR retrieve failure for
 'SourceEventConfig:SourceEventConfigEnvironment:SourceEventConfigProfile':
 StartConfigurationSession: api error AccessDenied: User:
 arn:aws:sts::0123456789:assumed-role/us-east-1-LambdaRole/
extension1 is not authorized to perform: sts:AssumeRole on resource:
 arn:aws:iam::0123456789:role/test1 (retry in 60s)
```

Configuration de l'extension AWS AppConfig Agent Lambda

Vous pouvez configurer l'extension en modifiant les variables d' AWS Lambda environnement suivantes. Pour plus d'informations, consultez la section Utilisation de variables d' AWS Lambda environnement dans le Guide du AWS Lambda développeur.

Préextraction des données de configuration

La variable d'environnement `AWS_APPCONFIG_EXTENSION_PREFETCH_LIST` peut améliorer le temps de démarrage de votre fonction. Lorsque l'extension AWS AppConfig Agent Lambda est initialisée, elle récupère la configuration spécifiée avant que AWS AppConfig Lambda ne commence à initialiser votre fonction et à appeler votre gestionnaire. Dans certains cas, les données de configuration sont déjà disponibles dans le cache local avant que votre fonction ne les demande.

Pour utiliser la fonctionnalité de prélecture, définissez la valeur de la variable d'environnement sur le chemin correspondant à vos données de configuration. Par exemple, si votre configuration correspond à une application, à un environnement et à un profil de configuration nommés respectivement « my_application », « my_environment » et « my_configuration_data », le chemin serait. `/applications/my_application/environments/my_environment/configurations/my_configuration_data` Vous pouvez spécifier plusieurs éléments de configuration en les répertoriant sous forme de liste séparée par des virgules (si le nom d'une ressource inclut une virgule, utilisez l'ID de la ressource au lieu de son nom).

Accès aux données de configuration depuis un autre compte

L'extension AWS AppConfig Agent Lambda peut récupérer les données de configuration d'un autre compte en spécifiant un rôle IAM qui accorde des autorisations sur les données. Pour configurer cela, procédez comme suit :

1. Dans le compte utilisé AWS AppConfig pour gérer les données de configuration, créez un rôle doté d'une politique de confiance qui accorde au compte exécutant la fonction Lambda l'accès aux `appconfig:GetLatestConfiguration` actions `appconfig:StartConfigurationSession` et, ainsi qu'aux actions partielles ou complètes ARNs correspondant aux ressources de AWS AppConfig configuration.
2. Dans le compte exécutant la fonction Lambda, ajoutez la variable d'`AWS_APPCONFIG_EXTENSION_ROLE_ARN`environnement à la fonction Lambda avec l'ARN du rôle créé à l'étape 1.
3. (Facultatif) Si nécessaire, un identifiant externe peut être spécifié à l'aide de la variable d'`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`environnement. De même, un nom de session peut être configuré à l'aide de la variable d'`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`environnement.

> **Note**
>
> Notez les informations suivantes.
>
> - L'extension AWS AppConfig Agent Lambda ne peut récupérer les données que d'un seul compte. Si vous spécifiez un rôle IAM, l'extension ne sera pas en mesure de récupérer les données de configuration du compte sur lequel la fonction Lambda est exécutée.
> - AWS Lambda enregistre les informations relatives à l'extension AWS AppConfig Agent Lambda et à la fonction Lambda à l'aide d'Amazon Logs. CloudWatch
> - Le tableau suivant inclut une colonne de valeurs d'exemple. Selon la résolution de votre écran, vous devrez peut-être faire défiler le tableau vers le bas, puis vers la droite pour afficher la colonne.

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTE`	Cette variable d'environnement	2772	2772

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
NSION_HTTP_PORT	indique le port sur lequel s'exécute le serveur HTTP local hébergeant l'extension.		
AWS_APPCONFIG_EXTENSION_LOG_LEVEL	Cette variable d'environnement indique le niveau de détail enregistré par l'agent. Chaque niveau inclut le niveau actuel et tous les niveaux supérieurs. La valeur ne distingue pas les majuscules et minuscules. Du plus détaillé au moins détaillé, les niveaux de journalisation sont les suivants : trace debug infowarn,error,,fatetnone. Le trace journal contient des informations détaillées, y compris des informations temporelles, sur l'agent.	info	tracer debug info prévenir error fatal none

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS`	Cette variable d'environnement configure le nombre maximal de connexions utilisées par l'extension pour récupérer les configurations. AWS AppConfig	3	3

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS`	Cette variable d'environnement contrôle la fréquence à laquelle l'agent interroge les données AWS AppConfig de configuration mises à jour. Vous pouvez spécifier un nombre de secondes pour l'intervalle. Vous pouvez également spécifier un nombre avec une unité de temps : s pour les secondes, m pour les minutes et h pour les heures. Si aucune unité n'est spécifiée, l'agent utilise par défaut les secondes. Par exemple, 60, 60 s et 1 m donnent le même intervalle d'interrogation.	45	45 45 s 5 min 1 h

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS`	Cette variable d'environnement contrôle la durée maximale, en millisecondes, pendant laquelle l'extension attend une réponse AWS AppConfig lors de l'actualisation des données dans le cache. Si AWS AppConfig elle ne répond pas dans le délai spécifié, l'extension ignore cet intervalle d'interrogation et renvoie les données mises en cache précédemment mises à jour.	3000 ms	3000 300 ms 5s

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_PREFETCH_LIST`	Cette variable d'environnement spécifie les données de configuration que l'agent demande AWS AppConfig dès son démarrage. Plusieurs identifiants de configuration peuvent être fournis dans une liste séparée par des virgules. La préextraction des données de configuration AWS AppConfig peut réduire considérablement le temps de démarrage à froid de votre fonction.	Aucun	MyApp:MyEnv:MyConfig abcd123 : efgh456 : ijkl789 MyApp::Config1, ::Config2 MyEnv MyApp MyEnv

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
AWS_APPCONFIG_EXTENSION_PROXY_HEADERS	Cette variable d'environnement spécifie les en-têtes requis par le proxy référencé dans la variable d'AWS_APPCONFIG_EXTENSION_PROXY_URL environnement. La valeur est une liste d'en-têtes séparés par des virgules.	Aucun	en-tête : valeur h1 : v1, h2 : v2
AWS_APPCONFIG_EXTENSION_PROXY_URL	Cette variable d'environnement indique l'URL du proxy à utiliser pour les connexions entre l' AWS AppConfig extension et Services AWS. HTTPSet HTTP URLs sont pris en charge.	Aucun	http://localhost:7474 https://my-proxy.example.com
AWS_APPCONFIG_EXTENSION_ROLE_ARN	Cette variable d'environnement spécifie l'ARN du rôle IAM correspondant à un rôle qui doit être assumé par l' AWS AppConfig extension pour récupérer la configuration.	Aucun	arn:aws:iam : :123456789012:role/ MyRole

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`	Cette variable d'environnement spécifie l'identifiant externe à utiliser conjointement avec l'ARN du rôle assumé.	Aucun	MyExternalId
`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`	Cette variable d'environnement spécifie le nom de session à associer aux informations d'identification pour le rôle IAM assumé.	Aucun	AWSAppConfigAgentSession
`AWS_APPCONFIG_EXTENSION_SERVICE_REGION`	Cette variable d'environnement indique une région alternative que l'extension doit utiliser pour appeler le AWS AppConfig service. Lorsqu'elle n'est pas définie, l'extension utilise le point de terminaison de la région actuelle.	Aucun	us-east-1 eu-west-1

Variable d'environnement	Détails	Valeur par défaut	Exemples de valeurs
`AWS_APPCONFIG_EXTENSION_MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour tirer parti de fonctionnalités supplémentaires par configuration, telles que la récupération de plusieurs comptes et l'enregistrement de la configuration sur disque. Pour de plus amples informations sur l'utilisation de ces modèles, consultez Utilisation d'un manifeste pour activer des fonctionnalités de récupération supplémentaires.	Aucun	Lorsque vous utilisez AWS AppConfig la configuration comme manifeste :`MyApp:MyEnv:MyManifestConfig` . Lors du chargement du manifeste depuis le disque : `file:/path/to/manifest.json`
`AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour qu'il attende que le manifeste soit traité avant de terminer le démarrage.	true	vrai false

Présentation des versions disponibles de l'extension AWS AppConfig Agent Lambda

Cette rubrique contient des informations sur les versions de AWS AppConfig l'extension Agent Lambda. L'extension AWS AppConfig Agent Lambda prend en charge les fonctions Lambda développées pour les plateformes x86-64 et (Graviton2). ARM64 Pour fonctionner correctement, votre fonction Lambda doit être configurée pour utiliser le nom de ressource Amazon (ARN) spécifique à l' Région AWS endroit où elle est actuellement hébergée. Vous pouvez consulter Région AWS les détails de l'ARN plus loin dans cette section.

Important

Notez les informations importantes suivantes concernant l'extension AWS AppConfig Agent Lambda.

- L'action d'`GetConfiguration`API est devenue obsolète le 28 janvier 2022. Les appels destinés à recevoir des données de configuration doivent utiliser le `StartConfigurationSession` et à la `GetLatestConfiguration` APIs place. Si vous utilisez une version de l'extension AWS AppConfig Agent Lambda créée avant le 28 janvier 2022, vous devrez peut-être configurer l'autorisation d'accès à la nouvelle. APIs Pour de plus amples informations, veuillez consulter Récupération des données de configuration sans AWS AppConfig agent.
- AWS AppConfig prend en charge toutes les versions répertoriées dansAnciennes versions d'extension. Nous vous recommandons de passer régulièrement à la dernière version afin de tirer parti des améliorations apportées aux extensions.

Rubriques

AWS AppConfig Notes de mise à jour de l'extension Agent Lambda

Le tableau suivant décrit les modifications apportées aux versions récentes de l'extension AWS AppConfig Lambda.

Version	Date de lancement	Remarques
2,0,2037	12/05/2025	Ajout d'un `/ping` chemin, qui expose un simple bilan de santé renvoyant la version de cet agent. Inclut également des améliorations mineures et des corrections de bogues.
2,0.1079	12/12/2024	Améliorations mineures et corrections de bugs.
2,0.719	08/08/2024	Améliorations mineures et corrections de bugs.
2,0,678	23/07/2024	Améliorations visant à prendre en charge les cibles, les variantes et les divisions des indicateurs de fonctionn alité. Pour de plus amples informations, veuillez consulter Création d'indicateurs de fonctionnalités à variantes multiples.
2,0,501	01/07/2024	Améliorations mineures et corrections de bugs.
2,0,358	01/12/2023	Ajout de la prise en charge des fonctionnalités de récupération suivantes : • Récupération multi-com ptes : utilisez l' AWS AppConfig agent d'un compte principal ou d'une extraction Compte AWS pour récupérer les données

Version	Date de lancement	Remarques
		de configuration de plusieurs comptes fournisseurs. • Écrire une copie de configuration sur le disque : utilisez l' AWS AppConfig agent pour écrire les données de configuration sur le disque. Cette fonctionnalité permet aux clients utilisant des applications qui lisent les données de configuration sur disque de s'y intégrer AWS AppConfig.
2,0,181	14/08/2023	Ajout du support pour l' Région AWS il-central-1 d'Israël (Tel Aviv).

Version	Date de lancement	Remarques
2,0,165	21/02/2023	Correctifs de bogues mineurs. Ne limitez plus l'utilisation des extensions à des versions d'exécution spécifiques via la AWS Lambda console. Ajout de la prise en charge des éléments suivants Régions AWS : • Moyen-Orient (Émirats arabes unis), me-central-1 • Asie-Pacifique (Hyderabad), ap-south-2 • Asie-Pacifique (Melbourne), ap-southeast-4 • Europe (Espagne), eu-south-2 • Europe (Zurich), eu-central-2

Version	Date de lancement	Remarques
2,0,122	23/08/2022	Ajout de la prise en charge d'un proxy de tunneling, qui peut être configuré avec les variables d'`AWS_APPCONFIG_EXTENSION_PROXY_HEADERS` environnement `AWS_APPCONFIG_EXTENSION_PROXY_URL` et. .NET 6 a été ajouté en tant qu'environnement d'exécution. Pour plus d'informations sur les variables d'environnement, consultezConfiguration de l'extension AWS AppConfig Agent Lambda.
2,0,58	03/05/2022	Support amélioré pour les processeurs Graviton2 (ARM64) dans Lambda.

Version	Date de lancement	Remarques
2,0,45	15/03/2022	Ajout de la prise en charge de l'appel d'un indicateu r de fonctionnalité unique. Auparavant, les clients appelaient les indicateurs de fonctionnalité regroupés dans un profil de configuration et devaient analyser la réponse côté client. Avec cette version, les clients peuvent utiliser un `flag=<flag-name>` paramètre lorsqu'ils appellent le point de terminaison HTTP localhost pour obtenir la valeur d'un indicateur unique. Le support initial pour les processeurs Graviton2 (ARM64) a également été ajouté.

Trouver le numéro de version de votre extension Lambda

Utilisez la procédure suivante pour trouver le numéro de version de votre extension AWS AppConfig Agent Lambda actuellement configurée. Pour fonctionner correctement, votre fonction Lambda doit être configurée pour utiliser le nom de ressource Amazon (ARN) spécifique à l' Région AWS endroit où elle est actuellement hébergée.

1. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez la fonction Lambda à laquelle vous souhaitez ajouter la `AWS-AppConfig-Extension` couche.
3. Dans la zone Couches, choisissez Ajouter une couche.
4. Dans la section Choisir une couche, choisissez AWS- AppConfig -Extension dans la liste des AWS couches.

5. Utilisez la liste des versions pour choisir un numéro de version.
6. Choisissez Ajouter.
7. Utilisez l'onglet Test pour tester la fonction.
8. Une fois le test terminé, consultez la sortie du journal. Recherchez la version de l'extension AWS AppConfig Agent Lambda dans la section Détails de l'exécution. Cette version doit correspondre à celle requise URLs pour cette version.

plate-forme x86-64

Lorsque vous ajoutez l'extension sous forme de couche à votre Lambda, vous devez spécifier un ARN. Dans le tableau suivant, choisissez un ARN correspondant à l' Région AWS endroit où vous avez créé le Lambda. Elles ARNs concernent les fonctions Lambda développées pour la plate-forme x86-64.

Version 2.0.2037

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:207`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:162`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:258`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:262`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:152`

Région	ARN
Canada-Ouest (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:57
Europe (Francfort)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:189
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:106
Europe (Irlande)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:189
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:133
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:162
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:259
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:140
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:102

Région	ARN
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:133`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:131`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:142`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:155`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:165`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:159`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:156`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:199`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:150`

Région	ARN
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:78`
Asie-Pacifique (Malaisie)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:55`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:175`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:104`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:215`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:152`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:81`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:120`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:154`

Région	ARN
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:110`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:109`

ARM64plateforme

Lorsque vous ajoutez l'extension sous forme de couche à votre Lambda, vous devez spécifier un ARN. Dans le tableau suivant, choisissez un ARN correspondant à l' Région AWS endroit où vous avez créé le Lambda. Elles ARNs concernent les fonctions Lambda développées pour la ARM64 plateforme.

Version 2.0.2037

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:140`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:114`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:135`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:164`

Région	ARN
Canada (Centre)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:72
Canada-Ouest (Calgary)	arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:47
Europe (Francfort)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:132
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:64
Europe (Irlande)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:127
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:85
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:81
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:118
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:68

Région	ARN
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:63
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:70
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:108
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:73
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:74
Asie-Pacifique (Singapour)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:108
Asie-Pacifique (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:142
Asie-Pacifique (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:87
Asie-Pacifique (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:63

Région	ARN
Asie-Pacifique (Malaisie)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:30`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:117`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:62`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:103`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:80`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:76`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:82`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:64`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:55`

Région	ARN
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:53`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:56`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:55`

Anciennes versions d'extension

Cette section répertorie les versions ARNs et Régions AWS les anciennes versions de l'extension AWS AppConfig Lambda. Cette liste ne contient pas d'informations pour toutes les versions précédentes de l'extension AWS AppConfig Agent Lambda, mais elle sera mise à jour lorsque de nouvelles versions seront publiées.

Rubriques

- Anciennes versions d'extension (plate-forme x86-64)
- Anciennes versions d'extension (ARM64 plateforme)

Anciennes versions d'extension (plate-forme x86-64)

Les tableaux suivants répertorient ARNs les Régions AWS anciennes versions de l'extension AWS AppConfig Agent Lambda développée pour la plate-forme x86-64.

Date remplacée par une nouvelle extension : 20/05/2025

La version 2.0.1079

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:174`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:133`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:223`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:230`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:123`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:159`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:77`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:160`

Région	ARN
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:121
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:133
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:225
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:111
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:74
Chine (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:106
Chine (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:104
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:113
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:126

Région	ARN
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:136
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:130
Asie-Pacifique (Singapour)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:134
Asie-Pacifique (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:165
Asie-Pacifique (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:121
Asie-Pacifique (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:49
Asie-Pacifique (Malaisie)	arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:26
Asie-Pacifique (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:146
Asie-Pacifique (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:75

Région	ARN
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:179`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:123`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:52`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:91`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:125`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:80`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:80`

Date de remplacement par une nouvelle extension : 12/12/2024

La version 2.0.719

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:173`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:132`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:221`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:229`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:121`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:27`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:158`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:75`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:159`

Région	ARN
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:120
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:132
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:224
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:110
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:72
Chine (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:104
Chine (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:102
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:112
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:125

Région	ARN
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:135`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:129`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:132`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:164`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:120`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:48`
Asie-Pacifique (Malaisie)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension:25`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:145`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:74`

Région	ARN
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:178`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:122`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:50`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:90`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:124`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:79`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:79`

Date remplacée par une nouvelle extension : 08/08/2024

La version 2.0.678

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:167`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:126`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:213`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:223`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:116`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:21`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:152`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:70`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:153`

Région	ARN
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:114`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:126`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:218`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:104`
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:67`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:99`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:97`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:106`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:119`

Région	ARN
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:129
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:123
Asie-Pacifique (Singapour)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:127
Asie-Pacifique (Sydney)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:158
Asie-Pacifique (Jakarta)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:114
Asie-Pacifique (Melbourne)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:42
Asie-Pacifique (Mumbai)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:139
Asie-Pacifique (Hyderabad)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:68
Amérique du Sud (São Paulo)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:172

Région	ARN
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:116`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:45`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:84`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:118`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:73`

Date remplacée par une nouvelle extension : 23/07/2024

La version 2.0.501

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:153`

Région	ARN
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:112`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:195`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:210`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:101`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:136`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:53`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:144`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:99`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:111`

Région	ARN
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:201`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:89`
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:50`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:85`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:83`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:91`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:104`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:114`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:107`

Région	ARN
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:112`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:142`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:98`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:26`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:125`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:53`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:155`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:102`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:28`

Région	ARN
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:68`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:103`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:59`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:59`

Date remplacée par une nouvelle extension : 01/07/2024

La version 2.0.358

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141`

Région	ARN
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83`

Région	ARN
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:76`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:76`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106`

Région	ARN
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:83`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:22`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85`

Région	ARN
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54`

Date remplacée par une nouvelle extension : 01/12/2023

La version 2.0.181

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:113`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:81`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81`

Région	ARN
Europe (Francfort)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32
Europe (Irlande)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29
Chine (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:68

Région	ARN
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:68`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5`

Région	ARN
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:73`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:7`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46`

Date remplacée par une nouvelle extension : 14/08/2023

La version 2.0.165

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108`

Région	ARN
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26
Chine (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:66
Chine (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:66
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82

Région	ARN
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110`

Région	ARN
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44`

Date remplacée par une nouvelle extension : 21/02/2023

La version 2.0.122

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59`

Région	ARN
USA Ouest (Californie du Nord)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93
US West (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114
Canada (Centre)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59
Europe (Francfort)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70
Europe (Irlande)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54

Région	ARN
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:52`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:52`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37`

Région	ARN
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29`

Date de remplacement par une nouvelle extension : 23/08/2022

La version 2.0.58

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`

Région	ARN
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98`

Région	ARN
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59`

Région	ARN
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23`

Date de remplacement par une nouvelle extension : 21/04/2022

La version 2.0.45

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50`

Région	ARN
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46
Chine (Beijing)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45
Chine (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45
Asie-Pacifique (Singapour)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50

Région	ARN
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22`

Date de remplacement par une nouvelle extension : 15/03/2022

La version 2.0.30

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48`

Région	ARN
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42`
Asia Pacific (Seoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45`

Région	ARN
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20`

Anciennes versions d'extension (ARM64 plateforme)

Les tableaux suivants répertorient ARNs les versions Régions AWS les plus anciennes de l'extension AWS AppConfig Agent Lambda développée pour la ARM64 plateforme.

Date remplacée par une nouvelle extension : 20/05/2025

La version 2.0.1079

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:107`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:85`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:100`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:132`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:43`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:18`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:102`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:35`

Région	ARN
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:98`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:73`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:52`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:84`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:39`
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:35`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:41`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:79`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:44`

Région	ARN
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:45`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:86`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:108`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:58`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:34`
Asie-Pacifique (Malaisie)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:88`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:33`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:67`

Région	ARN
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:51`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:47`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:53`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:35`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:28`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:26`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:26`

Date de remplacement par une nouvelle extension : 12/12/2024

La version 2.0.678

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:106`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:84`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:98`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:131`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:41`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension-Arm64:17`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:101`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:33`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:97`

Région	ARN
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:72
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:51
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:83
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:38
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:33
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:40
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:78
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:43
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:44

Région	ARN
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:84`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:107`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:57`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:33`
Asie-Pacifique (Malaisie)	`arn:aws:lambda:ap-southeast-5:631746059939:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:87`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:32`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:66`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:50`

Région	ARN
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:46`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:52`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:33`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:26`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:24`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:25`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:25`

Date remplacée par une nouvelle extension : 08/08/2024

La version 2.0.678

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:100`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:78`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:90`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:125`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:11`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:36`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:95`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:28`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:91`

Région	ARN
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:66
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:45
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:77
Europe (Milan)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:32
Europe (Espagne)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:28
Asie-Pacifique (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:34
Asie-Pacifique (Tokyo)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:72
Asie-Pacifique (Séoul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:37
Asie-Pacifique (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:38

Région	ARN
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:79`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:101`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:51`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:27`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:81`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:26`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:60`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:44`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:40`

Région	ARN
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:46`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:28`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:21`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:19`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:19`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:19`

Date remplacée par une nouvelle extension : 23/07/2024

La version 2.0.501

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:86`

Région	ARN
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:64`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:72`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:112`
Canada-Ouest (Calgary)	`arn:aws:lambda:ca-west-1:436199621743:layer:AWS-AppConfig-Extension:1`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:21`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:79`
Europe (Zurich)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:11`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:82`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:51`

Région	ARN
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:30`
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:60`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:17`
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:11`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:19`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:57`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:22`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:22`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:64`

Région	ARN
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:85`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:35`
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:11`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:67`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:11`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:43`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:30`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:24`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:31`

Région	ARN
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:11`
Chine (Beijing)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension-Arm64:7`
Chine (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension-Arm64:5`
AWS GovCloud (USA Est)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension-Arm64:5`
AWS GovCloud (US-Ouest)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension-Arm64:5`

Date remplacée par une nouvelle extension : 01/07/2024

La version 2.0.358

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45`

Région	ARN
USA Ouest (Californie du Nord)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18
US West (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63
Canada (Centre)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13
Europe (Francfort)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49
Europe (Zurich)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5
Europe (Irlande)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63
Europe (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45
Europe (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17
Europe (Stockholm)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18

Région	ARN
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11`
Europe (Espagne)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16`

Région	ARN
Asie-Pacifique (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49`
Asie-Pacifique (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11`
Moyen-Orient (EAU)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:5`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:13`
Israël (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:5`

Date remplacée par une nouvelle extension : 01/12/2023

La version 2.0.181

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:46`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:33`
USA Ouest (Californie du Nord)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:1`
US West (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:48`
Canada (Centre)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:1`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:36`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:48`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33`
Europe (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1`

Région	ARN
Europe (Stockholm)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1`
Europe (Milan)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37`
Asie-Pacifique (Séoul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36`
Asie-Pacifique (Jakarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1`

Région	ARN
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36`
Amérique du Sud (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1`
Afrique (Le Cap)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1`
Moyen-Orient (Bahreïn)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1`

Date remplacée par une nouvelle extension : 30/03/2023

La version 2.0.165

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31`
USA Ouest (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45`

Région	ARN
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34`

Date remplacée par une nouvelle extension : 21/02/2023

La version 2.0.122

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:15`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:11`
USA Ouest (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:16`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:13`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:20`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:11`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:15`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:16`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13`

Région	ARN
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:13`

Date de remplacement par une nouvelle extension : 23/08/2022

La version 2.0.58

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2`
USA Ouest (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3`
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2`

Région	ARN
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2`

Date de remplacement par une nouvelle extension : 21/04/2022

La version 2.0.45

Région	ARN
USA Est (Virginie du Nord)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1`
USA Est (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1`
USA Ouest (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2`

Région	ARN
Europe (Francfort)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1`
Europe (Irlande)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6`
Europe (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Tokyo)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Singapour)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2`
Asie-Pacifique (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1`
Asie-Pacifique (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1`

Utilisation de AWS AppConfig l'agent avec Amazon EC2 et des machines sur site

Vous pouvez intégrer AWS AppConfig des applications exécutées sur vos instances Linux Amazon Elastic Compute Cloud (Amazon EC2) à l'aide de l' AWS AppConfig agent. L'agent améliore le traitement et la gestion des demandes de la manière suivante :

- L'agent appelle AWS AppConfig en votre nom en utilisant un rôle AWS Identity and Access Management (IAM) et en gérant un cache local de données de configuration. En extrayant les données de configuration du cache local, votre application nécessite moins de mises à jour de code pour gérer les données de configuration, récupère les données de configuration en quelques millisecondes et n'est pas affectée par les problèmes réseau susceptibles de perturber les appels pour ces données. *
- L'agent propose une expérience native pour récupérer et résoudre les indicateurs de AWS AppConfig fonctionnalités.
- Prêt à l'emploi, l'agent fournit les meilleures pratiques en matière de stratégies de mise en cache, d'intervalles d'interrogation et de disponibilité des données de configuration locales, tout en suivant les jetons de configuration nécessaires pour les appels de service suivants.
- Lorsqu'il s'exécute en arrière-plan, l'agent interroge régulièrement le plan de AWS AppConfig données pour les mises à jour des données de configuration. Votre application peut récupérer les données en se connectant à localhost sur le port 2772 (une valeur de port par défaut personnalisable) et en appelant HTTP GET pour récupérer les données.

*AWS AppConfig L'agent met en cache les données la première fois que le service récupère vos données de configuration. Pour cette raison, le premier appel pour récupérer les données est plus lent que les appels suivants.

Rubriques

Étape 1 : (Obligatoire) Création de ressources et configuration des autorisations

Pour intégrer AWS AppConfig les applications exécutées sur vos EC2 instances Amazon, vous devez créer des AWS AppConfig artefacts et des données de configuration, notamment des indicateurs de

fonctionnalité ou des données de configuration sous forme libre. Pour de plus amples informations, veuillez consulter Création d'indicateurs de fonctionnalités et de données de configuration sous forme libre dans AWS AppConfig.

Pour récupérer les données de configuration hébergées par AWS AppConfig, vos applications doivent être configurées de manière à accéder au plan de AWS AppConfig données. Pour autoriser l'accès à vos applications, mettez à jour la politique d'autorisation IAM attribuée au rôle d' EC2 instance Amazon. Plus précisément, vous devez ajouter les `appconfig:GetLatestConfiguration` actions `appconfig:StartConfigurationSession` et à la politique. Voici un exemple :

JSON

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

Pour plus d'informations sur l'ajout d'autorisations à une politique, consultez la section Ajouter et supprimer des autorisations d'identité IAM dans le guide de l'utilisateur IAM.

Étape 2 : (obligatoire) Installation et démarrage de AWS AppConfig l'agent sur les EC2 instances Amazon

AWS AppConfig L'agent est hébergé dans un compartiment Amazon Simple Storage Service (Amazon S3) géré par. AWS Utilisez la procédure suivante pour installer la dernière version de l'agent sur votre instance Linux. Si votre application est distribuée sur plusieurs instances, vous devez exécuter cette procédure sur chaque instance hébergeant l'application.

> **Note**
>
> Veuillez noter les informations suivantes :
>
> - AWS AppConfig L'agent est disponible pour les systèmes d'exploitation Linux exécutant la version 4.15 ou supérieure du noyau. Les systèmes basés sur Debian, tels qu'Ubuntu, ne sont pas pris en charge.
> - L'agent prend en charge x86_64 et les architectures. ARM64
> - Pour les applications distribuées, nous recommandons d'ajouter les commandes d'installation et de démarrage aux données EC2 utilisateur Amazon de votre groupe Auto Scaling. Dans ce cas, chaque instance exécute les commandes automatiquement. Pour plus d'informations, consultez la section Exécuter des commandes sur votre instance Linux au lancement dans le guide de EC2 l'utilisateur Amazon. Consultez également Tutoriel : Configurer les données utilisateur pour récupérer l'état du cycle de vie cible via les métadonnées de l'instance dans le manuel Amazon EC2 Auto Scaling User Guide.
> - Les procédures décrites dans cette rubrique décrivent comment effectuer des actions telles que l'installation de l'agent en vous connectant à l'instance pour exécuter la commande. Vous pouvez exécuter les commandes depuis un ordinateur client local et cibler une ou plusieurs instances à l'aide de l'outil Run Command, qui est un outil de AWS Systems Manager. Pour plus d'informations, consultez AWS Systems Manager Run Command dans le AWS Systems Manager Guide de l'utilisateur.
> - AWS AppConfig L'agent sur les instances Amazon EC2 Linux est un `systemd` service.

Pour installer et démarrer AWS AppConfig l'agent sur une instance

1. Connectez-vous à votre instance Linux.
2. Ouvrez un terminal et exécutez l'une des commandes suivantes avec les autorisations d'administrateur :

   x86_64

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

   ARM64

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

Si vous souhaitez installer une version spécifique de l' AWS AppConfig Agent, remplacez `latest` l'URL par un numéro de version spécifique. Voici un exemple pour x86_64 :

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

3. Exécutez la commande suivante pour démarrer l'agent :

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

4. Exécutez la commande suivante pour vérifier que l'agent est en cours d'exécution :

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

En cas de succès, la commande renvoie des informations telles que les suivantes :

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

> **Note**
>
> Pour arrêter l'agent, exécutez la commande suivante :
>
> ```
> sudo systemctl stop aws-appconfig-agent
> ```

Étape 3 : (Facultatif, mais recommandé) Envoi de fichiers CloudWatch journaux à Logs

Par défaut, AWS AppConfig l'agent publie les journaux sur STDERR. Systemd redirige STDOUT et STDERR pour tous les services exécutés sur l'instance Linux vers le journal systemd. Vous pouvez afficher et gérer les données du journal dans le journal systemd si vous n'exécutez AWS AppConfig l'Agent que sur une ou deux instances. Une meilleure solution, que nous recommandons vivement

pour les applications distribuées, consiste à écrire des fichiers journaux sur disque, puis à utiliser l' CloudWatch agent Amazon pour télécharger les données du journal AWS dans le cloud. En outre, vous pouvez configurer l' CloudWatch agent pour supprimer les anciens fichiers journaux de votre instance, afin d'éviter que celle-ci ne manque d'espace disque.

Pour activer la journalisation sur le disque, vous devez définir la variable d'`LOG_PATH`environnement, comme décrit dansÉtape 4 : (Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon EC2.

Pour commencer à utiliser l' CloudWatch agent, consultez la section Collecter des métriques et des journaux à partir d' EC2 instances Amazon et de serveurs locaux avec l' CloudWatch agent dans le guide de l' CloudWatch utilisateur Amazon. Vous pouvez utiliser Quick Setup, un outil de Systems Manager pour installer rapidement l' CloudWatch agent. Pour plus d'informations, voir Configuration rapide de la gestion des hôtes dans le guide de AWS Systems Manager l'utilisateur.

> **Warning**
>
> Si vous choisissez d'écrire des fichiers journaux sur disque sans utiliser l' CloudWatch agent, vous devez supprimer les anciens fichiers journaux. AWS AppConfig L'agent fait automatiquement pivoter les fichiers journaux toutes les heures. Si vous ne supprimez pas les anciens fichiers journaux, votre instance risque de manquer d'espace disque.

Après avoir installé l' CloudWatch agent sur votre instance, créez un fichier de configuration de CloudWatch l'agent. Le fichier de configuration indique à CloudWatch l'agent comment utiliser les fichiers journaux de AWS AppConfig l'agent. Pour plus d'informations sur la création d'un fichier de configuration de CloudWatch l'agent, consultez la section Création du fichier de configuration de l' CloudWatch agent.

Ajoutez la `logs` section suivante au fichier de configuration de l' CloudWatch agent sur l'instance et enregistrez vos modifications :

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

```
          "auto_removal": true
        },
        ...
      ]
    },
    ...
  },
  ...
}
```

Si la valeur `auto_removal` est est`true`, l' CloudWatch agent supprime automatiquement les fichiers journaux de l' AWS AppConfig agent pivotés.

Étape 4 : (Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon EC2

Vous pouvez configurer AWS AppConfig l'agent pour Amazon à EC2 l'aide de variables d'environnement. Pour définir les variables d'environnement d'un `systemd` service, vous devez créer un fichier d'unité intégré. L'exemple suivant montre comment créer un fichier d'unité intégré pour définir le niveau de journalisation de l' AWS AppConfig agent sur`DEBUG`.

Exemple de création d'un fichier d'unité intégré pour les variables d'environnement

1. Connectez-vous à votre instance Linux.
2. Ouvrez un terminal et exécutez la commande suivante avec les autorisations d'administrateur. La commande crée un répertoire de configuration :

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

3. Exécutez la commande suivante pour créer le fichier d'unité intégré. Remplacez *`file_name`* par le nom du fichier. L'extension doit être `.conf` :

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

4. Entrez les informations dans le fichier de l'unité d'accueil. L'exemple suivant ajoute une `Service` section qui définit une variable d'environnement. L'exemple définit le niveau de journalisation de l' AWS AppConfig agent sur`DEBUG`.

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

5. Exécutez la commande suivante pour recharger la configuration systemd :

```
sudo systemctl daemon-reload
```

6. Exécutez la commande suivante pour redémarrer AWS AppConfig l'agent :

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

Vous pouvez configurer AWS AppConfig Agent for Amazon EC2 en spécifiant les variables d'environnement suivantes dans un fichier d'unité intégré.

> **Note**
>
> Le tableau suivant inclut une colonne de valeurs d'exemple. Selon la résolution de votre écran, vous devrez peut-être faire défiler le tableau vers le bas, puis vers la droite pour afficher la colonne.

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`ACCESS_TOKEN`	Cette variable d'environnement définit un jeton qui doit être fourni lors de la demande de données de configuration auprès du serveur HTTP de l'agent. La valeur du jeton doit être définie dans l'en-tête d'autorisation de la demande HTTP avec un type d'autorisation de`Bearer`. Voici un exemple.	Aucun	MyAccessToken

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon

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

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`BACKUP_DIRECTORY`	Cette variable d'environnement permet à l' AWS AppConfig agent d'enregistrer une sauvegarde de chaque configuration récupérée dans le répertoire spécifié. **Important** Les configurations sauvegardées sur disque ne sont pas cryptées. Si votre configuration contient des données sensibles, il est AWS AppConfig recommandé de mettre en pratique le principe du moindre privilège avec les autorisations de votre système	Aucun	/path/to/backups

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
	de fichiers. Pour de plus amples informations, veuillez consulter Sécurité dans AWS AppConfig.		
`HTTP_PORT`	Cette variable d'environnement indique le port sur lequel s'exécute le serveur HTTP de l'agent.	2772	2772

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`LOG_LEVEL`	Cette variable d'environnement indique le niveau de détail enregistré par l'agent. Chaque niveau inclut le niveau actuel et tous les niveaux supérieurs. La valeur ne distingue pas les majuscules et minuscules. Du plus détaillé au moins détaillé, les niveaux de journalisation sont les suivants : `trace` `debug` `info` `warn` `error` `fatal` `none`. Le `trace` journal contient des informations détaillées, y compris des informations temporelles, sur l'agent.	info	tracer debug info prévenir error fatal none
`LOG_PATH`	Emplacement du disque où les journaux sont écrits. Si ce n'est pas spécifié, les journaux sont écrits dans stderr.	Aucun	/path/to/logs/agent.journal

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour tirer parti de fonctionnalités supplémentaires par configuration, telles que la récupération de plusieurs comptes et l'enregistrement de la configuration sur disque. Pour de plus amples informations sur l'utilisation de ces modèles, consultez Utilisation d'un manifeste pour activer des fonctionnalités de récupération supplémentaires.	Aucun	Lorsque vous utilisez AWS AppConfig la configuration comme manifeste :`MyApp:MyEnv:MyManifestConfig` . Lors du chargement du manifeste depuis le disque : `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Cette variable d'environnement configure le nombre maximal de connexions que l'agent utilise pour récupérer des AWS AppConfig configurations.	3	3

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`POLL_INTERVAL`	Cette variable d'environnement contrôle la fréquence à laquelle l'agent interroge les données AWS AppConfig de configuration mises à jour. Vous pouvez spécifier un nombre de secondes pour l'intervalle. Vous pouvez également spécifier un nombre avec une unité de temps : s pour les secondes, m pour les minutes et h pour les heures. Si aucune unité n'est spécifiée, l'agent utilise par défaut les secondes. Par exemple, 60, 60 s et 1 m donnent le même intervalle d'interrogation.	45 secondes	45 45 s 5 min 1 h

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
PREFETCH_LIST	Cette variable d'environnement spécifie les données de configuration que l'agent demande AWS AppConfig dès son démarrage. Plusieurs identifiants de configuration peuvent être fournis dans une liste séparée par des virgules.	Aucun	MyApp:MyEnv:MyConfig abcd123 : efgh456 : ijkl789 MyApp::Config1, ::Config2 MyEnv MyApp MyEnv

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
PRELOAD_BACKUPS	S'il est défini sur`true`, AWS AppConfig l'agent charge les sauvegardes de configuration présentes `BACKUP_DIRECTORY` dans la mémoire et vérifie immédiatement s'il existe une version plus récente du service. S'il est défini sur`false`, l' AWS AppConfig Agent charge le contenu d'une sauvegarde de configuration uniquement s'il ne peut pas récupérer les données de configuration du service, par exemple en cas de problème avec votre réseau.	true	vrai false

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
PROXY_HEADERS	Cette variable d'environnement spécifie les en-têtes requis par le proxy référencé dans la variable d'PROXY_URL environnement. La valeur est une liste d'en-têtes séparés par des virgules.	Aucun	en-tête : valeur h1 : v1, h2 : v2
PROXY_URL	Cette variable d'environnement spécifie l'URL du proxy à utiliser pour les connexions entre l'agent Services AWS et, notamment AWS AppConfig. HTTPSet HTTP URLs sont pris en charge.	Aucun	http://localhost:7474 https://my-proxy.example.com

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`REQUEST_TIMEOUT`	Cette variable d'environnement contrôle le temps pendant lequel l'agent attend une réponse. AWS AppConfig Si le service ne répond pas, la demande échoue. Si la demande concerne la récupération initiale des données, l'agent renvoie une erreur à votre application. Si le délai d'attente survient lors d'une vérification des données mises à jour en arrière-plan, l'agent enregistre l'erreur et réessaie après un court laps de temps. Vous pouvez spécifier le nombre de millisecondes pour le délai d'expiration. Vous pouvez également spécifier un nombre avec une unité de	3000 ms	3000 3000 ms 5s

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
	temps : ms pour les millisecondes et s pour les secondes. Si aucune unité n'est spécifiée, l'agent utilise par défaut les millisecondes. Par exemple, 5 000, 5 000 ms et 5 s entraînent la même valeur de délai d'expiration de la demande.		
`ROLE_ARN`	Cette variable d'environnement spécifie le nom de ressource Amazon (ARN) d'un rôle IAM. AWS AppConfig L'agent assume ce rôle pour récupérer les données de configuration.	Aucun	arn:aws:iam : :123456789012:role/ MyRole
`ROLE_EXTERNAL_ID`	Cette variable d'environnement indique l'ID externe à utiliser avec l'ARN du rôle assumé.	Aucun	MyExternalId

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`ROLE_SESSION_NAME`	Cette variable d'environnement spécifie le nom de session à associer aux informations d'identification pour le rôle IAM assumé.	Aucun	AWSAppConfigAgentSession
`SERVICE_REGION`	Cette variable d'environnement indique une alternative Région AWS que AWS AppConfig l'agent utilise pour appeler le AWS AppConfig service. Si ce paramètre n'est pas défini, l'agent tente de déterminer la région actuelle. Si ce n'est pas le cas, l'agent ne démarre pas.	Aucun	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour qu'il attende que le manifeste soit traité avant de terminer le démarrage.	true	vrai false

Étape 5 : (Obligatoire) Récupération des données de configuration

Vous pouvez récupérer les données de configuration de AWS AppConfig l'agent à l'aide d'un appel HTTP localhost. Les exemples suivants sont utilisés `curl` avec un client HTTP. Vous pouvez appeler l'agent en utilisant n'importe quel client HTTP compatible avec le langage de votre application ou les bibliothèques disponibles, y compris un AWS SDK.

Pour récupérer le contenu complet de toute configuration déployée

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

Pour récupérer un seul drapeau et ses attributs à partir d'une AWS AppConfig configuration de type **`Feature Flag`**

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

Pour accéder à plusieurs drapeaux et à leurs attributs à partir d'une AWS AppConfig configuration de type **`Feature Flag`**

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

Étape 6 (facultative, mais recommandée) : Automatisation des mises à AWS AppConfig jour de l'agent

AWS AppConfig L'agent est mis à jour périodiquement. Pour vous assurer que vous exécutez la dernière version d' AWS AppConfig Agent sur vos instances, nous vous recommandons d'ajouter les commandes suivantes à vos données EC2 utilisateur Amazon. Vous pouvez ajouter les commandes aux données utilisateur sur l'instance ou sur le groupe EC2 Auto Scaling. Le script installe et démarre la dernière version de l'agent chaque fois qu'une instance démarre ou redémarre.

```
#!/bin/bash
# install the latest version of the agent
yum install -y https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/
linux/x86_64/latest/aws-appconfig-agent.rpm
```

```
# optional: configure the agent
mkdir /etc/systemd/system/aws-appconfig-agent.service.d
echo "${MY_AGENT_CONFIG}" > /etc/systemd/system/aws-appconfig-agent.service.d/
overrides.conf
systemctl daemon-reload
# start the agent
systemctl start aws-appconfig-agent
```

Utilisation de AWS AppConfig l'agent avec Amazon ECS et Amazon EKS

Vous pouvez intégrer Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS) à l' AWS AppConfig aide d'un agent. AWS AppConfig L'agent fonctionne comme un conteneur annexe s'exécutant parallèlement à vos applications de conteneur Amazon ECS et Amazon EKS. L'agent améliore le traitement et la gestion des applications conteneurisées de la manière suivante :

- L'agent appelle AWS AppConfig en votre nom en utilisant un rôle AWS Identity and Access Management (IAM) et en gérant un cache local de données de configuration. En extrayant les données de configuration du cache local, votre application nécessite moins de mises à jour de code pour gérer les données de configuration, récupère les données de configuration en quelques millisecondes et n'est pas affectée par les problèmes réseau susceptibles de perturber les appels pour ces données. *
- L'agent propose une expérience native pour récupérer et résoudre les indicateurs de AWS AppConfig fonctionnalités.
- Prêt à l'emploi, l'agent fournit les meilleures pratiques en matière de stratégies de mise en cache, d'intervalles d'interrogation et de disponibilité des données de configuration locales, tout en suivant les jetons de configuration nécessaires pour les appels de service suivants.
- Lorsqu'il s'exécute en arrière-plan, l'agent interroge régulièrement le plan de AWS AppConfig données pour les mises à jour des données de configuration. Votre application conteneurisée peut récupérer les données en se connectant à localhost sur le port 2772 (une valeur de port par défaut personnalisable) et en appelant HTTP GET pour récupérer les données.
- AWS AppConfig L'agent met à jour les données de configuration de vos conteneurs sans avoir à redémarrer ou à recycler ces conteneurs.

*AWS AppConfig L'agent met en cache les données la première fois que le service récupère vos données de configuration. Pour cette raison, le premier appel pour récupérer les données est plus lent que les appels suivants.

Avant de commencer

Pour intégrer vos applications AWS AppConfig de conteneur, vous devez créer des AWS AppConfig artefacts et des données de configuration, notamment des indicateurs de fonctionnalité ou des données de configuration sous forme libre. Pour de plus amples informations, veuillez consulter Création d'indicateurs de fonctionnalités et de données de configuration sous forme libre dans AWS AppConfig.

Pour récupérer les données de configuration hébergées par AWS AppConfig, vos applications de conteneur doivent être configurées de manière à accéder au plan de AWS AppConfig données. Pour autoriser l'accès à vos applications, mettez à jour la politique d'autorisations IAM utilisée par votre rôle IAM de service de conteneurs. Plus précisément, vous devez ajouter les `appconfig:GetLatestConfiguration` actions `appconfig:StartConfigurationSession` et à la politique. Les rôles IAM du service de conteneur sont les suivants :

- Le rôle de tâche Amazon ECS
- Le rôle du nœud Amazon EKS
- Le rôle d'exécution du AWS Fargate pod (si vos conteneurs Amazon EKS utilisent Fargate pour le traitement informatique)

Pour plus d'informations sur l'ajout d'autorisations à une politique, consultez la section Ajouter et supprimer des autorisations d'identité IAM dans le guide de l'utilisateur IAM.

Rubriques

Démarrage de l' AWS AppConfig agent pour l'intégration d'Amazon ECS

Le conteneur annexe de l' AWS AppConfig agent est automatiquement disponible dans votre environnement Amazon ECS. Pour l'utiliser, vous devez le démarrer, comme décrit dans la procédure suivante.

Pour démarrer Amazon ECS (console)

1. Ouvrez la console à la https://console.aws.amazon.com/ecs/version 2.
2. Dans le panneau de navigation, choisissez Task definitions (Définition des tâches).
3. Choisissez la définition de tâche pour votre application, puis sélectionnez la dernière révision.
4. Choisissez Créer une nouvelle révision, puis Créer une nouvelle révision.
5. Choisissez Ajouter d'autres conteneurs.
6. Dans Nom, entrez un nom unique pour le conteneur de l' AWS AppConfig agent.
7. Pour l'URI de l'image, entrez : **`public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x`**
8. Pour le contenant Essential, sélectionnez Oui.
9. Dans la section Mappages de ports, choisissez Ajouter un mappage de port.
10. Pour Port à conteneurs, entrez**`2772`**.

> **Note**
>
> AWS AppConfig L'agent s'exécute sur le port 2772, par défaut. Vous pouvez spécifier un autre port.

11. Choisissez Créer. Amazon ECS crée une nouvelle révision de conteneur et affiche les détails.
12. Dans le volet de navigation, choisissez Clusters, puis choisissez votre cluster d'applications dans la liste.
13. Dans l'onglet Services, sélectionnez le service correspondant à votre application.
14. Choisissez Mettre à jour.
15. Sous Configuration du déploiement, pour Révision, choisissez la dernière révision.
16. Choisissez Mettre à jour. Amazon ECS déploie la dernière définition de tâche.
17. Une fois le déploiement terminé, vous pouvez vérifier que AWS AppConfig l'agent est en cours d'exécution dans l'onglet Configuration et tâches. Dans l'onglet Tâches, choisissez la tâche en cours d'exécution.

18. Dans la section Conteneurs, vérifiez que le conteneur de l' AWS AppConfig agent est répertorié.
19. Pour vérifier que AWS AppConfig l'agent a démarré, cliquez sur l'onglet Logs. Recherchez une instruction semblable à la suivante pour le conteneur de l' AWS AppConfig agent : `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Notez les informations suivantes.
>
> - AWS AppConfig L'agent est un processus de longue haleine. En tant que bonne pratique pour les conteneurs Amazon ECS, configurez les contrôles de santé de vos conteneurs, en définissant spécifiquement la dépendance du conteneur sur la condition SAINE. Pour plus d'informations, consultez le ContainerDependencymanuel Amazon Elastic Container Service API Reference.
> - Vous pouvez ajuster le comportement par défaut de l' AWS AppConfig Agent en saisissant ou en modifiant des variables d'environnement. Pour plus d'informations sur les variables d'environnement disponibles, consultez(Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon ECS et Amazon EKS. Pour plus d'informations sur la modification des variables d'environnement dans Amazon ECS, consultez la section Transmission de variables d'environnement à un conteneur dans le manuel Amazon Elastic Container Service Developer Guide.

Démarrage de l' AWS AppConfig agent pour l'intégration d'Amazon EKS

Le conteneur annexe de l' AWS AppConfig agent est automatiquement disponible dans votre environnement Amazon EKS. Pour l'utiliser, vous devez le démarrer. La procédure suivante décrit comment utiliser l'outil de ligne de `kubectl` commande Amazon EKS pour démarrer l'agent.

> **Note**
>
> Avant de continuer, assurez-vous que votre `kubeconfig` fichier est à jour. Pour plus d'informations sur la création ou la modification d'un `kubeconfig` fichier, consultez la section Création ou mise à jour d'un fichier kubeconfig pour un cluster Amazon EKS dans le guide de l'utilisateur Amazon EKS.

Pour démarrer AWS AppConfig l'agent (outil de ligne de commande kubectl)

1. Ouvrez le manifeste de votre application et vérifiez que votre application Amazon EKS s'exécute en tant que déploiement à conteneur unique. Le contenu du fichier doit ressembler à ce qui suit.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-application-label
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-application-label
  template:
    metadata:
      labels:
        app: my-application-label
    spec:
      containers:
      - name: my-app
        image: my-repo/my-image
        imagePullPolicy: IfNotPresent
```

2. Ajoutez les détails de la définition du conteneur de l' AWS AppConfig agent à votre manifeste de déploiement.

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

> **Note**
>
> Notez les informations suivantes.
>
> - AWS AppConfig L'agent s'exécute sur le port 2772, par défaut. Vous pouvez spécifier un autre port.
> - Vous pouvez ajuster le comportement par défaut de l' AWS AppConfig Agent en saisissant des variables d'environnement. Pour de plus amples informations, veuillez consulter (Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon ECS et Amazon EKS.
> - Pour*Région AWS*, spécifiez le Région AWS code (par exemple,`us-west-1`) dans lequel AWS AppConfig l'agent récupère les données de configuration.

3. Exécutez la `kubectl` commande suivante pour appliquer les modifications à votre cluster. *my-deployment*Remplacez-le par le nom de votre manifeste de déploiement.

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

4. Une fois le déploiement terminé, vérifiez que AWS AppConfig l'agent est en cours d'exécution. Utilisez la commande suivante pour afficher le fichier journal du pod de l'application.

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

Recherchez une instruction semblable à la suivante pour le conteneur de l' AWS AppConfig agent : `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Vous pouvez ajuster le comportement par défaut de l' AWS AppConfig Agent en saisissant ou en modifiant des variables d'environnement. Pour plus d'informations sur les variables d'environnement disponibles, consultez(Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon ECS et Amazon EKS.

(Facultatif) Exécution DaemonSet en AWS AppConfig tant que dans Amazon EKS

Avec Amazon EKS, vous pouvez exécuter AWS AppConfig l'agent en tant que sidecar, ce qui se traduit par un conteneur d'agent par module d'application. Ou, si vous préférez, vous pouvez exécuter l' AWS AppConfig agent en tant que tel DaemonSet, ce qui se traduit par un conteneur d'agent par nœud de votre cluster.

> **Note**
>
> Si vous exécutez AWS AppConfig l'Agent en tant que DaemonSet, l'agent s'exécute dans un module séparé, ce qui signifie que vous ne pouvez pas y accéder avec des appels à`localhost`. Vous devez injecter ou découvrir l'adresse IP de l'agent pod pour pouvoir l'appeler.

Pour exécuter AWS AppConfig l'Agent en tant que DaemonSet, créez un fichier manifeste avec le contenu suivant. Remplacez *highlighted* le texte par des informations relatives à votre application et à votre environnement. Pour*Région AWS*, spécifiez un Région AWS code (par exemple,`us-west-1`).

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: aws-appconfig-agent
  namespace: my_namespace
  labels:
    app: my_application_label
spec:
   selector:
    matchLabels:
      app: my_application_label
  template:
    metadata:
      labels:
        app: my_application_label
    spec:
      containers:
      - name: aws-appconfig-agent
        image: public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x
        ports:
        - name: http
```

```
          containerPort: 2772
          protocol: TCP
        env:
        - name: SERVICE_REGION
          value: Région AWS
        imagePullPolicy: IfNotPresent
      # set a high priority class to ensure the agent is running on every node
      priorityClassName: system-node-critical
```

Exécutez la commande suivante pour appliquer l' AWS AppConfig agent DaemonSet à votre cluster. Remplacez *aws_appconfig_agent_daemonset* par le nom de votre DaemonSet manifeste.

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

(Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon ECS et Amazon EKS

Vous pouvez configurer AWS AppConfig l'agent en modifiant les variables d'environnement suivantes pour votre conteneur d'agents.

> **Note**
>
> Le tableau suivant inclut une colonne de valeurs d'exemple. Selon la résolution de votre écran, vous devrez peut-être faire défiler le tableau vers le bas, puis vers la droite pour afficher la colonne.

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
ACCESS_TOKEN	Cette variable d'environnement définit un jeton qui doit être fourni lors de la demande de données de configuration auprès du serveur HTTP de l'agent. La	Aucun	MyAccessToken

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon

	valeur du jeton doit être définie dans l'en-tête d'autorisation de la demande HTTP avec un type d'autorisation de`Bearer`. Voici un exemple.
	`GET /applications/my_app/...`
	`Host: localhost:2772`
	`Authorization: Bearer <token value>`		

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`BACKUP_DIRECTORY`	Cette variable d'environnement permet à l' AWS AppConfig agent d'enregistrer une sauvegarde de chaque configuration récupérée dans le répertoire spécifié. **Important** Les configurations sauvegardées sur disque ne sont pas cryptées. Si votre configuration contient des données sensibles, il est AWS AppConfig recommandé de mettre en pratique le principe du moindre privilège avec les autorisations de votre système	Aucun	/path/to/backups

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
	de fichiers. Pour de plus amples informations, veuillez consulter Sécurité dans AWS AppConfig.		
HTTP_PORT	Cette variable d'environnement indique le port sur lequel s'exécute le serveur HTTP de l'agent.	2772	2772

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`LOG_LEVEL`	Cette variable d'environnement indique le niveau de détail enregistré par l'agent. Chaque niveau inclut le niveau actuel et tous les niveaux supérieurs. La valeur ne distingue pas les majuscules et minuscules. Du plus détaillé au moins détaillé, les niveaux de journalisation sont les suivants : `trace` `debug` `info` `warn`, `error`, `fatal` et `none`. Le `trace` journal contient des informations détaillées, y compris des informations temporelles, sur l'agent.	info	tracer debug info prévenir error fatal none
`LOG_PATH`	Emplacement du disque où les journaux sont écrits. Si ce n'est pas spécifié, les journaux sont écrits dans stderr.	Aucun	/path/to/logs/agent.journal

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour tirer parti de fonctionnalités supplémentaires par configuration, telles que la récupération de plusieurs comptes et l'enregistrement de la configuration sur disque. Pour de plus amples informations sur l'utilisation de ces modèles, consultez Utilisation d'un manifeste pour activer des fonctionnalités de récupération supplémentaires.	Aucun	Lorsque vous utilisez AWS AppConfig la configuration comme manifeste :`MyApp:MyEnv:MyManifestConfig` . Lors du chargement du manifeste depuis le disque : `file:/path/to/manifest.json`
`MAX_CONNECTIONS`	Cette variable d'environnement configure le nombre maximal de connexions que l'agent utilise pour récupérer des AWS AppConfig configurations.	3	3

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`POLL_INTERVAL`	Cette variable d'environnement contrôle la fréquence à laquelle l'agent interroge les données AWS AppConfig de configuration mises à jour. Vous pouvez spécifier un nombre de secondes pour l'intervalle. Vous pouvez également spécifier un nombre avec une unité de temps : s pour les secondes, m pour les minutes et h pour les heures. Si aucune unité n'est spécifiée, l'agent utilise par défaut les secondes. Par exemple, 60, 60 s et 1 m donnent le même intervalle d'interrogation.	45 secondes	45 45 s 5 min 1 h

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
PREFETCH_LIST	Cette variable d'environnement spécifie les données de configuration que l'agent demande AWS AppConfig dès son démarrage. Plusieurs identifiants de configuration peuvent être fournis dans une liste séparée par des virgules.	Aucun	MyApp:MyEnv:MyConfig abcd123 : efgh456 : ijkl789 MyApp::Config1, ::Config2 MyEnv MyApp MyEnv

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`PRELOAD_BACKUPS`	S'il est défini sur`true`, AWS AppConfig l'agent charge les sauvegardes de configuration présentes `BACKUP_DIRECTORY` dans la mémoire et vérifie immédiatement s'il existe une version plus récente du service. S'il est défini sur`false`, l' AWS AppConfig Agent charge le contenu d'une sauvegarde de configuration uniquement s'il ne peut pas récupérer les données de configuration du service, par exemple en cas de problème avec votre réseau.	true	vrai false

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
PROXY_HEADERS	Cette variable d'environnement spécifie les en-têtes requis par le proxy référencé dans la variable d'PROXY_URL environnement. La valeur est une liste d'en-têtes séparés par des virgules.	Aucun	en-tête : valeur h1 : v1, h2 : v2
PROXY_URL	Cette variable d'environnement spécifie l'URL du proxy à utiliser pour les connexions entre l'agent Services AWS et, notamment AWS AppConfig. HTTPSet HTTP URLs sont pris en charge.	Aucun	http://localhost:7474 https://my-proxy.example.com

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`REQUEST_TIMEOUT`	Cette variable d'environnement contrôle le temps pendant lequel l'agent attend une réponse. AWS AppConfig Si le service ne répond pas, la demande échoue. Si la demande concerne la récupération initiale des données, l'agent renvoie une erreur à votre application. Si le délai d'attente survient lors d'une vérification des données mises à jour en arrière-plan, l'agent enregistre l'erreur et réessaie après un court laps de temps. Vous pouvez spécifier le nombre de millisecondes pour le délai d'expiration. Vous pouvez également spécifier un nombre avec une unité de	3000 ms	3000 3000 ms 5s

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
	temps : ms pour les millisecondes et s pour les secondes. Si aucune unité n'est spécifiée, l'agent utilise par défaut les millisecondes. Par exemple, 5 000, 5 000 ms et 5 s entraînent la même valeur de délai d'expiration de la demande.		
`ROLE_ARN`	Cette variable d'environnement spécifie le nom de ressource Amazon (ARN) d'un rôle IAM. AWS AppConfig L'agent assume ce rôle pour récupérer les données de configuration.	Aucun	arn:aws:iam : :123456789012:role/ MyRole
`ROLE_EXTERNAL_ID`	Cette variable d'environnement indique l'ID externe à utiliser avec l'ARN du rôle assumé.	Aucun	MyExternalId

Variable d'environnement	Détails	Valeur par défaut	Valeur (s) de l'échantillon
`ROLE_SESSION_NAME`	Cette variable d'environnement spécifie le nom de session à associer aux informations d'identification pour le rôle IAM assumé.	Aucun	AWSAppConfigAgentSession
`SERVICE_REGION`	Cette variable d'environnement indique une alternative Région AWS que AWS AppConfig l'agent utilise pour appeler le AWS AppConfig service. Si ce paramètre n'est pas défini, l'agent tente de déterminer la région actuelle. Si ce n'est pas le cas, l'agent ne démarre pas.	Aucun	us-east-1 eu-west-1
`WAIT_ON_MANIFEST`	Cette variable d'environnement configure l' AWS AppConfig agent pour qu'il attende que le manifeste soit traité avant de terminer le démarrage.	true	vrai false

Extraction des données de configuration pour les applications exécutées dans Amazon ECS et Amazon EKS

Vous pouvez récupérer les données de configuration de AWS AppConfig l'agent pour les applications exécutées dans Amazon ECS et Amazon EKS à l'aide d'un appel HTTP localhost. Les exemples suivants sont utilisés `curl` avec un client HTTP. Vous pouvez appeler l'agent à l'aide de n'importe quel client HTTP disponible compatible avec le langage de votre application ou les bibliothèques disponibles.

> **Note**
>
> Pour récupérer les données de configuration si votre application utilise une barre oblique, par exemple « test-backend/test-service », vous devez utiliser le codage URL.

Pour récupérer le contenu complet de toute configuration déployée

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

Pour récupérer un seul drapeau et ses attributs à partir d'une AWS AppConfig configuration de type **`Feature Flag`**

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

Pour accéder à plusieurs drapeaux et à leurs attributs à partir d'une AWS AppConfig configuration de type **`Feature Flag`**

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

L'appel renvoie les métadonnées de configuration dans les en-têtes HTTP, y compris la version de configuration, le type de contenu et l'étiquette de version de configuration (le cas échéant). Le corps de la réponse de l'agent contient le contenu de configuration. Voici un exemple :

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

My test config
```

Récupération des indicateurs de fonctionnalités de base et multivariantes

Pour les configurations d'indicateurs de fonctionnalités (configurations de typeAWS.AppConfig.FeatureFlags), l' AWS AppConfig agent vous permet de récupérer un seul indicateur ou un sous-ensemble d'indicateurs dans une configuration. La récupération d'un ou deux indicateurs est utile si votre cas d'utilisation ne nécessite que quelques indicateurs du profil de configuration. Les exemples suivants utilisent cURL.

> **Note**
>
> La possibilité d'appeler un indicateur de fonctionnalité unique ou un sous-ensemble d'indicateurs dans une configuration n'est disponible que dans les versions 2.0.45 et supérieures de l' AWS AppConfig Agent.

Vous pouvez récupérer les données AWS AppConfig de configuration à partir d'un point de terminaison HTTP local. Pour accéder à un indicateur spécifique ou à une liste d'indicateurs, utilisez le paramètre de `?flag=`*`FLAG_KEY`* requête pour un profil AWS AppConfig de configuration.

Pour récupérer un seul drapeau et ses attributs

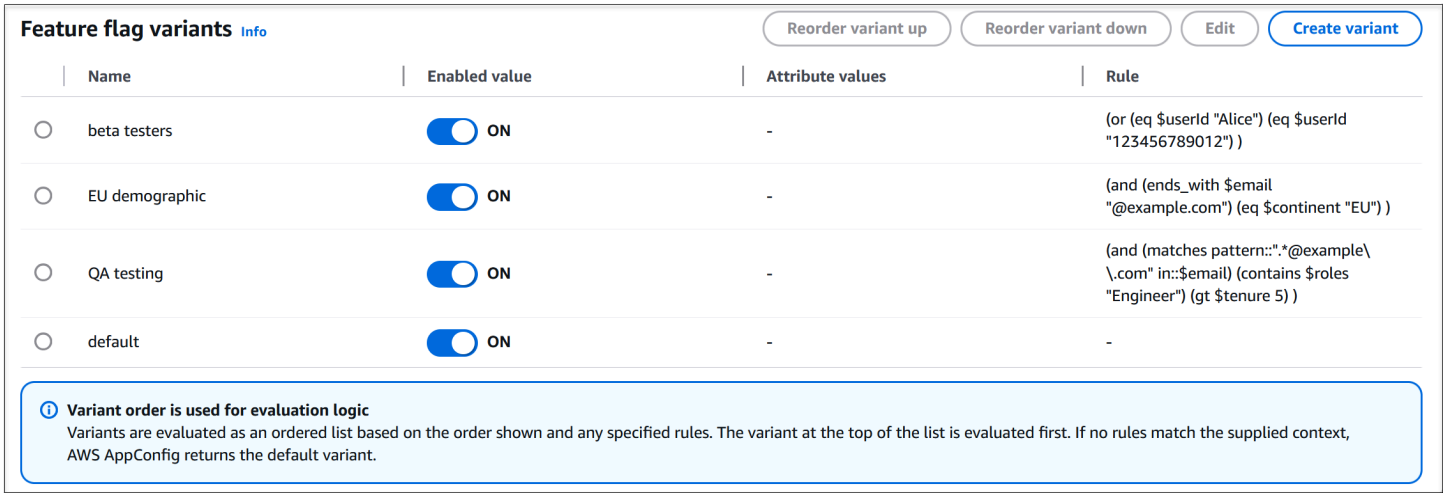
```
curl "http://localhost:2772/applications/APPLICATION_NAME/
environments/ENVIRONMENT_NAME/configurations/CONFIGURATION_NAME?flag=FLAG_KEY"
```

Pour récupérer plusieurs drapeaux et leurs attributs

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

Pour récupérer les variantes des indicateurs de fonctionnalité en fonction du contexte de l'appelant

Les exemples Python suivants montrent comment récupérer des variantes d'indicateurs de fonctionnalité en fonction du contexte de l'appelant. Pour illustrer au mieux la manière de passer ces appels, cette section utilise des exemples d'appels basés sur un scénario dans lequel un client a créé des variantes similaires aux suivantes :



> **Note**
>
> Pour récupérer des variantes d'indicateur, vous devez utiliser la dernière version de AWS AppConfig l'Agent dans votre environnement informatique. Pour plus d'informations, consultez les rubriques suivantes qui décrivent comment mettre à jour, installer ou ajouter l'agent pour chacun des environnements informatiques suivants :
>
> - Pour les environnements de calcul Lambda : Ajout de l'extension AWS AppConfig Agent Lambda
> - Pour les environnements EC2 informatiques Amazon : Étape 2 : (obligatoire) Installation et démarrage de AWS AppConfig l'agent sur les EC2 instances Amazon
> - Pour les environnements de calcul Amazon ECS : Démarrage de l' AWS AppConfig agent pour l'intégration d'Amazon ECS
> - Pour les environnements informatiques Amazon EKS : Démarrage de l' AWS AppConfig agent pour l'intégration d'Amazon EKS

Pour récupérer les données du drapeau en utilisant le contexte de l'appelant jane_doe@example.org (qui n'a pas opté pour le programme bêta) :

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
```

```
-H "Context: email=jane_doe@example.org" \
-H "Context: opted_in_to_beta=false"
{
  "ui_refresh": {"_variant":"QA","dark_mode_support":true,"enabled":true}
}
```

Pour récupérer les données du drapeau en utilisant le contexte de l'appelant jane_doe@example.org (qui a opté pour le programme bêta) :

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

Pour récupérer les données du drapeau en utilisant le contexte de l'appelant jane_doe@qa-testers.example.org (qui est testeur d'assurance qualité chez Example Organization) :

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

Pour récupérer les données du drapeau sans le contexte de l'appelant (qui renvoie la variante par défaut)

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

Pour récupérer des données d'indicateur pour un scénario de répartition du trafic afin de déterminer si 1 appelant aléatoire sur 10 reçoit la variante « population échantillon »

```
for i in {0..9} do ; \
```

```
curl http://localhost:2772/applications/UIRefresh/environments/Production/
configurations/Features \
-H "Context: email=$i@example.org"
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Sample
 Population","dark_mode_support":false,"enabled":true}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
{
  "ui_refresh": {"_variant":"Default Variant","enabled":false}
}
```

Utilisation d'un manifeste pour activer des fonctionnalités de récupération supplémentaires

AWS AppConfig L'agent propose les fonctionnalités supplémentaires suivantes pour vous aider à récupérer les configurations de vos applications.

- Configuration de AWS AppConfig l'agent pour récupérer les configurations de plusieurs comptes: utilisez l' AWS AppConfig agent d'un serveur principal ou d'une source Compte AWS de récupération pour récupérer les données de configuration de plusieurs comptes fournisseurs.
- Configuration de AWS AppConfig l'agent pour écrire des copies de configuration sur disque: utilisez l' AWS AppConfig agent pour écrire les données de configuration sur le disque. Cette fonctionnalité permet aux clients utilisant des applications qui lisent les données de configuration sur disque de s'y intégrer AWS AppConfig.

Comprendre les manifestes des agents

Pour activer ces fonctionnalités de AWS AppConfig l'agent, vous devez créer un manifeste. Un manifeste est un ensemble de données de configuration que vous fournissez pour contrôler les actions que l'agent peut effectuer. Un manifeste est écrit en JSON. Il contient un ensemble de clés de haut niveau correspondant aux différentes configurations que vous avez déployées avec. AWS AppConfig

Un manifeste peut inclure plusieurs configurations. En outre, chaque configuration du manifeste peut identifier une ou plusieurs fonctionnalités d'agent à utiliser pour la configuration spécifiée. Le contenu du manifeste utilise le format suivant :

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

Voici un exemple de code JSON pour un manifeste avec deux configurations. La première configuration (*MyApp*) n'utilise aucune fonctionnalité de AWS AppConfig l'agent. La deuxième

configuration (*My2ndApp*) utilise la copie de configuration écrite sur disque et les fonctionnalités de récupération multi-comptes :

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

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

Comment fournir un manifeste d'agent

Vous pouvez stocker le manifeste sous forme de fichier dans un emplacement où AWS AppConfig l'agent peut le lire. Vous pouvez également enregistrer le manifeste sous forme de AWS AppConfig configuration et pointer l'agent vers celui-ci. Pour fournir un manifeste d'agent, vous devez définir une variable d'`MANIFEST`environnement avec l'une des valeurs suivantes :

Emplacement du manifeste	Valeur de la variable d'environnement	Cas d'utilisation
Fichier	fichier :/path/to/agent-manifest.json	Utilisez cette méthode si votre manifeste ne change pas souvent.
AWS AppConfig configuration	*application-name*:*environment-name*:*configuration-name*	Utilisez cette méthode pour les mises à jour dynamiques. Vous pouvez mettre à jour et déployer un manifeste stocké dans une configuration de la même manière AWS

Emplacement du manifeste	Valeur de la variable d'environnement	Cas d'utilisation
		AppConfig que vous stockez d'autres AWS AppConfig configurations.
Variable d'environnement	Contenu du manifeste (JSON)	Utilisez cette méthode si votre manifeste ne change pas souvent. Cette méthode est utile dans les environnements de conteneurs où il est plus facile de définir une variable d'environnement que d'exposer un fichier.

Pour plus d'informations sur la définition de variables pour l' AWS AppConfig Agent, consultez la rubrique correspondant à votre cas d'utilisation :

- Configuration de l'extension AWS AppConfig Agent Lambda
- Utilisation de AWS AppConfig l'agent avec Amazon EC2
- Utilisation de AWS AppConfig l'agent avec Amazon ECS et Amazon EKS

Configuration de AWS AppConfig l'agent pour récupérer les configurations de plusieurs comptes

Vous pouvez configurer AWS AppConfig l'agent pour récupérer des configurations à partir de plusieurs Comptes AWS en saisissant les remplacements d'informations d'identification dans le manifeste de l' AWS AppConfig agent. Les remplacements d'informations d'identification incluent le nom Amazon Resource (ARN) d'un rôle AWS Identity and Access Management (IAM), un ID de rôle, un nom de session et la durée pendant laquelle l'agent peut assumer le rôle.

Vous entrez ces informations dans une section « informations d'identification » du manifeste. La section « informations d'identification » utilise le format suivant :

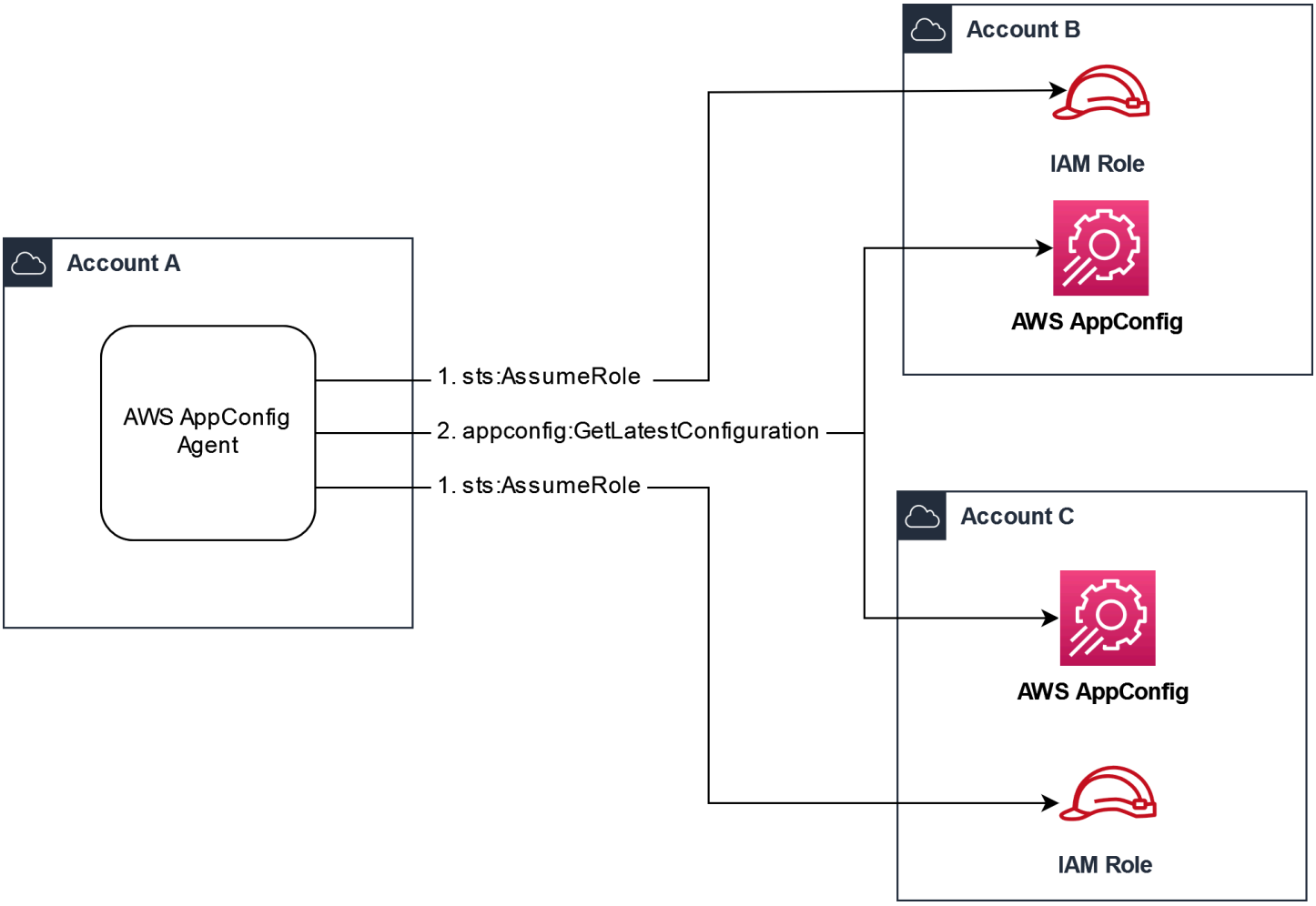
```
{
    "application_name:environment_name:configuration_name": {
```

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

Voici un exemple :

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

Avant de récupérer une configuration, l'agent lit les informations d'identification de la configuration dans le manifeste, puis assume le rôle IAM spécifié pour cette configuration. Vous pouvez spécifier un ensemble différent de remplacements d'informations d'identification pour différentes configurations dans un seul manifeste. Le schéma suivant montre comment l' AWS AppConfig agent, lorsqu'il s'exécute dans le compte A (le compte de récupération), assume les rôles distincts spécifiés pour les comptes B et C (les comptes fournisseurs), puis appelle l'opération GetLatestConfigurationAPI pour récupérer les données de configuration relatives à l' AWS AppConfig exécution sur ces comptes :



Configurer les autorisations pour récupérer les données de configuration des comptes fournisseurs

AWS AppConfig L'agent exécuté dans le compte de récupération doit être autorisé pour récupérer les données de configuration des comptes fournisseurs. Vous donnez l'autorisation à l'agent en créant un rôle AWS Identity and Access Management (IAM) dans chacun des comptes fournisseurs. AWS AppConfig L'agent du compte de récupération assume ce rôle pour obtenir des données à partir des comptes fournisseurs. Suivez les procédures décrites dans cette section pour créer une politique d'autorisations IAM, un rôle IAM et ajouter des remplacements d'agents au manifeste.

Avant de commencer

Collectez les informations suivantes avant de créer une politique d'autorisation et un rôle dans IAM.

- Le IDs pour chacun Compte AWS. Le compte de récupération est le compte qui appellera d'autres comptes pour obtenir des données de configuration. Les comptes fournisseurs sont les comptes qui vendront les données de configuration au compte de récupération.

- Nom du rôle IAM utilisé par le AWS AppConfig compte de récupération. Voici une liste des rôles utilisés par AWS AppConfig défaut par :
  - Pour Amazon Elastic Compute Cloud (Amazon EC2), AWS AppConfig utilise le rôle d'instance.
  - Pour AWS Lambda, AWS AppConfig utilise le rôle d'exécution Lambda.
  - Pour Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon AWS AppConfig EKS), utilise le rôle de conteneur.

  Si vous avez configuré AWS AppConfig l'Agent pour utiliser un rôle IAM différent en spécifiant la variable d'`ROLE_ARN`environnement, notez ce nom.

Création de la politique d'autorisations

Utilisez la procédure suivante pour créer une politique d'autorisations à l'aide de la console IAM. Effectuez la procédure décrite dans chacune d'elles Compte AWS qui vendra les données de configuration pour le compte de récupération.

Pour créer une stratégie IAM

1. Connectez-vous au compte AWS Management Console d'un fournisseur.
2. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
3. Dans le volet de navigation, sélectionnez Politiques, puis Créer une politique.
4. Choisissez l'option JSON.
5. Dans l'éditeur de stratégie, remplacez le JSON par défaut par la déclaration de politique suivante. Mettez à jour chacune *`example resource placeholder`* d'elles avec les détails du compte fournisseur.

   JSON

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

```
            "Resource": "arn:aws:us-
east-1:vendor_account_ID:application/vendor_application_ID/
environment/vendor_environment_ID/configuration/vendor_configuration_ID"
        }
    ]
}
```

Voici un exemple :

JSON

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

6. Choisissez Suivant.
7. Dans le champ Nom de la politique, entrez un nom.
8. (Facultatif) Pour Ajouter des balises, ajoutez une ou plusieurs paires balise-clé-valeur pour organiser, suivre ou contrôler l'accès pour cette politique.
9. Choisissez Create Policy (Créer une politique). Le système vous renvoie à la page Policies (Stratégies).
10. Répétez cette procédure pour chaque appareil Compte AWS destiné à vendre les données de configuration du compte de récupération.

Création du rôle IAM

Utilisez la procédure suivante pour créer un rôle IAM à l'aide de la console IAM. Effectuez la procédure décrite dans chacune d'elles Compte AWS qui vendra les données de configuration pour le compte de récupération.

Pour créer un rôle IAM

1. Connectez-vous au compte AWS Management Console d'un fournisseur.
2. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
3. Dans le volet de navigation, choisissez Rôles, puis Create policy.
4. Pour Trusted entity (Entité de confiance), choisissez Compte AWS.
5. Dans la Compte AWSsection, choisissez Autre Compte AWS.
6. Dans le champ ID de compte, entrez l'ID du compte de récupération.
7. (Facultatif) Pour garantir la sécurité de ce rôle, choisissez Exiger un ID externe et entrez une chaîne.
8. Choisissez Suivant.
9. Sur la page Ajouter des autorisations, utilisez le champ de recherche pour localiser la politique que vous avez créée lors de la procédure précédente. Cochez la case en regard de son nom.
10. Choisissez Suivant.
11. Pour Nom du rôle (Role name), saisissez un nom.
12. (Facultatif) Sous Description, entrez une description.
13. Pour l'étape 1 : Sélectionnez les entités de confiance, choisissez Modifier. Remplacez la politique de confiance JSON par défaut par la politique suivante. Mettez à jour chacune *example resource placeholder* d'elles avec les informations de votre compte de récupération.

    JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::111122223333:role/appconfig_role_in_retrieval_account"
```

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

14. (Facultatif) Pour Tags (Balises), ajoutez une ou plusieurs paires clé-valeur de balise afin d'organiser, de suivre ou de contrôler l'accès pour ce rôle.
15. Sélectionnez Créer un rôle. Le système vous renvoie à la page Rôles.
16. Recherchez le rôle que vous venez de créer. Choisissez-le. Dans la section ARN, copiez l'ARN. Vous allez spécifier ces informations dans la procédure suivante.

Ajouter des remplacements d'informations d'identification au manifeste

Après avoir créé le rôle IAM dans votre compte fournisseur, mettez à jour le manifeste dans le compte de récupération. Plus précisément, ajoutez le bloc d'informations d'identification et l'ARN du rôle IAM pour récupérer les données de configuration du compte fournisseur. Voici le format JSON :

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

Voici un exemple :

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

```
    }
}
```

Vérifiez que la récupération multi-comptes fonctionne

Vous pouvez vérifier que cet agent est capable de récupérer les données de configuration de plusieurs comptes en consultant les journaux de l' AWS AppConfig agent. Le journal des `INFO` niveaux pour les données initiales récupérées pour « `YourApplicationName` `YourEnvironmentName` : : `YourConfigurationName` » est le meilleur indicateur d'une extraction réussie. Si les extractions échouent, vous devriez voir un journal de `ERROR` niveau indiquant la raison de l'échec. Voici un exemple de récupération réussie depuis un compte fournisseur :

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MyTestApplication:MyTestEnvironment:MyDenyListConfiguration' in XX.Xms
```

Configuration de AWS AppConfig l'agent pour écrire des copies de configuration sur disque

Vous pouvez configurer AWS AppConfig l'agent pour stocker automatiquement une copie d'une configuration sur le disque en texte brut. Cette fonctionnalité permet aux clients utilisant des applications qui lisent les données de configuration sur disque de s'y intégrer AWS AppConfig.

Cette fonctionnalité n'est pas conçue pour être utilisée comme fonction de sauvegarde de configuration. AWS AppConfig L'agent ne lit pas les fichiers de configuration copiés sur le disque. Si vous souhaitez sauvegarder des configurations sur disque, consultez les variables d'`PRELOAD_BACKUP`environnement `BACKUP_DIRECTORY` et relatives à l'utilisation de l' AWS AppConfig agent avec Amazon EC2 ou à l'utilisation de l' AWS AppConfig agent avec Amazon ECS et Amazon EKS.

> **Warning**
>
> Notez les informations importantes suivantes concernant cette fonctionnalité :
>
> - Les configurations enregistrées sur disque sont stockées en texte brut et sont lisibles par l'homme. N'activez pas cette fonctionnalité pour les configurations qui incluent des données sensibles.

- Cette fonctionnalité écrit sur le disque local. Utilisez le principe du moindre privilège pour les autorisations du système de fichiers. Pour de plus amples informations, veuillez consulter Implémentation d'un accès sur la base du moindre privilège.

Pour activer l'écriture, copiez la configuration sur disque

1. Modifiez le manifeste.
2. Choisissez la configuration que vous AWS AppConfig souhaitez écrire sur le disque et ajoutez un `writeTo` élément. Voici un exemple :

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

   Voici un exemple :

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

3. Enregistrez vos modifications. Le fichier configuration.json est mis à jour chaque fois que de nouvelles données de configuration sont déployées.

Vérifiez que la copie de configuration d'écriture sur le disque fonctionne

Vous pouvez vérifier que des copies d'une configuration sont écrites sur le disque en consultant les journaux de l' AWS AppConfig agent. L'entrée du `INFO` journal contenant la phrase « INFO a écrit la configuration '*application*: *environment* :*configuration*' dans *file_path* » indique que l' AWS AppConfig agent écrit des copies de configuration sur disque.

Voici un exemple :

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

Génération d'un client à l'aide de la spécification OpenAPI

Vous pouvez utiliser la spécification YAML suivante pour OpenAPI afin de créer un SDK à l'aide d'un outil tel qu'OpenAPI Generator. Vous pouvez mettre à jour cette spécification pour inclure des valeurs codées en dur pour l'application, l'environnement ou la configuration. Vous pouvez également ajouter des chemins supplémentaires (si vous avez plusieurs types de configuration) et inclure des schémas de configuration afin de générer des modèles typés spécifiques à la configuration pour vos clients SDK. Pour plus d'informations sur OpenAPI (également connu sous le nom de Swagger), consultez la spécification OpenAPI.

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

```
          required: true
          schema:
            type: string
        - in: path
          name: Environment
          description: The environment for the configuration to get. Specify either the
environment name or the environment ID.
          required: true
          schema:
            type: string
        - in: path
          name: Configuration
          description: The configuration to get. Specify either the configuration name
or the configuration ID.
          required: true
          schema:
            type: string
        - in: query
          name: flag
          description: The key(s) of the feature flag(s) to retrieve. If not provided,
all flags are returned.
          required: false
          schema:
            type: array
            items:
              type: string
        - in: header
          name: context
          description: Request context used to evaluate multi-variant feature flags.
          required: false
          schema:
            type: array
            items:
              type: string
              pattern: '^\w+=\w+$'
      responses:
        200:
          headers:
            ConfigurationVersion:
              schema:
                type: string
          content:
            application/octet-stream:
              schema:
```

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

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

Utilisation du mode de développement local de l' AWS AppConfig agent

AWS AppConfig L'agent prend en charge un mode de développement local. Si vous activez le mode de développement local, l'agent lit les données de configuration depuis un répertoire spécifique sur le disque. Il ne récupère pas les données de configuration de AWS AppConfig. Vous pouvez simuler des déploiements de configuration en mettant à jour les fichiers dans le répertoire spécifié. Nous recommandons le mode de développement local pour les cas d'utilisation suivants :

- Testez différentes versions de configuration avant de les déployer à l'aide de AWS AppConfig.
- Testez différentes options de configuration pour une nouvelle fonctionnalité avant de valider les modifications dans votre référentiel de code.
- Testez différents scénarios de configuration pour vérifier qu'ils fonctionnent comme prévu.

> **Warning**
>
> N'utilisez pas le mode de développement local dans les environnements de production. Ce mode ne prend pas en charge les fonctionnalités de AWS AppConfig sécurité importantes telles que la validation du déploiement et les annulations automatisées.

Utilisez la procédure suivante pour configurer AWS AppConfig l'agent en mode de développement local.

Pour configurer AWS AppConfig l'agent en mode de développement local

1. Installez l'agent à l'aide de la méthode décrite pour votre environnement informatique. AWS AppConfig L'agent fonctionne avec les éléments suivants Services AWS :
   - AWS Lambda
   - Amazon EC2
   - Amazon ECS et Amazon EKS
2. Si l'agent est en cours d'exécution, arrêtez-le.
3. Ajoutez `LOCAL_DEVELOPMENT_DIRECTORY` à la liste des variables d'environnement. Spécifiez un répertoire sur le système de fichiers qui fournit à l'agent des autorisations de lecture. Par exemple, `/tmp/local_configs`.
4. Créez un fichier dans le répertoire. Le nom du fichier doit utiliser le format suivant :

```
application_name:environment_name:configuration_profile_name
```

Voici un exemple :

```
Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
```

> **Note**
>
> - Pour consulter des exemples d'indicateurs de fonctionnalités que vous pouvez ajouter à un fichier de votre `LOCAL_DEVELOPMENT_DIRECTORY` répertoire, consultezExemples d'indicateurs de fonctionnalités pour le mode de développement local de l' AWS AppConfig agent.
> - (Facultatif) Vous pouvez contrôler le type de contenu renvoyé par l'agent pour vos données de configuration en fonction de l'extension que vous attribuez au fichier. Par exemple, si vous attribuez au fichier une extension .json, l'agent renvoie le type de contenu `application/json` lorsque votre application le demande. Si vous omettez l'extension, l'agent l'utilise `application/octet-stream` pour le type de contenu. Si vous avez besoin d'un contrôle précis, vous pouvez fournir une extension au format `.type%subtype`. L'agent renverra un type de contenu de `type/subtype`.

5. Exécutez la commande suivante pour redémarrer l'agent et demander les données de configuration.

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

L'agent vérifie les modifications apportées au fichier local à l'intervalle d'interrogation spécifié pour l'agent. Si l'intervalle d'interrogation n'est pas spécifié, l'agent utilise l'intervalle par défaut de 45 secondes. Cette vérification effectuée à intervalles d'interrogation garantit que l'agent se comporte de la même manière dans un environnement de développement local que lorsqu'il est configuré pour interagir avec le AWS AppConfig service.

> **Note**
>
> Pour déployer une nouvelle version d'un fichier de configuration de développement local, mettez à jour le fichier avec de nouvelles données.

Exemples d'indicateurs de fonctionnalités pour le mode de développement local de l' AWS AppConfig agent

Cette section inclut des exemples d'indicateurs de fonctionnalités que vous pouvez utiliser avec AWS AppConfig l'Agent en mode développement local. Le mode de développement local attend des données d'indicateur de fonctionnalité au format de récupération des données. Le format au moment de la récupération est le format renvoyé lorsque l'indicateur est extrait de l'GetLatestConfigurationAPI, qui ne contient que la valeur de l'indicateur. Le format de récupération n'inclut pas la définition complète d'un indicateur (telle qu'elle a été transmise à l'CreateHostedConfigurationVersionAPI). La définition complète d'un indicateur contient également des informations telles que les noms et les valeurs des attributs, les contraintes et l'état activé de l'indicateur.

Rubriques

- Exemples de drapeaux relatifs aux fonctionnalités de base
- Exemples de drapeaux de fonctionnalités à variantes multiples

Exemples de drapeaux relatifs aux fonctionnalités de base

Utilisez les exemples d'indicateurs de fonctionnalités de base suivants avec AWS AppConfig l'Agent en mode de développement local.

> **Note**
>
> Si vous souhaitez que l'agent indique le type de contenu de vos données d'indicateur de fonctionnalité locales `application/json` (comme il le ferait lors de la récupération de données d'indicateur AWS AppConfig dans un environnement autre que le mode de développement local), vos fichiers d'indicateurs de fonctionnalités locaux doivent utiliser l'extension .json. Par exemple, `Local:MyFeatureFlags:SampleB1.json`.

Exemple 1 : indicateur unique représentant une actualisation de l'interface utilisateur.

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

Exemple 2 : plusieurs drapeaux représentant des indicateurs de fonctionnalités opérationnelles.

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

Exemples de drapeaux de fonctionnalités à variantes multiples

Le format de récupération d'une configuration d'indicateur de fonctionnalité contenant au moins un indicateur de fonctionnalité à variantes multiples est représenté sous forme de données Amazon Ion au lieu de données JSON. Dans ce format, les indicateurs à variantes multiples sont représentés sous forme de liste annotée, et les indicateurs de base sont représentés sous forme de chaîne annotée. Les éléments de liste d'un indicateur multivariant sont soit un tuple (une liste d'une longueur de deux), qui représente une seule variante, soit une chaîne, qui représente la variante par défaut. Dans un tuple variant, le premier élément est une expression s qui représente la règle du variant, et le second élément est une chaîne qui représente le contenu du variant.

Pour que l'agent interprète correctement ces fichiers, vos fichiers d'indicateurs de fonctionnalités locaux doivent porter l'extension suivante :. `application%ion%type=AWS.AppConfig.FeatureFlags`. Par

exemple, `Local:MyFeatureFlags:SampleMV1.application%ion %type=AWS.AppConfig.FeatureFlags`.

Exemple 1 : indicateur à plusieurs variantes représentant une version échelonnée d'une nouvelle fonctionnalité.

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

Exemple 2 : plusieurs drapeaux représentant différents écrans UX en fonction de l'identifiant de l'utilisateur. Les deux premiers drapeaux sont multivariants et le drapeau final est basique.

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

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

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

AWS AppConfig considérations relatives à l'utilisation du navigateur et des appareils mobiles

Les indicateurs de fonctionnalité vous permettent de mettre à jour l'expérience de vos pages Web et de votre application mobile à la volée, sans les frais généraux, les risques ou la rigidité d'une version de l'App Store. À l'aide des indicateurs de fonctionnalité, vous pouvez apporter progressivement une modification à votre base d'utilisateurs au moment de votre choix. Si vous rencontrez une erreur, vous pouvez annuler instantanément la modification sans obliger les utilisateurs à effectuer une mise à niveau vers une nouvelle version du logiciel. En bref, les indicateurs de fonctionnalité offrent un contrôle et une flexibilité accrus lors du déploiement des modifications apportées à votre application.

Les sections suivantes décrivent les points importants à prendre en compte lors de l'utilisation des indicateurs de AWS AppConfig fonctionnalité sur les pages Web et les appareils mobiles.

Rubriques

Récupération des données de configuration et des drapeaux

Pour les cas d'utilisation du navigateur et du mobile, de nombreux clients choisissent d'utiliser une couche proxy entre le Web ou l'application mobile et AWS AppConfig. Cela permet de AWS AppConfig dissocier votre volume d'appels de la taille de votre base d'utilisateurs, ce qui réduit les coûts. Cela vous permet également de tirer parti de l'AWS AppConfig agent, qui optimise les performances de récupération des indicateurs et prend en charge des fonctionnalités telles que les indicateurs à variantes multiples. AWS AppConfig recommande de l' AWS Lambda utiliser pour créer le proxy. Au lieu de récupérer les indicateurs directement depuis AWS AppConfig, configurez l'extension AWS AppConfig Lambda pour récupérer vos indicateurs de fonctionnalité au sein d'une fonction Lambda. Écrivez la fonction pour accepter les paramètres de AWS AppConfig récupération de la demande d'événement et pour renvoyer les données de configuration correspondantes dans la réponse Lambda. Exposez votre proxy à Internet à l'aide de la fonction Lambda. URLs

Après avoir configuré votre proxy, prenez en compte la fréquence à laquelle vous récupérez les données. Les cas d'utilisation des appareils mobiles ne nécessitent généralement pas d'intervalles d'interrogation à haute fréquence. Configurez AWS AppConfig l'agent pour qu'il actualise les données AWS AppConfig plus fréquemment que votre application ne le fait depuis le proxy.

Authentification et Amazon Cognito

La fonction Lambda URLs prend en charge deux formes de contrôle d'accès, et`AWS_IAM`. `NONE` À utiliser `NONE` si vous préférez implémenter votre propre authentification et autorisation dans votre fonction Lambda. `NONE`est également l'option recommandée si votre cas d'utilisation permet d'exposer votre point de terminaison au public et que vos données de configuration ne contiennent pas de données sensibles. Pour tous les autres cas d'utilisation, utilisez`AWS_IAM`.

> **Important**
>
> Si vous exposez votre terminal à Internet sans authentification, assurez-vous que vos données de configuration ne divulguent pas de données sensibles, notamment des informations d'identification personnelle (PII), des utilisateurs ou des IDs noms de fonctionnalités inédits.

Si vous choisissez de l'utiliser`AWS_IAM`, vous devrez gérer les informations d'identification avec Amazon Cognito. Pour commencer à utiliser Amazon Cognito, vous devez créer un pool d'identités. Un pool d'identités vous permet de vendre des informations d'identification à court terme à votre application pour les utilisateurs authentifiés ou invités. Vous devrez ajouter des rôles dans le pool d'identités qui permettent aux utilisateurs d'utiliser la fonction `InvokeFunctionUrl` for your Lambda. Cela permet aux instances de votre application d'accéder aux informations d'identification nécessaires pour récupérer vos données de configuration.

Lorsque vous utilisez Amazon Cognito dans votre application, pensez à utiliser. AWS Amplify Amplify simplifie les interactions entre les mobile/web applications AWS et fournit un support intégré pour Amazon Cognito.

mise en cache

Lors de l'utilisation AWS AppConfig, vous devez toujours mettre en cache vos données de configuration localement sur l'appareil ou dans le navigateur. La mise en cache présente les avantages suivants :

- Améliore les performances en réduisant la latence et l'épuisement de la batterie
- Assure la stabilité en éliminant les dépendances liées à l'accès au réseau
- Réduit les coûts en réduisant la fréquence de récupération des données

Pour les cas d'utilisation mobiles, nous vous recommandons d'implémenter des caches en mémoire et persistants sur l'appareil. Configurez votre application pour tenter de récupérer la configuration souhaitée depuis le cache en mémoire et revenez à la récupération depuis votre proxy, si nécessaire. Une fois la récupération réussie depuis votre proxy, mettez à jour le cache en mémoire, puis conservez la configuration sur l'appareil. Utilisez un processus en arrière-plan pour parcourir le cache et actualiser chaque configuration. Lorsque vous récupérez la configuration pour la première fois après le démarrage de l'application, si la récupération échoue, reportez-vous à la configuration persistante (et utilisez-la pour amorcer le cache en mémoire).

Segmentation

Lorsque vous utilisez des indicateurs de fonctionnalités, vous souhaiterez peut-être segmenter l'expérience de signalisation des fonctionnalités au sein de votre clientèle. Pour ce faire, fournissez un contexte à vos appels de récupération d'indicateurs et configurez des règles pour renvoyer différentes variantes de vos indicateurs de fonctionnalités en fonction du contexte fourni. Par exemple, vous pouvez avoir une variante d'indicateur de fonctionnalité pour les utilisateurs d'iOS 18.X, une variante pour les utilisateurs d'iOS 17.X et un indicateur par défaut pour toutes les autres versions d'iOS. Avec les variantes, vous pouvez configurer chaque version iOS de votre application pour cibler la même configuration dans le même environnement, mais en fonction du contexte fourni lors de l'appel de récupération (par exemple, « version » : « i OS18 .1 »), les appareils recevront la variante appropriée de la configuration.

> **Note**
>
> Si vous utilisez des variantes d'indicateurs de AWS AppConfig fonctionnalité pour un cas d'utilisation mobile, vous devez utiliser l' AWS AppConfig agent et un proxy pour récupérer les indicateurs de fonctionnalité.

Si vous choisissez de ne pas utiliser l' AWS AppConfig Agent pour récupérer les indicateurs de fonctionnalités, vous pouvez tirer parti AWS AppConfig des environnements pour une segmentation simple et à faible cardinalité. Un environnement est un groupe de déploiement logique pour vos cibles. Outre le partitionnement de vos configurations en environnements de développement, de test

et de production, vous pouvez subdiviser votre clientèle en créant des environnements spécifiques aux mobiles, tels que le type d'appareil (tablette ou téléphone) ou les versions majeures du système d'exploitation. Avec des environnements distincts, vous pouvez déployer des ensembles de données de configuration identiques ou différents pour répondre aux exigences particulières de votre clientèle.

Bande passante (cas d'utilisation mobile)

En général, essayez de réduire la taille de chaque ensemble de drapeaux. Les cas d'utilisation mobiles ont tendance à impliquer des contraintes de faible bande passante. La réduction de la taille de vos données vous aidera à maintenir une expérience cohérente au sein de votre base d'utilisateurs. Sachez également que, dans la mesure où les appareils mobiles fonctionnent souvent entre des environnements à faible bande passante et à bande passante nulle, la mise en cache sur l'appareil est essentielle. Un code d'application qui échoue correctement si aucune donnée de configuration ne peut être récupérée est également essentiel.

Autres cas d'utilisation des drapeaux

Le pouvoir des indicateurs de fonctionnalités ne se limite pas à la commodité de publication des fonctionnalités. Des indicateurs opérationnels de longue date peuvent être utilisés pour améliorer la posture opérationnelle de votre application. Par exemple, vous pouvez créer un bouton de surveillance des performances qui émet des métriques supplémentaires et des données de débogage lors d'un événement. Vous pouvez également souhaiter maintenir et ajuster les taux de rafraîchissement de vos applications pour un segment de votre clientèle.

Récupération des données de configuration sans AWS AppConfig agent

La méthode recommandée pour récupérer les données de configuration AWS AppConfig consiste à utiliser l'agent développé et géré AWS AppConfig par Amazon. Avec l'agent, vous pouvez mettre en cache les données de configuration localement et interroger le service de plan de AWS AppConfig données de manière asynchrone pour obtenir des mises à jour. Ce caching/polling processus garantit que vos données de configuration sont toujours disponibles pour votre application tout en minimisant le temps de latence et les coûts. Si vous préférez ne pas utiliser l'agent, vous pouvez appeler le public APIs directement depuis le service de plan de AWS AppConfig données.

Le service de plan de données utilise deux actions d'API, StartConfigurationSessionet GetLatestConfiguration. Le service de plan de données utilise également des points de terminaison distincts de ceux du plan de AWS AppConfig contrôle.

> **Note**
>
> Le service de plan de données remplace le processus précédent de récupération des données de configuration à l'aide de l'action `GetConfiguration` API. L'`GetConfiguration`API est obsolète.

Comment ça marche

Voici comment fonctionne le processus d'appel direct à AWS AppConfig APIs l'aide du service de plan de données.

Votre application récupère les données de configuration en établissant d'abord une session de configuration à l'aide de l'opération StartConfigurationSessionAPI. Le client de votre session passe ensuite des appels périodiques GetLatestConfigurationpour vérifier et récupérer les dernières données disponibles.

Lorsque vous appelez`StartConfigurationSession`, votre code envoie les informations suivantes :

- Identifiants (ID ou nom) d'une AWS AppConfig application, d'un environnement et d'un profil de configuration suivis par la session.
- (Facultatif) Durée minimale pendant laquelle le client de la session doit attendre entre les appels à`GetLatestConfiguration`.

En réponse, AWS AppConfig fournit un `InitialConfigurationToken` à donner au client de la session et à utiliser la première fois qu'il appelle `GetLatestConfiguration` cette session.

> **Important**
>
> Ce jeton ne doit être utilisé qu'une seule fois lors de votre premier appel à`GetLatestConfiguration`. Vous devez utiliser le nouveau jeton dans la `GetLatestConfiguration` réponse (`NextPollConfigurationToken`) lors de chaque appel suivant à`GetLatestConfiguration`. Pour prendre en charge les longs cas d'utilisation des sondages, les jetons sont valables jusqu'à 24 heures. Si un `GetLatestConfiguration` appel utilise un jeton expiré, le système revient`BadRequestException`.

Lorsque vous appelez`GetLatestConfiguration`, votre code client envoie la `ConfigurationToken` valeur la plus récente qu'il possède et reçoit en réponse :

- `NextPollConfigurationToken`: `ConfigurationToken` valeur à utiliser lors du prochain appel à`GetLatestConfiguration`.
- `NextPollIntervalInSeconds`: la durée pendant laquelle le client doit attendre avant de passer son prochain appel`GetLatestConfiguration`.
- La configuration : les dernières données destinées à la session. Ce champ peut être vide si le client dispose déjà de la dernière version de la configuration.

> **Important**
>
> Notez les informations importantes suivantes.
>
> - L'StartConfigurationSessionAPI ne doit être appelée qu'une seule fois par application, environnement, profil de configuration et client pour établir une session avec le service. Cela se fait généralement au démarrage de votre application ou juste avant la première récupération d'une configuration.
> - Si votre configuration est déployée à l'aide d'un`KmsKeyIdentifier`, votre demande de réception de la configuration doit inclure l'autorisation d'appel`kms:Decrypt`. Pour plus d'informations, consultez Déchiffrer dans la référence de l'AWS Key Management Service API.
> - L'opération d'API précédemment utilisée pour récupérer les données de `GetConfiguration` configuration est obsolète. Le fonctionnement de `GetConfiguration` l'API ne prend pas en charge les configurations chiffrées.

(Exemple) Récupération d'une configuration en appelant AWS AppConfig APIs

L' AWS CLI exemple suivant montre comment récupérer des données de configuration à l'aide des opérations AWS AppConfig Data `StartConfigurationSession` et `GetLatestConfiguration` API. La première commande lance une session de configuration. Cet appel inclut le IDs (ou les noms) de l' AWS AppConfig application, de l'environnement et du profil de configuration. L'API renvoie un fichier `InitialConfigurationToken` utilisé pour récupérer vos données de configuration.

```
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
```

Le système répond avec les informations au format suivant.

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

Après avoir démarré une session, utilisez InitialConfigurationTokento call GetLatestConfigurationpour récupérer vos données de configuration. Les données de configuration sont enregistrées dans le `mydata.json` fichier.

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

Le premier `GetLatestConfiguration` appel à utiliser le format `ConfigurationToken` obtenu à partir de`StartConfigurationSession`. Les informations suivantes sont renvoyées.

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

Les appels suivants `GetLatestConfiguration` doivent fournir des informations `NextPollConfigurationToken` issues de la réponse précédente.

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

> **Important**
>
> Notez les informations importantes suivantes concernant le fonctionnement de `GetLatestConfiguration` l'API :

- La `GetLatestConfiguration` réponse inclut une `Configuration` section qui présente les données de configuration. La `Configuration` section apparaît uniquement si le système trouve des données de configuration nouvelles ou mises à jour. Si le système ne trouve pas de données de configuration nouvelles ou mises à jour, les `Configuration` données sont vides.
- Vous recevez un nouveau message `ConfigurationToken` à chaque réponse de`GetLatestConfiguration`.
- Nous vous recommandons de régler la fréquence d'interrogation de vos appels d'API `GetLatestConfiguration` en fonction de votre budget, de la fréquence prévue de vos déploiements de configuration et du nombre de cibles pour une configuration.

Extension des AWS AppConfig workflows à l'aide d'extensions

Une extension augmente votre capacité à injecter de la logique ou du comportement à différents moments du AWS AppConfig flux de travail de création ou de déploiement d'une configuration. Par exemple, vous pouvez utiliser des extensions pour effectuer les types de tâches suivants (pour n'en citer que quelques-unes) :

- Envoyez une notification à une rubrique Amazon Simple Notification Service (Amazon SNS) lorsqu'un profil de configuration est déployé.
- Nettoyez le contenu d'un profil de configuration pour détecter les données sensibles avant le début du déploiement.
- Créez ou mettez à jour un problème Atlassian Jira chaque fois qu'une modification est apportée à un indicateur de fonctionnalité.
- Fusionnez le contenu d'un service ou d'une source de données dans vos données de configuration lorsque vous démarrez un déploiement.
- Sauvegardez une configuration dans un compartiment Amazon Simple Storage Service (Amazon S3) chaque fois qu'une configuration est déployée.

Vous pouvez associer ces types de tâches à des AWS AppConfig applications, à des environnements et à des profils de configuration.

Table des matières

Comprendre les AWS AppConfig extensions

Cette rubrique présente les concepts et la terminologie des AWS AppConfig extensions. Les informations sont abordées dans le contexte de chaque étape requise pour configurer et utiliser les AWS AppConfig extensions.

Rubriques

Étape 1 : Déterminez ce que vous voulez faire avec les extensions

Souhaitez-vous recevoir une notification à un webhook qui envoie des messages à Slack chaque fois qu'un AWS AppConfig déploiement est terminé ? Voulez-vous sauvegarder un profil de configuration dans un compartiment Amazon Simple Storage Service (Amazon S3) avant qu'une configuration ne soit déployée ? Voulez-vous nettoyer les données de configuration pour y trouver des informations sensibles avant que la configuration ne soit déployée ? Vous pouvez utiliser des extensions pour effectuer ce type de tâches et bien plus encore. Vous pouvez créer des extensions personnalisées ou utiliser les AWS extensions créées incluses dans. AWS AppConfig

> **Note**
>
> Dans la plupart des cas d'utilisation, pour créer une extension personnalisée, vous devez créer une AWS Lambda fonction pour effectuer les calculs et les traitements définis dans l'extension. Pour de plus amples informations, veuillez consulter Procédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Les extensions AWS créées ci-dessous peuvent vous aider à intégrer rapidement les déploiements de configuration à d'autres services. Vous pouvez utiliser ces extensions dans la AWS AppConfig console ou en appelant des actions d'API d'extension directement depuis le AWS CLI SDK ou le SDK. Outils AWS pour PowerShell

Extension	Description
Amazon CloudWatch teste évidemment A/B	Cette extension permet à votre application d'attribuer des variations aux sessions utilisateur localement plutôt qu'en appelant l'EvaluateFeatureopération. Pour de plus amples informations, veuillez consulter Utilisation de l'extension Amazon CloudWatch Evidently.

Extension	Description
AWS AppConfig événements de déploiement vers EventBridge	Cette extension envoie des événements au bus d'événements EventBridge par défaut lorsqu'une configuration est déployée.
AWS AppConfig événements de déploiement sur Amazon Simple Notification Service (Amazon SNS)	Cette extension envoie des messages à une rubrique Amazon SNS que vous spécifiez lors du déploiement d'une configuration.
AWS AppConfig événements de déploiement sur Amazon Simple Queue Service (Amazon SQS)	Cette extension place les messages dans votre file d'attente Amazon SQS lorsqu'une configuration est déployée.
Extension d'intégration — Atlassian Jira	Cette extension permet AWS AppConfig de créer et de mettre à jour des problèmes chaque fois que vous modifiez un indicateur de fonctionnalité.

Étape 2 : déterminez à quel moment vous souhaitez que l'extension s'exécute

Une extension définit une ou plusieurs actions qu'elle exécute au cours d'un AWS AppConfig flux de travail. Par exemple, l'`AWS AppConfig deployment events to Amazon SNS`extension AWS créée inclut une action permettant d'envoyer une notification à une rubrique Amazon SNS. Chaque action est invoquée soit lorsque vous interagissez avec, AWS AppConfig AWS AppConfig soit lorsque vous exécutez un processus en votre nom. C'est ce que l'on appelle des points d'action. AWS AppConfig les extensions prennent en charge les points d'action suivants :

Points d'action PRE_* : les actions d'extension configurées sur les points `PRE_*` d'action sont appliquées après la validation de la demande, mais avant AWS AppConfig d'exécuter l'activité correspondant au nom du point d'action. Ces appels d'action sont traités en même temps qu'une demande. Si plusieurs demandes sont effectuées, les appels d'action s'exécutent de manière séquentielle. Notez également que les points `PRE_*` d'action reçoivent et peuvent modifier le contenu d'une configuration. `PRE_*`les points d'action peuvent également répondre à une erreur et empêcher une action de se produire.

- PRE_CREATE_HOSTED_CONFIGURATION_VERSION
- PRE_START_DEPLOYMENT

Points d'action ON_* : une extension peut également s'exécuter en parallèle avec un AWS AppConfig flux de travail en utilisant un point d'ON_*action. ON_*les points d'action sont invoqués de manière asynchrone. ON_*les points d'action ne reçoivent pas le contenu d'une configuration. Si une extension rencontre une erreur pendant un point ON_* d'action, le service ignore l'erreur et poursuit le flux de travail.

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

Points d'action AT_* : les actions d'extension configurées sur les points AT_* d'action sont invoquées de manière synchrone et en parallèle à un flux de travail. AWS AppConfig Si une extension rencontre une erreur pendant un point AT_* d'action, le service arrête le flux de travail et annule le déploiement.

- AT_DEPLOYMENT_TICK

Étape 3 : créer une association d'extensions

Pour créer une extension ou configurer une extension AWS créée par un auteur, vous définissez les points d'action qui invoquent une extension lorsqu'une AWS AppConfig ressource spécifique est utilisée. Par exemple, vous pouvez choisir d'exécuter l'AWS AppConfig deployment events to Amazon SNSextension et de recevoir des notifications sur une rubrique Amazon SNS chaque fois qu'un déploiement de configuration est lancé pour une application spécifique. La définition des points d'action invoquant une extension pour une AWS AppConfig ressource spécifique s'appelle une association d'extension. Une association d'extension est une relation spécifiée entre une extension et une AWS AppConfig ressource, telle qu'une application ou un profil de configuration.

Une seule AWS AppConfig application peut inclure plusieurs environnements et profils de configuration. Si vous associez une extension à une application ou à un environnement, AWS

AppConfig invoque l'extension pour tous les flux de travail liés à l'application ou aux ressources de l'environnement, le cas échéant.

Supposons, par exemple, que vous ayez une AWS AppConfig application appelée MobileApps qui inclut un profil de configuration appelé AccessList. Supposons que l' MobileApps application inclut des environnements bêta, d'intégration et de production. Vous créez une association d'extension pour l'extension AWS de notification Amazon SNS créée et vous associez l'extension à MobileApps l'application. L'extension de notification Amazon SNS est invoquée chaque fois que la configuration est déployée pour l'application dans l'un des trois environnements.

> **Note**
>
> Il n'est pas nécessaire de créer une extension pour AWS utiliser des extensions créées, mais vous devez créer une association d'extensions.

Étape 4 : Déployer une configuration et vérifier que les actions d'extension sont effectuées

Après avoir créé une association, lorsqu'une configuration hébergée est créée ou qu'une configuration est déployée AWS AppConfig , appelle l'extension et exécute les actions spécifiées. Lorsqu'une extension est invoquée, si le système rencontre une erreur lors d'un point `PRE-*` d'action, AWS AppConfig renvoie des informations sur cette erreur.

Utilisation d' AWS extensions créées

AWS AppConfig inclut les extensions AWS créées suivantes. Ces extensions peuvent vous aider à intégrer le AWS AppConfig flux de travail à d'autres services. Vous pouvez utiliser ces extensions dans AWS Management Console ou en appelant des actions d'API d'extension directement depuis le AWS CLI Outils AWS pour PowerShell, ou le SDK.

Extension	Description
Amazon CloudWatch teste évidemment A/B	Cette extension permet à votre application d'attribuer des variations aux sessions utilisateur localement plutôt qu'en appelant l'EvaluateFeatureopération. Pour de plus amples informati

Extension	Description
	ons, veuillez consulter Utilisation de l'extension Amazon CloudWatch Evidently.
AWS AppConfig événements de déploiement vers EventBridge	Cette extension envoie des événements au bus d'événements EventBridge par défaut lorsqu'une configuration est déployée.
AWS AppConfig événements de déploiement sur Amazon Simple Notification Service (Amazon SNS)	Cette extension envoie des messages à une rubrique Amazon SNS que vous spécifiez lors du déploiement d'une configuration.
AWS AppConfig événements de déploiement sur Amazon Simple Queue Service (Amazon SQS)	Cette extension place les messages dans votre file d'attente Amazon SQS lorsqu'une configuration est déployée.
Extension d'intégration — Atlassian Jira	Cette extension permet AWS AppConfig de créer et de mettre à jour des problèmes chaque fois que vous modifiez un indicateur de fonctionnalité.

Utilisation de l'extension Amazon CloudWatch Evidently

Vous pouvez utiliser Amazon CloudWatch Evidently pour valider de nouvelles fonctionnalités en toute sécurité en les proposant à un pourcentage spécifique de vos utilisateurs pendant que vous déployez la fonctionnalité. Vous pouvez surveiller les performances de la nouvelle fonction afin de décider du moment où vous souhaitez augmenter le trafic vers vos utilisateurs. Ainsi, vous pouvez réduire les risques et identifier les conséquences involontaires avant de lancer pleinement la fonction. Vous pouvez également effectuer des A/B expériences pour prendre des décisions de conception des fonctionnalités sur la base de preuves et de données.

L' AWS AppConfig extension pour CloudWatch Evidently permet à votre application d'attribuer des variations aux sessions utilisateur localement plutôt qu'en appelant l'EvaluateFeatureopération. Une session locale atténue les risques de latence et de disponibilité associés à un appel d'API. Pour plus d'informations sur la configuration et l'utilisation de l'extension, consultez la section Effectuer des lancements et A/B des expériences avec CloudWatch Evidently dans le guide de l' CloudWatch utilisateur Amazon.

Utilisation des événements AWS AppConfig de déploiement vers l' EventBridge extension Amazon

L'`AWS AppConfig deployment events to Amazon EventBridge`extension est une extension AWS créée qui vous aide à surveiller et à agir sur le flux de travail de déploiement AWS AppConfig de la configuration. L'extension envoie des notifications d'événements au bus d'événements EventBridge par défaut chaque fois qu'une configuration est déployée. Une fois que vous avez associé l'extension à l'une de vos AWS AppConfig applications, environnements ou profils de configuration, AWS AppConfig envoie des notifications d'événements au bus d'événements après le début, la fin et l'annulation de chaque déploiement de configuration.

Si vous souhaitez mieux contrôler les points d'action qui envoient EventBridge des notifications, vous pouvez créer une extension personnalisée et saisir le bus d'événements EventBridge par défaut Amazon Resource Name (ARN) pour le champ URI. Pour plus d'informations sur la création d'une extension, consultezProcédure pas à pas : création d'extensions personnalisées AWS AppConfig.

> **Important**
>
> Cette extension prend uniquement en charge le bus d'événements EventBridge par défaut.

Utilisation de l'extension

Pour utiliser l'`AWS AppConfig deployment events to Amazon EventBridge`extension, vous devez d'abord l'associer à l'une de vos AWS AppConfig ressources en créant une association d'extensions. Vous créez l'association à l'aide de la AWS AppConfig console ou de l'action CreateExtensionAssociationAPI. Lorsque vous créez l'association, vous spécifiez l'ARN d'une AWS AppConfig application, d'un environnement ou d'un profil de configuration. Si vous associez l'extension à une application ou à un environnement, une notification d'événement est envoyée pour tout profil de configuration contenu dans l'application ou l'environnement spécifié.

Après avoir créé l'association, lorsqu'une configuration pour la AWS AppConfig ressource spécifiée est déployée, AWS AppConfig appelle l'extension et envoie des notifications en fonction des points d'action spécifiés dans l'extension.

> **Note**
>
> Cette extension est invoquée par les points d'action suivants :

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

Vous ne pouvez pas personnaliser les points d'action pour cette extension. Pour invoquer différents points d'action, vous pouvez créer votre propre extension. Pour de plus amples informations, veuillez consulter Procédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Utilisez les procédures suivantes pour créer une association d' AWS AppConfig extension à l'aide de la AWS Systems Manager console ou du AWS CLI.

Pour créer une association d'extensions (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez AWS AppConfig.
3. Dans l'onglet Extensions, choisissez Ajouter à la ressource.
4. Dans la section Détails de la ressource d'extension, pour Type de ressource, choisissez un type de AWS AppConfig ressource. En fonction de la ressource que vous choisissez, vous AWS AppConfig invite à choisir d'autres ressources.
5. Choisissez Créer une association à la ressource.

Voici un exemple d'événement envoyé EventBridge lorsque l'extension est invoquée.


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

```
    ],
    "detail":{
       "InvocationId":"5tfjcig",
        "Parameters":{

       },
       "Type":"OnDeploymentComplete",
       "Application":{
          "Id":"ba8toh7",
          "Name":"MyApp"
       },
       "Environment":{
          "Id":"pgil2o7",
          "Name":"MyEnv"
       },
       "ConfigurationProfile":{
          "Id":"ga3tqep",
          "Name":"MyConfigProfile"
       },
       "DeploymentNumber":1,
       "ConfigurationVersion":"1"
    }
}
```

Utilisation des événements AWS AppConfig de déploiement vers l'extension Amazon SNS

L'`AWS AppConfig deployment events to Amazon SNS`extension est une extension AWS créée qui vous aide à surveiller et à agir sur le flux de travail de déploiement AWS AppConfig de la configuration. L'extension publie des messages sur une rubrique Amazon SNS chaque fois qu'une configuration est déployée. Une fois que vous avez associé l'extension à l'une de vos AWS AppConfig applications, environnements ou profils de configuration, AWS AppConfig publiez un message dans le sujet après le début, la fin et l'annulation de chaque déploiement de configuration.

Si vous souhaitez mieux contrôler les points d'action qui envoient des notifications Amazon SNS, vous pouvez créer une extension personnalisée et saisir une rubrique Amazon SNS (Amazon Resource Name (ARN) pour le champ URI. Pour plus d'informations sur la création d'une extension, consultezProcédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Utilisation de l'extension

Cette section décrit comment utiliser l'`AWS AppConfig deployment events to Amazon SNS`extension.

Étape 1 : configurer AWS AppConfig pour publier des messages dans un sujet

Ajoutez une politique de contrôle d'accès à votre rubrique Amazon SNS en accordant AWS AppConfig (`appconfig.amazonaws.com`) des autorisations de publication (`sns:Publish`). Pour plus d'informations, consultez Exemples de cas relatifs au contrôle d'accès Amazon SNS.

Étape 2 : créer une association d'extensions

Associez l'extension à l'une de vos AWS AppConfig ressources en créant une association d'extensions. Vous créez l'association à l'aide de la AWS AppConfig console ou de l'action CreateExtensionAssociationAPI. Lorsque vous créez l'association, vous spécifiez l'ARN d'une AWS AppConfig application, d'un environnement ou d'un profil de configuration. Si vous associez l'extension à une application ou à un environnement, une notification est envoyée pour tout profil de configuration contenu dans l'application ou l'environnement spécifié. Lorsque vous créez l'association, vous devez saisir une valeur pour le `topicArn` paramètre qui contient l'ARN de la rubrique Amazon SNS que vous souhaitez utiliser.

Après avoir créé l'association, lorsqu'une configuration pour la AWS AppConfig ressource spécifiée est déployée, AWS AppConfig appelle l'extension et envoie des notifications en fonction des points d'action spécifiés dans l'extension.

> **Note**
>
> Cette extension est invoquée par les points d'action suivants :
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Vous ne pouvez pas personnaliser les points d'action pour cette extension. Pour invoquer différents points d'action, vous pouvez créer votre propre extension. Pour de plus amples informations, veuillez consulter Procédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Utilisez les procédures suivantes pour créer une association d' AWS AppConfig extension à l'aide de la AWS Systems Manager console ou du AWS CLI.

Pour créer une association d'extensions (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez AWS AppConfig.
3. Dans l'onglet Extensions, choisissez Ajouter à la ressource.
4. Dans la section Détails de la ressource d'extension, pour Type de ressource, choisissez un type de AWS AppConfig ressource. En fonction de la ressource que vous choisissez, vous AWS AppConfig invite à choisir d'autres ressources.
5. Choisissez Créer une association à la ressource.

Voici un exemple du message envoyé à la rubrique Amazon SNS lorsque l'extension est invoquée.

```
{
    "Type": "Notification",
    "MessageId": "ae9d702f-9a66-51b3-8586-2b17932a9f28",
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic",
    "Message": {
        "InvocationId": "7itcaxp",
        "Parameters": {
            "topicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic"
        },
        "Application": {
            "Id": "1a2b3c4d",
            "Name": MyApp
        },
        "Environment": {
            "Id": "1a2b3c4d",
            "Name": MyEnv
        },
        "ConfigurationProfile": {
            "Id": "1a2b3c4d",
            "Name": "MyConfigProfile"
        },
        "Description": null,
        "DeploymentNumber": "3",
        "ConfigurationVersion": "1",
```

```
        "Type": "OnDeploymentComplete"
    },
    "Timestamp": "2022-06-30T20:26:52.067Z",
    "SignatureVersion": "1",
    "Signature": "<...>",
    "SigningCertURL": "<...>",
    "UnsubscribeURL": "<...>",
    "MessageAttributes": {
        "MessageType": {
            "Type": "String",
            "Value": "OnDeploymentStart"
        }
    }
}
```

Utilisation des événements AWS AppConfig de déploiement vers l'extension Amazon SQS

L'`AWS AppConfig deployment events to Amazon SQS`extension est une extension AWS créée qui vous aide à surveiller et à agir sur le flux de travail de déploiement AWS AppConfig de la configuration. L'extension place les messages en file d'attente dans votre file d'attente Amazon Simple Queue Service (Amazon SQS) chaque fois qu'une configuration est déployée. Après avoir associé l'extension à l'une de vos AWS AppConfig applications, environnements ou profils de configuration, AWS AppConfig place un message dans la file d'attente après chaque début, fin et annulation de chaque déploiement de configuration.

Si vous souhaitez mieux contrôler les points d'action qui envoient des notifications Amazon SQS, vous pouvez créer une extension personnalisée et saisir une file d'attente Amazon SQS (ARN) pour le champ URI. Pour plus d'informations sur la création d'une extension, consultezProcédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Utilisation de l'extension

Cette section décrit comment utiliser l'`AWS AppConfig deployment events to Amazon SQS`extension.

Étape 1 : Configuration AWS AppConfig pour mettre les messages en file d'attente

Ajoutez une politique Amazon SQS à votre file d'attente Amazon SQS en AWS AppConfig accordant `appconfig.amazonaws.com` () les autorisations d'envoi de message (). `sqs:SendMessage` Pour plus d'informations, consultez Exemples de base des politiques Amazon SQS.

Étape 2 : créer une association d'extensions

Associez l'extension à l'une de vos AWS AppConfig ressources en créant une association d'extensions. Vous créez l'association à l'aide de la AWS AppConfig console ou de l'action CreateExtensionAssociationAPI. Lorsque vous créez l'association, vous spécifiez l'ARN d'une AWS AppConfig application, d'un environnement ou d'un profil de configuration. Si vous associez l'extension à une application ou à un environnement, une notification est envoyée pour tout profil de configuration contenu dans l'application ou l'environnement spécifié. Lorsque vous créez l'association, vous devez entrer un `Here` paramètre contenant l'ARN de la file d'attente Amazon SQS que vous souhaitez utiliser.

Après avoir créé l'association, lorsqu'une configuration pour la AWS AppConfig ressource spécifiée est créée ou déployée, AWS AppConfig appelle l'extension et envoie des notifications en fonction des points d'action spécifiés dans l'extension.

> **Note**
>
> Cette extension est invoquée par les points d'action suivants :
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Vous ne pouvez pas personnaliser les points d'action pour cette extension. Pour invoquer différents points d'action, vous pouvez créer votre propre extension. Pour de plus amples informations, veuillez consulter Procédure pas à pas : création d'extensions personnalisées AWS AppConfig.

Utilisez les procédures suivantes pour créer une association d' AWS AppConfig extension à l'aide de la AWS Systems Manager console ou du AWS CLI.

Pour créer une association d'extensions (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez AWS AppConfig.
3. Dans l'onglet Extensions, choisissez Ajouter à la ressource.

4. Dans la section Détails de la ressource d'extension, pour Type de ressource, choisissez un type de AWS AppConfig ressource. En fonction de la ressource que vous choisissez, vous AWS AppConfig invite à choisir d'autres ressources.
5. Choisissez Créer une association à la ressource.

Voici un exemple du message envoyé à la file d'attente Amazon SQS lorsque l'extension est invoquée.

```
{
   "InvocationId":"7itcaxp",
   "Parameters":{
      "queueArn":"arn:aws:sqs:us-east-1:111122223333:MySQSQueue"
   },
   "Application":{
      "Id":"1a2b3c4d",
      "Name":MyApp
   },
   "Environment":{
      "Id":"1a2b3c4d",
      "Name":MyEnv
   },
   "ConfigurationProfile":{
      "Id":"1a2b3c4d",
      "Name":"MyConfigProfile"
   },
   "Description":null,
   "DeploymentNumber":"3",
   "ConfigurationVersion":"1",
   "Type":"OnDeploymentComplete"
}
```

Utilisation de l'extension Atlassian Jira pour AWS AppConfig

Grâce à l'intégration à Atlassian Jira, vous AWS AppConfig pouvez créer et mettre à jour des problèmes dans la console Atlassian chaque fois que vous modifiez un indicateur de fonctionnalité dans votre formulaire spécifié. Compte AWS Région AWS Chaque problème de Jira inclut le nom du drapeau, l'ID de l'application, l'ID du profil de configuration et les valeurs du drapeau. Une fois que vous avez mis à jour, enregistré et déployé vos modifications d'indicateur, Jira met à jour les problèmes existants avec les détails de la modification.

> **Note**
>
> Jira met à jour les problèmes chaque fois que vous créez ou mettez à jour un indicateur de fonctionnalité. Jira corrige également les problèmes lorsque vous supprimez un attribut de drapeau de niveau enfant d'un indicateur de niveau parent. Jira n'enregistre aucune information lorsque vous supprimez un drapeau au niveau du parent.

Pour configurer l'intégration, vous devez effectuer les opérations suivantes :

- Configuration des autorisations pour l'intégration AWS AppConfig à Jira
- Configuration de l'application d'intégration AWS AppConfig Jira

Configuration des autorisations pour l'intégration AWS AppConfig à Jira

Lorsque vous configurez AWS AppConfig l'intégration avec Jira, vous spécifiez les informations d'identification d'un utilisateur. Plus précisément, vous entrez l'ID de la clé d'accès et la clé secrète de l'utilisateur dans l'application AWS AppConfig pour Jira. Cet utilisateur autorise Jira à communiquer avec AWS AppConfig. AWS AppConfig utilise ces informations d'identification une fois pour établir une association entre AWS AppConfig et Jira. Les informations d'identification ne sont pas stockées. Vous pouvez supprimer l'association en désinstallant l'application AWS AppConfig for Jira.

Le compte utilisateur nécessite une politique d'autorisation qui inclut les actions suivantes :

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

Effectuez les tâches suivantes pour créer une politique d'autorisation IAM et un utilisateur pour l'intégration AWS AppConfig de Jira :

Tâches

Tâche 1 : créer une politique d'autorisation IAM pour l'intégration AWS AppConfig de Jira

Utilisez la procédure suivante pour créer une politique d'autorisation IAM qui autorise Atlassian Jira à communiquer avec. AWS AppConfig Nous vous recommandons de créer une nouvelle politique et de l'associer à un nouveau rôle IAM. L'ajout de l'autorisation requise à une politique et à un rôle IAM existants va à l'encontre du principe du moindre privilège et n'est pas recommandé.

Pour créer une politique IAM pour l'intégration AWS AppConfig de Jira

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Politiques, puis Créer une politique.
3. Sur la page Créer une politique, choisissez l'onglet JSON et remplacez le contenu par défaut par la politique suivante. Dans la politique suivante, remplacez, *Region* *account_IDapplication_ID*, et *configuration_profile_ID* par les informations provenant de votre environnement AWS AppConfig d'indicateurs de fonctionnalités.

   JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:ListApplications"
            ],
            "Resource": [
                "arn:aws:appconfig:us-east-1:account_ID:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:ListConfigurationProfiles"
            ],
            "Resource": [
                "arn:aws:appconfig:us-
east-1:account_ID:application/application_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
            "Resource": "*"
        }
    ]
}
```

4. Sélectionnez Suivant : Étiquettes.
5. (Facultatif) Ajoutez une ou plusieurs paires clé-valeur de balise afin d'organiser, de suivre ou de contrôler l'accès pour ce rôle, puis sélectionnez Next : Review (Suivant : Vérifier).
6. Dans la page Review policy (Vérification de la politique), saisissez un nom dans la zone Name (Nom), tel que **`AppConfigJiraPolicy`**, puis saisissez une description facultative.
7. Choisissez Create Policy (Créer une politique).

Tâche 2 : créer un utilisateur pour AWS AppConfig et intégrer Jira

Utilisez la procédure suivante pour créer un utilisateur pour l'intégration AWS AppConfig d'Atlassian Jira. Après avoir créé l'utilisateur, vous pouvez copier l'ID de la clé d'accès et la clé secrète, que vous spécifierez une fois l'intégration terminée.

Pour créer un utilisateur AWS AppConfig et intégrer Jira

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, sélectionnez Users (Utilisateurs), puis Add users (Ajouter des utilisateurs).
3. Dans le champ Nom d'utilisateur, entrez un nom, tel que**`AppConfigJiraUser`**.
4. Pour Sélectionner le type AWS d'identifiant, choisissez Clé d'accès - Accès par programmation.
5. Choisissez Suivant : Autorisations.
6. Sur la page Définir les autorisations, choisissez Joindre directement les politiques existantes. Recherchez et cochez la case correspondant à la politique que vous avez créée dansTâche 1 : créer une politique d'autorisation IAM pour l'intégration AWS AppConfig de Jira, puis choisissez Next : Tags.
7. Sur la page Ajouter des balises (facultatif), ajoutez une ou plusieurs paires balise-clé-valeur pour organiser, suivre ou contrôler l'accès de cet utilisateur. Choisissez Suivant : Vérification.
8. Sur la page de révision, vérifiez les informations de l'utilisateur.
9. Choisissez Create user (Créer un utilisateur). Le système affiche l'identifiant de la clé d'accès et la clé secrète de l'utilisateur. Téléchargez le fichier .csv ou copiez ces informations d'identification dans un autre emplacement. Vous devez spécifier ces informations d'identification lors de la configuration de l'intégration.

Configuration de l'application d'intégration AWS AppConfig Jira

Utilisez la procédure suivante pour configurer les options requises dans l'application AWS AppConfig for Jira. Une fois cette procédure terminée, Jira crée un nouveau problème pour chaque indicateur de fonctionnalité indiqué Région AWS dans votre Compte AWS formulaire. Si vous modifiez un indicateur de fonctionnalité dans AWS AppConfig, Jira enregistre les détails dans les problèmes existants.

> **Note**
>
> Un indicateur de AWS AppConfig fonctionnalité peut inclure plusieurs attributs de drapeau au niveau de l'enfant. Jira crée un problème pour chaque indicateur de fonctionnalité au niveau du parent. Si vous modifiez un attribut de drapeau au niveau de l'enfant, vous pouvez consulter les détails de cette modification dans le numéro de Jira pour le drapeau au niveau des parents.

Pour configurer l'intégration

1. Connectez-vous à l'Atlassian Marketplace.
2. Tapez **`AWS AppConfig`** dans le champ de recherche et appuyez sur Entrée.
3. Installez l'application sur votre instance Jira.
4. Dans la console Atlassian, choisissez Gérer les applications, puis choisissez Jira AWS AppConfig .
5. Choisissez Configurer.
6. Sous Détails de configuration, choisissez le projet Jira, puis le projet que vous souhaitez associer à votre indicateur de AWS AppConfig fonctionnalité.
7. Choisissez Région AWS, puis choisissez la région dans laquelle se trouve votre drapeau de AWS AppConfig fonctionnalité.
8. Dans le champ ID de l'application, entrez le nom de l' AWS AppConfig application qui contient votre indicateur de fonctionnalité.
9. Dans le champ ID du profil de configuration, entrez le nom du profil de AWS AppConfig configuration pour votre indicateur de fonctionnalité.
10. Dans les champs ID de clé d'accès et clé secrète, entrez les informations d'identification que vous avez copiéesTâche 2 : créer un utilisateur pour AWS AppConfig et intégrer Jira. Facultativement, vous pouvez également spécifier un jeton de session.
11. Sélectionnez Envoyer.
12. Dans la console Atlassian, choisissez Projects, puis choisissez le projet que vous avez sélectionné pour AWS AppConfig l'intégration. La page Problèmes affiche un problème pour chaque indicateur de fonctionnalité dans les valeurs spécifiées Compte AWS et Région AWS.

Suppression de l'application et des données AWS AppConfig for Jira

Si vous ne souhaitez plus utiliser l'intégration de Jira avec les indicateurs de AWS AppConfig fonctionnalité, vous pouvez supprimer l'application AWS AppConfig for Jira dans la console Atlassian. La suppression de l'application d'intégration permet d'effectuer les opérations suivantes :

- Supprime l'association entre votre instance Jira et AWS AppConfig
- Supprime les détails de votre instance Jira de AWS AppConfig

Pour supprimer l'application AWS AppConfig for Jira

1. Dans la console Atlassian, choisissez Gérer les applications.
2. Choisissez AWS AppConfig Jira.
3. Choisissez Désinstaller.

Procédure pas à pas : création d'extensions personnalisées AWS AppConfig

Pour créer une AWS AppConfig extension personnalisée, effectuez les tâches suivantes. Chaque tâche est décrite plus en détail dans les rubriques suivantes.

> **Note**
>
> Vous pouvez consulter des exemples d' AWS AppConfig extensions personnalisées sur GitHub :
>
> - Exemple d'extension qui empêche les déploiements avec un calendrier de `blocked day` moratoire à l'aide de Systems Manager Change Calendar
> - Exemple d'extension qui empêche les secrets de s'infiltrer dans les données de configuration à l'aide de git-secrets
> - Exemple d'extension qui empêche la fuite d'informations personnelles (PII) dans les données de configuration à l'aide d'Amazon Comprehend

1. Création d'une AWS Lambda fonction

   Dans la plupart des cas d'utilisation, pour créer une extension personnalisée, vous devez créer une AWS Lambda fonction pour effectuer les calculs et les traitements définis dans l'extension. Il existe une exception à cette règle si vous créez des versions personnalisées des extensions de notification AWS créées pour ajouter ou supprimer des points d'action. Pour plus de détails sur cette exception, consultezÉtape 3 : créer une AWS AppConfig extension personnalisée.

2. Configurez les autorisations pour votre extension personnalisée

   Pour configurer les autorisations pour votre extension personnalisée, vous pouvez effectuer l'une des opérations suivantes :

- Créez un rôle de service AWS Identity and Access Management (IAM) incluant des `InvokeFunction` autorisations.
- Créez une politique de ressources à l'aide de l'action d'AddPermissionAPI Lambda.

Cette procédure pas à pas décrit comment créer le rôle de service IAM.

3. Création d'une extension

   Vous pouvez créer une extension à l'aide de la AWS AppConfig console ou en appelant l'action d'CreateExtensionAPI depuis le AWS CLI SDK ou depuis le SDK. Outils AWS pour PowerShell La procédure pas à pas utilise la console.

4. Création d'une association d'extensions

   Vous pouvez créer une association d'extension à l'aide de la AWS AppConfig console ou en appelant l'action d'CreateExtensionAssociationAPI depuis AWS CLI le Outils AWS pour PowerShell SDK ou depuis le SDK. La procédure pas à pas utilise la console.

5. Effectuez une action qui invoque l'extension

   Après avoir créé l'association, AWS AppConfig invoque l'extension lorsque les points d'action définis par l'extension se produisent pour cette ressource. Par exemple, si vous associez une extension contenant une `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` action, l'extension est invoquée chaque fois que vous créez une nouvelle version de configuration hébergée.

Les rubriques de cette section décrivent chaque tâche impliquée dans la création d'une AWS AppConfig extension personnalisée. Chaque tâche est décrite dans le contexte d'un cas d'utilisation où un client souhaite créer une extension qui sauvegarde automatiquement une configuration dans un compartiment Amazon Simple Storage Service (Amazon S3). L'extension s'exécute chaque fois qu'une configuration hébergée est créée (`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`) ou déployée (`PRE_START_DEPLOYMENT`).

Rubriques

- Étape 1 : Création d'une fonction Lambda pour une extension personnalisée AWS AppConfig
- Étape 2 : configurer les autorisations pour une AWS AppConfig extension personnalisée
- Étape 3 : créer une AWS AppConfig extension personnalisée
- Étape 4 : Création d'une association d'extension pour une AWS AppConfig extension personnalisée

Étape 1 : Création d'une fonction Lambda pour une extension personnalisée AWS AppConfig

Dans la plupart des cas d'utilisation, pour créer une extension personnalisée, vous devez créer une AWS Lambda fonction pour effectuer les calculs et les traitements définis dans l'extension. Cette section inclut un exemple de code de fonction Lambda pour une extension personnalisée AWS AppConfig . Cette section inclut également les détails de référence des demandes de charge utile et des réponses. Pour plus d'informations sur la création d'une fonction Lambda, voir Getting started with Lambda dans le manuel du développeur.AWS Lambda

Exemple de code

L'exemple de code suivant pour une fonction Lambda, lorsqu'elle est invoquée, sauvegarde automatiquement une AWS AppConfig configuration dans un compartiment Amazon S3. La configuration est sauvegardée chaque fois qu'une nouvelle configuration est créée ou déployée. L'exemple utilise des paramètres d'extension afin que le nom du compartiment n'ait pas besoin d'être codé en dur dans la fonction Lambda. En utilisant les paramètres d'extension, l'utilisateur peut associer l'extension à plusieurs applications et sauvegarder les configurations dans différents buckets. L'exemple de code inclut des commentaires pour expliquer plus en détail la fonction.

Exemple de fonction Lambda pour une extension AWS AppConfig

```
from datetime import datetime
import base64
import json

import boto3


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

    # Extensions that use the PRE_CREATE_HOSTED_CONFIGURATION_VERSION and
 PRE_START_DEPLOYMENT
    # action points receive the contents of AWS AppConfig configurations in Lambda
 event parameters.
    # Configuration contents are received as a base64-encoded string, which the lambda
 needs to decode
    # in order to get the configuration data as bytes. For other action points, the
 content
    # of the configuration isn't present, so the code below will fail.
```

```
    config_data_bytes = base64.b64decode(event["Content"])

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

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

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

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


def write_backup_to_s3(bucket_name, config_data_bytes):
    s3 = boto3.resource('s3')
    new_object = s3.Object(bucket_name,
 f"config_backup_{datetime.now().isoformat()}.txt")
```

```
    new_object.put(Body=config_data_bytes)
```

Si vous souhaitez utiliser cet exemple au cours de cette procédure pas à pas, enregistrez-le sous le nom **`MyS3ConfigurationBackUpExtension`** et copiez l'Amazon Resource Name (ARN) de la fonction. Vous spécifiez l'ARN lorsque vous créez le rôle AWS Identity and Access Management (IAM) assume dans la section suivante. Vous spécifiez l'ARN et le nom lorsque vous créez l'extension.

Référence de charge utile

Cette section inclut les informations de référence relatives aux demandes de charge utile et aux réponses pour travailler avec des AWS AppConfig extensions personnalisées.

Structure de la demande

AtDeploymentTick

```
{
    'InvocationId': 'o2xbtm7',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'Type': 'OnDeploymentStart',
    'Application': {
        'Id': 'abcd123'
    },
    'Environment': {
        'Id': 'efgh456'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'ConfigurationVersion': '2',
    'DeploymentState': 'DEPLOYING',
    'PercentageComplete': '0.0'
}
```

Structure de la demande

PreCreateHostedConfigurationVersion

```
{
    'InvocationId': 'vlns753', // id for specific invocation
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh', // Base64 encoded content
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'Description': '',
    'Type': 'PreCreateHostedConfigurationVersion',
    'PreviousContent': {
        'ContentType': 'text/plain',
        'ContentVersion': '1',
        'Content': 'SGVsbG8gd29ybGQh'
    }
}
```

PreStartDeployment

```
{
    'InvocationId': '765ahdm',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh',
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
```

```
    'Environment': {
        'Id': 'ibpnqlq',
        'Name': 'EnvironmentName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'Type': 'PreStartDeployment'
}
```

Événements asynchrones

OnStartDeployment, OnDeploymentStep, OnDeployment

```
{
    'InvocationId': 'o2xbtm7',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'Type': 'OnDeploymentStart',
    'Application': {
        'Id': 'abcd123'
    },
    'Environment': {
        'Id': 'efgh456'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'ConfigurationVersion': '2'
}
```

Structure de réponse

Les exemples suivants montrent ce que renvoie votre fonction Lambda en réponse à la demande d'une extension personnalisée AWS AppConfig .

PRE_* Evénements synchrones - réponse réussie

Si vous souhaitez transformer le contenu, utilisez ce qui suit :

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

AT_* Événements synchrones - réponse réussie

Si vous souhaitez contrôler les étapes suivantes d'un déploiement (poursuivre un déploiement ou l'annuler), définissez `Directive` et `Description` attributs dans la réponse.

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

`Directive`prend en charge deux valeurs : `CONTINUE` ou`ROLL_BACK`. Utilisez ces énumérations dans votre réponse à la charge utile pour contrôler les prochaines étapes d'un déploiement.

Événements synchrones - réponse réussie

Si vous souhaitez transformer le contenu, utilisez ce qui suit :

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

Si vous ne souhaitez pas transformer le contenu, ne renvoyez rien.

Événements asynchrones - réponse réussie

Ne rien retourner.

Tous les événements d'erreur

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

```
        }]
    }
```

Étape 2 : configurer les autorisations pour une AWS AppConfig extension personnalisée

Utilisez la procédure suivante pour créer et configurer un rôle de service AWS Identity and Access Management (IAM) (ou assumer un rôle). AWS AppConfig utilise ce rôle pour appeler la fonction Lambda.

Pour créer un rôle de service IAM et autoriser AWS AppConfig à l'assumer

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, sélectionnez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'entité de confiance, choisissez Politique de confiance personnalisée.
4. Collez la politique JSON suivante dans le champ Politique de confiance personnalisée.

   JSON

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

   Choisissez Suivant.
5. Sur la page Ajouter des autorisations, choisissez Créer une politique. La page Créer une stratégie s'ouvre dans un nouvel onglet.
6. Choisissez l'onglet JSON, puis collez la politique d'autorisation suivante dans l'éditeur. L'`lambda:InvokeFunction`action est utilisée pour les points `PRE_*` d'action.

L'`lambda:InvokeAsync`action est utilisée pour les points `ON_*` d'action. *`Your Lambda ARN`*Remplacez-le par le Amazon Resource Name (ARN) de votre Lambda.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "Your Lambda ARN"
    }
  ]
}
```

7. Choisissez Suivant : Balises.
8. Sur la page Ajouter des balises (facultatif), ajoutez une ou plusieurs paires clé-valeur, puis choisissez Suivant : Réviser.
9. Sur la page Révision de la politique, entrez un nom et une description, puis choisissez Créer une politique.
10. Dans l'onglet du navigateur correspondant à votre politique de confiance personnalisée, cliquez sur l'icône Actualiser, puis recherchez la politique d'autorisation que vous venez de créer.
11. Cochez la case correspondant à votre politique d'autorisation, puis choisissez Suivant.
12. Sur la page Nom, révision et création, entrez un nom dans le champ Nom du rôle, puis entrez une description.
13. Sélectionnez Créer un rôle. Le système vous renvoie à la page Rôles. Choisissez Afficher le rôle dans la bannière.
14. Copiez l'ARN. Vous spécifiez cet ARN lorsque vous créez l'extension.

Étape 3 : créer une AWS AppConfig extension personnalisée

Une extension définit une ou plusieurs actions qu'elle exécute au cours d'un AWS AppConfig flux de travail. Par exemple, l'`AWS AppConfig deployment events to Amazon SNS`extension AWS créée inclut une action permettant d'envoyer une notification à une rubrique Amazon SNS. Chaque action est invoquée soit lorsque vous interagissez avec, AWS AppConfig AWS AppConfig soit lorsque vous exécutez un processus en votre nom. C'est ce que l'on appelle des points d'action. AWS AppConfig les extensions prennent en charge les points d'action suivants :

Points d'action PRE_* : les actions d'extension configurées sur les points `PRE_*` d'action sont appliquées après la validation de la demande, mais avant AWS AppConfig d'exécuter l'activité correspondant au nom du point d'action. Ces appels d'action sont traités en même temps qu'une demande. Si plusieurs demandes sont effectuées, les appels d'action s'exécutent de manière séquentielle. Notez également que les points `PRE_*` d'action reçoivent et peuvent modifier le contenu d'une configuration. `PRE_*`les points d'action peuvent également répondre à une erreur et empêcher une action de se produire.

- `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`
- `PRE_START_DEPLOYMENT`

Points d'action ON_* : une extension peut également s'exécuter en parallèle avec un AWS AppConfig flux de travail en utilisant un point d'`ON_*`action. `ON_*`les points d'action sont invoqués de manière asynchrone. `ON_*`les points d'action ne reçoivent pas le contenu d'une configuration. Si une extension rencontre une erreur pendant un point `ON_*` d'action, le service ignore l'erreur et poursuit le flux de travail.

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

Points d'action AT_* : les actions d'extension configurées sur les points `AT_*` d'action sont invoquées de manière synchrone et en parallèle à un flux de travail. AWS AppConfig Si une extension rencontre une erreur pendant un point `AT_*` d'action, le service arrête le flux de travail et annule le déploiement.

- `AT_DEPLOYMENT_TICK`

> **Note**
>
> Le point `AT_DEPLOYMENT_TICK` d'action prend en charge l'intégration de la surveillance par des tiers. `AT_DEPLOYMENT_TICK`est invoqué lors de l'orchestration du traitement du déploiement de la configuration. Si vous utilisez une solution de surveillance tierce (Datadog, par exemple), vous pouvez créer une AWS AppConfig extension qui vérifie la présence d'alarmes au point `AT_DEPLOYMENT_TICK` d'action et, à titre de garde-fou, annule le déploiement s'il déclenche une alarme. Pour consulter un exemple de code d'une AWS AppConfig extension qui utilise le point `AT_DEPLOYMENT_TICK` d'action pour s'intégrer à Datadog, consultez aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub

Exemple d'extension

L'exemple d'extension suivant définit une action qui appelle le point `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` d'action. Sur le `Uri` terrain, l'action spécifie le nom de ressource Amazon (ARN) de la fonction `MyS3ConfigurationBackUpExtension` Lambda créée précédemment dans cette procédure pas à pas. L'action spécifie également l'ARN AWS Identity and Access Management (IAM) assume le rôle créé précédemment dans cette procédure pas à pas.

Exemple d' AWS AppConfig extension

```
{
    "Name": "MySampleExtension",
    "Description": "A sample extension that backs up configurations to an S3 bucket.",
    "Actions": {
        "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
            {
                "Name": "PreCreateHostedConfigVersionActionForS3Backup",
                "Uri": "arn:aws:lambda:aws-
region:111122223333:function:MyS3ConfigurationBackUpExtension",
                "RoleArn": "arn:aws:iam::111122223333:role/ExtensionsTestRole"
            }
        ]
    },
    "Parameters" : {
        "S3_BUCKET": {
            "Required": false
        }
```

```
    }
}
```

> **Note**
>
> Pour consulter la syntaxe des demandes et les descriptions des champs lors de la création d'une extension, consultez la CreateExtensionrubrique du Guide de référence des AWS AppConfig API.

Pour créer une extension (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez AWS AppConfig.
3. Dans l'onglet Extensions, choisissez Créer une extension.
4. Pour Nom de l'extension, entrez un nom unique. Pour les besoins de cette procédure pas à pas, entrez**`MyS3ConfigurationBackUpExtension`**. Entrez éventuellement une description.
5. Dans la section Actions, choisissez Ajouter une nouvelle action.
6. Pour Nom de l'action, entrez un nom unique. Pour les besoins de cette procédure pas à pas, entrez**`PreCreateHostedConfigVersionActionForS3Backup`**. Ce nom décrit le point d'action utilisé par l'action et le but de l'extension.
7. Dans la liste des points d'action, choisissez PRE_CREATE_HOSTED_CONFIGURATION_VERSION.
8. Pour Uri, choisissez fonction Lambda, puis choisissez la fonction dans la liste des fonctions Lambda. Si vous ne voyez pas votre fonction, vérifiez que vous vous trouvez bien à l' Région AWS endroit où vous l'avez créée.
9. Pour le rôle IAM, choisissez le rôle que vous avez créé plus tôt dans cette procédure pas à pas.
10. Dans la section Paramètres d'extension (facultatif), choisissez Ajouter un nouveau paramètre.
11. Pour Nom du paramètre, entrez un nom. Pour les besoins de cette procédure pas à pas, entrez**`S3_BUCKET`**.
12. Répétez les étapes 5 à 11 pour créer une deuxième action pour le point PRE_START_DEPLOYMENT d'action.
13. Choisissez Créer une extension.

Personnalisation des extensions AWS de notification créées

Il n'est pas nécessaire de créer un Lambda ou une extension pour utiliser des extensions de notification AWS créées par des auteurs. Vous pouvez simplement créer une association d'extension, puis effectuer une opération qui appelle l'un des points d'action pris en charge. Par défaut, les extensions de AWS notification créées prennent en charge les points d'action suivants :

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

Si vous créez des versions personnalisées de l'`AWS AppConfig deployment events to Amazon SNS`extension et des `AWS AppConfig deployment events to Amazon SQS` extensions, vous pouvez spécifier les points d'action pour lesquels vous souhaitez recevoir des notifications.

> **Note**
>
> L'`AWS AppConfig deployment events to EventBridge`extension ne prend pas en charge les points `PRE_*` d'action. Vous pouvez créer une version personnalisée si vous souhaitez supprimer certains des points d'action par défaut attribués à la AWS version créée.

Il n'est pas nécessaire de créer une fonction Lambda si vous créez des versions personnalisées des extensions de notification AWS créées. Il vous suffit de spécifier un Amazon Resource Name (ARN) dans le `Uri` champ correspondant à la nouvelle version de l'extension.

- Pour une extension de EventBridge notification personnalisée, entrez l'ARN des événements EventBridge par défaut dans le `Uri` champ.
- Pour une extension de notification Amazon SNS personnalisée, entrez l'ARN d'une rubrique Amazon SNS dans le champ. `Uri`
- Pour une extension de notification Amazon SQS personnalisée, entrez l'ARN d'une file de messages Amazon SQS dans le champ. `Uri`

Étape 4 : Création d'une association d'extension pour une AWS AppConfig extension personnalisée

Pour créer une extension ou configurer une extension AWS créée par un auteur, vous définissez les points d'action qui invoquent une extension lorsqu'une AWS AppConfig ressource spécifique est utilisée. Par exemple, vous pouvez choisir d'exécuter l'`AWS AppConfig deployment events to Amazon SNS`extension et de recevoir des notifications sur une rubrique Amazon SNS chaque fois qu'un déploiement de configuration est lancé pour une application spécifique. La définition des points d'action invoquant une extension pour une AWS AppConfig ressource spécifique s'appelle une association d'extension. Une association d'extension est une relation spécifiée entre une extension et une AWS AppConfig ressource, telle qu'une application ou un profil de configuration.

Une seule AWS AppConfig application peut inclure plusieurs environnements et profils de configuration. Si vous associez une extension à une application ou à un environnement, AWS AppConfig invoque l'extension pour tous les flux de travail liés à l'application ou aux ressources de l'environnement, le cas échéant.

Supposons, par exemple, que vous ayez une AWS AppConfig application appelée MobileApps qui inclut un profil de configuration appelé AccessList. Supposons que l' MobileApps application inclut des environnements bêta, d'intégration et de production. Vous créez une association d'extension pour l'extension AWS de notification Amazon SNS créée et vous associez l'extension à MobileApps l'application. L'extension de notification Amazon SNS est invoquée chaque fois que la configuration est déployée pour l'application dans l'un des trois environnements.

Utilisez les procédures suivantes pour créer une association d' AWS AppConfig extension à l'aide de la AWS AppConfig console.

Pour créer une association d'extensions (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez AWS AppConfig.
3. Dans l'onglet Extensions, choisissez un bouton d'option pour une extension, puis choisissez Ajouter à la ressource. Pour les besoins de cette procédure pas à pas, choisissez ConfigurationBackUpExtensionMyS3.
4. Dans la section Détails de la ressource d'extension, pour Type de ressource, choisissez un type de AWS AppConfig ressource. En fonction de la ressource que vous choisissez, vous AWS

AppConfig invite à choisir d'autres ressources. Pour les besoins de cette procédure pas à pas, choisissez Application.

5. Choisissez une application dans la liste.
6. Dans la section Paramètres, vérifiez que S3_BUCKET est répertorié dans le champ Clé. Dans le champ Valeur, collez l'ARN des extensions Lambda. Par exemple : `arn:aws:lambda:aws-region:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Choisissez Créer une association à la ressource.

Après avoir créé l'association, vous pouvez invoquer l'`MyS3ConfigurationBackUpExtension`extension en créant un nouveau profil de configuration qui `hosted` la spécifie`SourceUri`. Dans le cadre du flux de travail visant à créer la nouvelle configuration, AWS AppConfig rencontre le point `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` d'action. La rencontre de ce point d'action appelle l'`MyS3ConfigurationBackUpExtension`extension, qui sauvegarde automatiquement la configuration nouvellement créée dans le compartiment S3 spécifié dans la `Parameter` section de l'association d'extensions.

Utilisation d'exemples de code pour effectuer des AWS AppConfig tâches courantes

Cette section inclut des exemples de code permettant d'exécuter des actions courantes AWS AppConfig par programmation. Nous vous recommandons d'utiliser ces exemples avec Java, Python et JavaScript SDKs d'effectuer les actions dans un environnement de test. Cette section inclut un exemple de code pour nettoyer votre environnement de test une fois que vous avez terminé.

Rubriques

Création ou mise à jour d'une configuration de forme libre stockée dans le magasin de configuration hébergé

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code. Les exemples présentés dans cette section appellent ce qui suit APIs :

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion

Java

```
public CreateHostedConfigurationVersionResponse createHostedConfigVersion() {
```

```
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));

        return hcv;
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a hosted, freeform configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
```

```
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
    Content=b'my config data',
    ContentType='text/plain')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a hosted, freeform configuration profile
const profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: profile.Id,
    ContentType: "text/plain",
    Content: "my config data",
  })
);
```

Création d'un profil de configuration pour un secret stocké dans Secrets Manager

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code. Les exemples présentés dans cette section appellent ce qui suit APIs :

- CreateApplication
- CreateConfigurationProfile

Java

```
private void createSecretsManagerConfigProfile() {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a configuration profile for Secrets Manager Secret
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("secretsmanager://MySecret")
            .retrievalRoleArn("arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret")
            .type("AWS.Freeform"));
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a configuration profile for Secrets Manager Secret
config_profile = appconfig.create_configuration_profile(
```

```
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='secretsmanager://MySecret',
    RetrievalRoleArn='arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret',
    Type='AWS.Freeform')
```

JavaScript

```
import {
  AppConfigClient,
  CreateConfigurationProfileCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a configuration profile for Secrets Manager Secret
await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "secretsmanager://MySecret",
    RetrievalRoleArn: "arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret",
    Type: "AWS.Freeform",
  })
);
```

Déploiement d'un profil de configuration

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code. Les exemples présentés dans cette section appellent ce qui suit APIs :

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion

- CreateEnvironment
- StartDeployment
- GetDeployment

Java

```
private void createDeployment() throws InterruptedException {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));


        // Create an environment
        CreateEnvironmentResponse env = appconfig.createEnvironment(req -> req
            .applicationId(app.id())
            .name("Beta")
            // If you have CloudWatch alarms that monitor the health of your
 service, you can add them here and they
            // will trigger a rollback if they fire during an appconfig deployment
            //.monitors(Monitor.builder().alarmArn("arn:aws:cloudwatch:us-
east-1:520900602629:alarm:MyAlarm")
            //
  .alarmRoleArn("arn:aws:iam::520900602629:role/MyAppConfigAlarmRole").build())
        );
```

```
        // Start a deployment
        StartDeploymentResponse deploymentResponse = appconfig.startDeployment(req -
> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .environmentId(env.id())
            .configurationVersion(hcv.versionNumber().toString())
            .deploymentStrategyId("AppConfig.Linear50PercentEvery30Seconds")
        );

        // Wait for deployment to complete
        List<DeploymentState> nonFinalDeploymentStates = Arrays.asList(
            DeploymentState.DEPLOYING,
            DeploymentState.BAKING,
            DeploymentState.ROLLING_BACK,
            DeploymentState.VALIDATING);
        GetDeploymentRequest getDeploymentRequest =
 GetDeploymentRequest.builder().applicationId(app.id())

 .environmentId(env.id())

 .deploymentNumber(deploymentResponse.deploymentNumber()).build();
        GetDeploymentResponse deployment =
 appconfig.getDeployment(getDeploymentRequest);
        while (nonFinalDeploymentStates.contains(deployment.state())) {
            System.out.println("Waiting for deployment to complete: " + deployment);
            Thread.sleep(1000L);
            deployment = appconfig.getDeployment(getDeploymentRequest);
        }

        System.out.println("Deployment complete: " + deployment);
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')
```

```
# create an environment
environment = appconfig.create_environment(
    ApplicationId=application['Id'],
    Name='MyEnvironment')

# create a configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
    Content=b'my config data',
    ContentType='text/plain')

# start a deployment
deployment = appconfig.start_deployment(
    ApplicationId=application['Id'],
    EnvironmentId=environment['Id'],
    ConfigurationProfileId=config_profile['Id'],
    ConfigurationVersion=str(hcv['VersionNumber']),
    DeploymentStrategyId='AppConfig.Linear20PercentEvery6Minutes')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
  CreateEnvironmentCommand,
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
  StartDeploymentCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
```

```
);

// create an environment
const environment = await appconfig.send(
  new CreateEnvironmentCommand({
    ApplicationId: application.Id,
    Name: "MyEnvironment",
  })
);

// create a configuration profile
const config_profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
const hcv = await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: config_profile.Id,
    Content: "my config data",
    ContentType: "text/plain",
  })
);

// start a deployment
await appconfig.send(
  new StartDeploymentCommand({
    ApplicationId: application.Id,
    EnvironmentId: environment.Id,
    ConfigurationProfileId: config_profile.Id,
    ConfigurationVersion: hcv.VersionNumber.toString(),
    DeploymentStrategyId: "AppConfig.Linear20PercentEvery6Minutes",
  })
);
```

Utilisation de l' AWS AppConfig agent pour lire un profil de configuration de forme libre

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code.

Java

```
public void retrieveConfigFromAgent() throws Exception {
        /*
        In this sample, we will retrieve configuration data from the AWS AppConfig
 Agent.
        The agent is a sidecar process that handles retrieving configuration data
 from AppConfig
        for you in a way that implements best practices like configuration caching.

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

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

Python

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

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

import requests

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

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

JavaScript

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

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

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

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

Utilisation de AWS AppConfig l'agent pour lire un indicateur de fonctionnalité spécifique

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code.

Java

```
public void retrieveSingleFlagFromAgent() throws Exception {
        /*
          You can retrieve a single flag's data from the agent by providing the
 "flag" query string parameter.
          Note: the configuration's type must be AWS.AppConfig.FeatureFlags
        */

        URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyFlagsProfile?flag=myFlagKey");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        StringBuilder content;
        try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
            content = new StringBuilder();
            int ch;
            while ((ch = in.read()) != -1) {
                content.append((char) ch);
            }
        }
        con.disconnect();
        System.out.println("MyFlagName from agent: " + content);
    }
```

Python

```
import requests

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

# retrieve a single flag's data by providing the "flag" query string parameter
# note: the configuration's type must be AWS.AppConfig.FeatureFlags
```

```
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}?
flag={flag_key}")
config = response.content
```

JavaScript

```
const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";
const flag_name = "MyFlag";

// retrieve a single flag's data by providing the "flag" query string parameter
// note: the configuration's type must be AWS.AppConfig.FeatureFlags
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}?flag=${flag_name}`;
const response = await fetch(url);
const flag = await response.json(); // { "enabled": true/false }
```

Utilisation de AWS AppConfig l'agent pour récupérer un indicateur de fonctionnalité avec des variantes

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code.

Java

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

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

    // Make a GET request to the agent's local server to retrieve the configuration
 data
    URL url = new URL("http://localhost:2772/applications/MyDemoApp/environments/
Beta/configurations/MyConfigProfile");
    HttpURLConnection con = (HttpURLConnection) url.openConnection();

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

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

    StringBuilder content;
    try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
        content = new StringBuilder();
        int ch;
        while ((ch = in.read()) != -1) {
            content.append((char) ch);
        }
    }
    con.disconnect();
    System.out.println("Configuration from agent via HTTP: " + content);
}
```

Python

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

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

import requests

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

# make a GET request to the agent's local server to retrieve the configuration data
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{configuration_profile_name}",
                        headers = {
                            "Context": "country=US" # Provide context in the
 'Context' header
                                                    # In the header value, use '='
 to separate context key from context value
                                                    # Note: Multiple context values
 may be passed either across
                                                    # multiple headers or as comma-
separated values in a single header
```

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

JavaScript

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

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

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

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

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

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

Utilisation de l'action GetLatestConfiguration API pour lire un profil de configuration de forme libre

Chacun des exemples suivants inclut des commentaires sur les actions effectuées par le code. Les exemples présentés dans cette section appellent ce qui suit APIs :

- GetLatestConfiguration
- StartConfigurationSession

Java

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

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

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

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

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

    /*
    Returns the latest configuration data stored in AWS AppConfig.
     */
    public SdkBytes getConfig() {
```

```
       /*
       If there is no configuration token yet, get one by starting a new session
with the StartConfigurationSession API.
       Note that this API does not return configuration data. Rather, it returns an
initial configuration token that is
       subsequently passed to the GetLatestConfiguration API.
        */
       if (this.configurationToken == null) {
           StartConfigurationSessionResponse session =
appConfigData.startConfigurationSession(req -> req
                   .applicationIdentifier("MyDemoApp")
                   .configurationProfileIdentifier("MyConfig")
                   .environmentIdentifier("Beta"));
           this.configurationToken = session.initialConfigurationToken();
       }

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

GetLatestConfigurationRequest.builder().configurationToken(this.configurationToken).build()

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

       /*
       If the GetLatestConfiguration API returned configuration data, update the
cached configuration with the returned data.
```

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

        return this.configuration;
    }
}
```

Python

```
# The example below uses two AWS AppConfig Data APIs: StartConfigurationSession and
 GetLatestConfiguration.
# For more information about these APIs, see AWS AppConfig Data.
#
# This class is meant to be used as a singleton to retrieve the latest configuration
 data from AWS AppConfig.
# This class maintains a cache of the latest configuration data in addition to the
 configuration token to be
# passed to the next GetLatestConfiguration API call.
class AppConfigApiRetriever:
    def __init__(self):
        # AWS AppConfig Data SDK client used to interact with the AWS AppConfig Data
 service.
        self.appconfigdata = boto3.client('appconfigdata')

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

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

    # Returns the latest configuration data stored in AWS AppConfig.
```

```
   def get_config(self):
       # If there is no configuration token yet, get one by starting a new session
with the StartConfigurationSession API.
       # Note that this API does not return configuration data. Rather, it returns
an initial configuration token that is
       # subsequently passed to the GetLatestConfiguration API.
       if not self.configuration_token:
           session = self.appconfigdata.start_configuration_session(
               ApplicationIdentifier='MyDemoApp',
               ConfigurationProfileIdentifier='MyConfig',
               EnvironmentIdentifier='Beta'
           )
           self.configuration_token = session['InitialConfigurationToken']

       # Retrieve the configuration from the GetLatestConfiguration API, providing
the current configuration token.
       # If this caller does not yet have the latest configuration (e.g. this is
the first call to GetLatestConfiguration
       # or new configuration data has been deployed since the first call), the
latest configuration data will be returned.
       # Otherwise, the GetLatestConfiguration API will not return any data since
the caller already has the latest.
       response =
self.appconfigdata.get_latest_configuration(ConfigurationToken=self.configuration_token)

       # Save the returned configuration token so that it can be passed to the next
GetLatestConfiguration API call.
       # Warning: Not persisting this token for use in the next
GetLatestConfiguration API call may result in higher
       # than expected usage costs.
       self.configuration_token = response['NextPollConfigurationToken']

       # If the GetLatestConfiguration API returned configuration data, update the
cached configuration with the returned data.
       # Otherwise, assume the configuration has not changed, and return the cached
configuration.
       config_from_api = response['Configuration'].read()
       if config_from_api:
           self.configuration = config_from_api
           print('Configuration contents have changed since the last
GetLatestConfiguration call, new contents = ' + str(self.configuration))
       else:
           print('GetLatestConfiguration returned an empty response because we
already have the latest configuration')
```

```
        return self.configuration
```

JavaScript

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

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

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

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

    /*
    Returns the latest configuration data stored in AWS AppConfig.
     */
    async getConfig() {
        /*
        If there is no configuration token yet, get one by starting a new session
 with the StartConfigurationSession API.
```

```
        Note that this API does not return configuration data. Rather, it returns an
initial configuration token that is
        subsequently passed to the GetLatestConfiguration API.
        */
        if (!this.configurationToken) {
            const session = await this.appconfigdata.send(
                new StartConfigurationSessionCommand({
                    ApplicationIdentifier: "MyDemoApp",
                    ConfigurationProfileIdentifier: "MyConfig",
                    EnvironmentIdentifier: "Beta"
                })
            );
            this.configurationToken = session.InitialConfigurationToken;
        }

        /*
        Retrieve the configuration from the GetLatestConfiguration API, providing
the current configuration token.
        If this caller does not yet have the latest configuration (e.g. this is the
first call to GetLatestConfiguration
        or new configuration data has been deployed since the first call), the
latest configuration data will be returned.
        Otherwise, the GetLatestConfiguration API will not return any data since the
caller already has the latest.
        */
        const response = await this.appconfigdata.send(
            new GetLatestConfigurationCommand({
                ConfigurationToken: this.configurationToken
            })
        );

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

        /*
        If the GetLatestConfiguration API returned configuration data, update the
cached configuration with the returned data.
```

```
        Otherwise, assume the configuration has not changed, and return the cached
 configuration.
        */
        const configFromApi = response.Configuration.transformToString();
        if (configFromApi) {
            this.configuration = configFromApi;
            console.log("Configuration contents have changed since the last
 GetLatestConfiguration call, new contents = " + this.configuration);
        } else {
            console.log("GetLatestConfiguration returned an empty response because
 we already have the latest configuration");
        }

        return this.configuration;
    }
}
```

Nettoyage de votre environnement

Si vous avez exécuté un ou plusieurs exemples de code de cette section, nous vous recommandons d'utiliser l'un des exemples suivants pour localiser et supprimer les AWS AppConfig ressources créées par ces exemples de code. Les exemples présentés dans cette section appellent ce qui suit APIs :

- ListApplications
- DeleteApplication
- ListEnvironments
- DeleteEnvironments
- ListConfigurationProfiles
- DeleteConfigurationProfile
- ListHostedConfigurationVersions
- DeleteHostedConfigurationVersion

Java

```
/*
    This sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.
```

```
   WARNING: this code will permanently delete the given application and all of its
sub-resources, including
   configuration profiles, hosted configuration versions, and environments. DO NOT
run this code against
   an application that you may need in the future.
   */

   public void cleanUpDemoResources() {
       AppConfigClient appconfig = AppConfigClient.create();

       // The name of the application to delete
       // IMPORTANT: verify this name corresponds to the application you wish to
delete
       String applicationToDelete = "MyDemoApp";


appconfig.listApplicationsPaginator(ListApplicationsRequest.builder().build()).items().forE
-> {
           if (app.name().equals(applicationToDelete)) {
               System.out.println("Deleting App: " + app);
               appconfig.listConfigurationProfilesPaginator(req ->
req.applicationId(app.id())).items().forEach(cp -> {
                   System.out.println("Deleting Profile: " + cp);
                   appconfig
                       .listHostedConfigurationVersionsPaginator(req -> req
                           .applicationId(app.id())
                           .configurationProfileId(cp.id()))
                       .items()
                       .forEach(hcv -> {
                           System.out.println("Deleting HCV: " + hcv);
                           appconfig.deleteHostedConfigurationVersion(req -> req
                               .applicationId(app.id())
                               .configurationProfileId(cp.id())
                               .versionNumber(hcv.versionNumber()));
                       });
                   appconfig.deleteConfigurationProfile(req -> req
                       .applicationId(app.id())
                       .configurationProfileId(cp.id()));
               });

               appconfig.listEnvironmentsPaginator(req-
>req.applicationId(app.id())).items().forEach(env -> {
                   System.out.println("Deleting Environment: " + env);
```

```
                    appconfig.deleteEnvironment(req-
>req.applicationId(app.id()).environmentId(env.id()));
                });

                appconfig.deleteApplication(req -> req.applicationId(app.id()));
            }
        });
    }
```

Python

```
# this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.
#
# WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
#   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
#   an application that you may need in the future.
#

import boto3

# the name of the application to delete
# IMPORTANT: verify this name corresponds to the application you wish to delete
application_name = 'MyDemoApp'

# create and iterate over a list paginator such that we end up with a list of pages,
 which are themselves lists of applications
# e.g. [ [{'Name':'MyApp1',...},{'Name':'MyApp2',...}], [{'Name':'MyApp3',...}] ]
list_of_app_lists = [page['Items'] for page in
 appconfig.get_paginator('list_applications').paginate()]
# retrieve the target application from the list of lists
application = [app for apps in list_of_app_lists for app in apps if app['Name'] ==
 application_name][0]
print(f"deleting application {application['Name']} (id={application['Id']})")

# delete all configuration profiles
list_of_config_lists = [page['Items'] for page in
 appconfig.get_paginator('list_configuration_profiles').paginate(ApplicationId=application['
for config_profile in [config for configs in list_of_config_lists for config in
 configs]:
```

```
    print(f"\tdeleting configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

    # delete all hosted configuration versions
    list_of_hcv_lists = [page['Items'] for page in
 appconfig.get_paginator('list_hosted_configuration_versions').paginate(ApplicationId=applic
 ConfigurationProfileId=config_profile['Id'])]
    for hcv in [hcv for hcvs in list_of_hcv_lists for hcv in hcvs]:

 appconfig.delete_hosted_configuration_version(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'], VersionNumber=hcv['VersionNumber'])
        print(f"\t\tdeleted hosted configuration version {hcv['VersionNumber']}")

    # delete the config profile itself
    appconfig.delete_configuration_profile(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'])
    print(f"\tdeleted configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

# delete all environments
list_of_env_lists = [page['Items'] for page in
 appconfig.get_paginator('list_environments').paginate(ApplicationId=application['Id'])]
for environment in [env for envs in list_of_env_lists for env in envs]:
    appconfig.delete_environment(ApplicationId=application['Id'],
 EnvironmentId=environment['Id'])
    print(f"\tdeleted environment {environment['Name']} (Id={environment['Id']})")

# delete the application itself
appconfig.delete_application(ApplicationId=application['Id'])
print(f"deleted application {application['Name']} (id={application['Id']})")
```

JavaScript

```
// this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.

// WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
//   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
//   an application that you may need in the future.

import {
```

```
  AppConfigClient,
  paginateListApplications,
  DeleteApplicationCommand,
  paginateListConfigurationProfiles,
  DeleteConfigurationProfileCommand,
  paginateListHostedConfigurationVersions,
  DeleteHostedConfigurationVersionCommand,
  paginateListEnvironments,
  DeleteEnvironmentCommand,
} from "@aws-sdk/client-appconfig";

const client = new AppConfigClient();

// the name of the application to delete
// IMPORTANT: verify this name corresponds to the application you wish to delete
const application_name = "MyDemoApp";

// iterate over all applications, deleting ones that have the name defined above
for await (const app_page of paginateListApplications({ client }, {})) {
  for (const application of app_page.Items) {

    // skip applications that dont have the name thats set
    if (application.Name !== application_name) continue;

    console.log( `deleting application ${application.Name} (id=${application.Id})`);

    // delete all configuration profiles
    for await (const config_page of paginateListConfigurationProfiles({ client },
 { ApplicationId: application.Id })) {
      for (const config_profile of config_page.Items) {
        console.log(`\tdeleting configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`);

        // delete all hosted configuration versions
        for await (const hosted_page of
 paginateListHostedConfigurationVersions({ client },
          { ApplicationId: application.Id, ConfigurationProfileId:
 config_profile.Id }
        )) {
          for (const hosted_config_version of hosted_page.Items) {
            await client.send(
              new DeleteHostedConfigurationVersionCommand({
                ApplicationId: application.Id,
                ConfigurationProfileId: config_profile.Id,
```

```
                VersionNumber: hosted_config_version.VersionNumber,
              })
            );
            console.log(`\t\tdeleted hosted configuration version
 ${hosted_config_version.VersionNumber}`);
          }
        }

        // delete the config profile itself
        await client.send(
          new DeleteConfigurationProfileCommand({
            ApplicationId: application.Id,
            ConfigurationProfileId: config_profile.Id,
          })
        );
        console.log(`\tdeleted configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`)
      }

      // delete all environments
      for await (const env_page of paginateListEnvironments({ client },
 { ApplicationId: application.Id })) {
        for (const environment of env_page.Items) {
          await client.send(
            new DeleteEnvironmentCommand({
              ApplicationId: application.Id,
              EnvironmentId: environment.Id,
            })
          );
          console.log(`\tdeleted environment ${environment.Name} (Id=
${environment.Id})`)
        }
      }
    }

    // delete the application itself
    await client.send(
      new DeleteApplicationCommand({ ApplicationId: application.Id })
    );
    console.log(`deleted application ${application.Name} (id=${application.Id})`)
  }
}
```

Configuration de la protection contre la AWS AppConfig suppression

AWS AppConfig fournit un paramètre de compte pour empêcher les utilisateurs de supprimer par inadvertance des environnements et des profils de configuration utilisés activement. AWS AppConfig surveille les appels vers GetLatestConfigurationGetConfigurationet suit les profils de configuration et les environnements inclus dans ces appels dans un intervalle de 60 minutes (paramètre par défaut). Tout profil ou environnement de configuration auquel vous avez accédé pendant cet intervalle sera considéré comme actif. Si vous tentez de supprimer un profil ou un environnement de configuration actif, AWS AppConfig renvoie une erreur. Si nécessaire, vous pouvez contourner cette erreur en utilisant le `DeletionProtectionCheck` paramètre. Pour de plus amples informations, veuillez consulter Contourner ou forcer une vérification de protection contre la suppression.

Configuration de la protection contre la suppression à l'aide de la console

Utilisez la procédure suivante pour configurer la protection contre la suppression à l'aide de la AWS Systems Manager console.

Pour configurer la protection contre les suppressions (console)

1. Ouvrez la AWS Systems Manager console à l'adresse https://console.aws.amazon.com/systems-manager/appconfig/.
2. Dans le panneau de navigation, sélectionnez Settings (Paramètres).
3. Utilisez le bouton pour activer ou désactiver la protection contre la suppression.
4. Pour la période de protection, définissez la définition d'une ressource active entre 15 et 1440 minutes.
5. Cliquez sur Apply.

Configurer la protection contre la suppression à l'aide du AWS CLI

Utilisez la procédure suivante pour configurer la protection contre la suppression à l'aide du AWS CLI. Remplacez *`value`* les commandes suivantes par la valeur que vous souhaitez utiliser dans votre environnement.

> **Note**
>
> Avant de commencer, nous vous recommandons de passer à la dernière version du AWS CLI. Pour plus d'informations, voir Installer ou mettre à jour la dernière version du AWS CLI dans le guide de AWS Command Line Interface l'utilisateur.

Pour configurer la protection contre la suppression (CLI)

1. Exécutez la commande suivante pour afficher les paramètres actuels de protection contre la suppression.

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

2. Exécutez la commande suivante pour activer ou désactiver la protection contre la suppression. Spécifiez `false` si vous souhaitez désactiver ou `true` activer la protection contre la suppression.

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

3. Vous pouvez augmenter l'intervalle par défaut jusqu'à 24 heures au maximum. Exécutez la commande suivante pour définir un nouvel intervalle.

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

Contourner ou forcer une vérification de protection contre la suppression

Pour vous aider à gérer la protection contre les suppressions, DeleteEnvironmentet DeleteConfigurationProfile APIs incluez un paramètre appelé`DeletionProtectionCheck`. Ce paramètre prend en charge les valeurs suivantes :

- `BYPASS`: indique de AWS AppConfig contourner le contrôle de protection contre la suppression et de supprimer un profil de configuration même si la protection contre la suppression l'aurait autrement empêché.

- `APPLY`: demande au contrôle de protection contre la suppression de s'exécuter, même si la protection contre la suppression est désactivée au niveau du compte. `APPLY`force également le contrôle de protection contre la suppression à être exécuté sur les ressources créées au cours de la dernière heure, qui sont normalement exclues des contrôles de protection contre la suppression.
- `ACCOUNT_DEFAULT`: paramètre par défaut, qui indique d' AWS AppConfig implémenter la valeur de protection contre la suppression spécifiée dans l'`UpdateAccountSettings`API.

> **Note**
>
> Par défaut, `DeletionProtectionCheck` ignore les profils de configuration et les environnements créés au cours de la dernière heure. La configuration par défaut est destinée à empêcher la protection contre la suppression d'interférer avec les tests et les démonstrations qui créent des ressources de courte durée. Vous pouvez annuler ce comportement en le transmettant `DeletionProtectionCheck=APPLY` lorsque vous appelez `DeleteEnvironment` ou`DeleteConfigurationProfile`.

La procédure pas à pas de la CLI suivante utilise des exemples de commandes pour illustrer l'utilisation du `DeletionProtectionCheck` paramètre. Remplacez *`ID`* les commandes suivantes par l'ID de vos AWS AppConfig artefacts.

1. Faites appel GetLatestConfigurationà une configuration déployée.

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

2. Attendez 60 secondes AWS AppConfig pour enregistrer que la configuration est active.
3. Exécutez la commande suivante pour appeler DeleteEnvironmentet appliquer la protection contre la suppression à l'environnement.

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

   La commande doit renvoyer l'erreur suivante.

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

4. Exécutez la commande suivante pour contourner la protection contre la suppression et supprimer l'environnement.

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

Sécurité dans AWS AppConfig

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Systems Manager, voir AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

AWS AppConfig est un outil dans AWS Systems Manager. Pour comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation AWS AppConfig, voir Security in AWS Systems Manager. Cette section décrit comment configurer Systems Manager pour répondre aux objectifs de sécurité et de conformité de AWS AppConfig.

Implémentation d'un accès sur la base du moindre privilège

En tant que bonne pratique de sécurité, accordez les autorisations minimales requises par les identités pour effectuer des actions spécifiques sur des ressources spécifiques dans des conditions spécifiques. AWS AppConfig L'agent propose deux fonctionnalités qui lui permettent d'accéder au système de fichiers d'une instance ou d'un conteneur : sauvegarde et écriture sur disque. Si vous activez ces fonctionnalités, vérifiez que seul l' AWS AppConfig agent est autorisé à écrire dans les fichiers de configuration désignés sur le système de fichiers. Vérifiez également que seuls les processus requis pour lire ces fichiers de configuration sont en mesure de le faire. L'implémentation d'un accès sur la base du moindre privilège est fondamentale pour réduire les risques en matière de sécurité et l'impact que pourraient avoir des d'erreurs ou des actes de malveillance.

Pour plus d'informations sur la mise en œuvre de l'accès au moindre privilège, consultez la section SEC03-BP02 Accorder l'accès au moindre privilège dans le guide de l'AWS Well-Architected Tool utilisateur. Pour plus d'informations sur les fonctionnalités de l' AWS AppConfig agent mentionnées dans cette section, consultezUtilisation d'un manifeste pour activer des fonctionnalités de récupération supplémentaires.

Chiffrement des données au repos pour AWS AppConfig

AWS AppConfig fournit un chiffrement par défaut pour protéger les données clients au repos lors de l'utilisation Clés détenues par AWS.

Clés détenues par AWS— AWS AppConfig utilise ces clés par défaut pour chiffrer automatiquement les données déployées par le service et hébergées dans le magasin de AWS AppConfig données. Vous ne pouvez ni afficher, ni gérer, ni utiliser Clés détenues par AWS, ni auditer leur utilisation. Toutefois, vous n'avez pas besoin de prendre de mesure ou de modifier les programmes pour protéger les clés qui chiffrent vos données. Pour plus d'informations, consultez Clés détenues par AWS dans le Guide du développeur AWS Key Management Service .

Bien que vous ne puissiez pas désactiver cette couche de chiffrement ou sélectionner un autre type de chiffrement, vous pouvez spécifier une clé gérée par le client à utiliser lorsque vous enregistrez les données de configuration hébergées dans le magasin de AWS AppConfig données et lorsque vous déployez vos données de configuration.

Clés gérées par le client : AWS AppConfig prend en charge l'utilisation d'une clé symétrique gérée par le client que vous créez, détenez et gérez pour ajouter une deuxième couche de chiffrement à la couche existante Clé détenue par AWS. Étant donné que vous avez le contrôle total de cette couche de chiffrement, vous pouvez effectuer les tâches suivantes :

- Établir et maintenir des politiques et des subventions clés
- Établir et maintenir des politiques IAM
- Activation et désactivation des stratégies de clé
- Rotation des matériaux de chiffrement de clé
- Ajout de balises
- Création d'alias de clé
- Planification des clés pour la suppression

Pour plus d'informations, consultez la section Clé gérée par le client dans le guide du AWS Key Management Service développeur.

AWS AppConfig prend en charge les clés gérées par le client

AWS AppConfig offre un support pour le chiffrement des clés géré par le client pour les données de configuration. Pour les versions de configuration enregistrées dans le magasin de données AWS AppConfig hébergé, les clients peuvent définir un `KmsKeyIdentifier` sur le profil de configuration correspondant. Chaque fois qu'une nouvelle version des données de configuration est créée à l'aide de l'opération `CreateHostedConfigurationVersion` API, une clé de AWS KMS données est AWS AppConfig générée `KmsKeyIdentifier` à partir du pour chiffrer les données avant de les stocker. Lorsque les données sont consultées ultérieurement, soit pendant les opérations soit pendant les `GetHostedConfigurationVersion` opérations de l'`StartDeployment`API, il AWS AppConfig déchiffre les données de configuration à l'aide des informations relatives à la clé de données générée.

AWS AppConfig offre également un support pour le chiffrement des clés géré par le client pour les données de configuration déployées. Pour chiffrer les données de configuration, les clients peuvent fournir un `KmsKeyIdentifier` à leur déploiement. AWS AppConfig génère la clé de AWS KMS données avec celle-ci `KmsKeyIdentifier` pour crypter les données sur le fonctionnement de l'`StartDeployment`API.

AWS AppConfig accès au chiffrement

Lorsque vous créez une clé gérée par le client, appliquez la politique de clé suivante pour vous assurer que la clé peut être utilisée.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
```

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

Pour chiffrer les données de configuration hébergées à l'aide d'une clé gérée par le client, l'appel d'identité `CreateHostedConfigurationVersion` a besoin de la déclaration de politique suivante, qui peut être attribuée à un utilisateur, à un groupe ou à un rôle :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey",
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Si vous utilisez un secret Secrets Manager ou toute autre donnée de configuration chiffrée à l'aide d'une clé gérée par le client, vous `retrievalRoleArn` `kms:Decrypt` devrez déchiffrer et récupérer les données.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:account_ID:configuration source/object"
    }
  ]
```

```
}
```

Lors de l'appel de l'opération AWS AppConfig StartDeploymentAPI, l'appel d'identité `StartDeployment` nécessite la politique IAM suivante, qui peut être attribuée à un utilisateur, à un groupe ou à un rôle :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:Region:account_ID:key_ID"
    }
  ]
}
```

Lors de l'appel de l'opération AWS AppConfig GetLatestConfigurationAPI, l'appel d'identité `GetLatestConfiguration` nécessite la politique suivante, qui peut être attribuée à un utilisateur, à un groupe ou à un rôle :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Contexte de chiffrement

Un contexte de chiffrement est un ensemble facultatif de paires clé-valeur qui contient des informations contextuelles supplémentaires sur les données.

AWS KMS utilise le contexte de chiffrement comme données authentifiées supplémentaires pour prendre en charge le chiffrement authentifié. Lorsque vous incluez un contexte de chiffrement dans une demande de chiffrement de données, AWS KMS lie le contexte de chiffrement aux données chiffrées. Pour déchiffrer les données, vous devez inclure le même contexte de chiffrement dans la demande.

AWS AppConfig contexte de chiffrement : AWS AppConfig utilise un contexte de chiffrement dans toutes les opérations AWS KMS cryptographiques pour les données de configuration hébergées chiffrées et les déploiements. Le contexte contient une clé correspondant au type de données et une valeur identifiant l'élément de données spécifique.

Surveillance de vos clés de chiffrement pour AWS

Lorsque vous utilisez des clés gérées par un AWS KMS client avec AWS AppConfig, vous pouvez utiliser AWS CloudTrail Amazon CloudWatch Logs pour suivre les demandes AWS AppConfig envoyées à AWS KMS.

L'exemple suivant est un CloudTrail événement destiné `Decrypt` à surveiller les AWS KMS opérations appelées pour accéder AWS AppConfig aux données chiffrées par votre clé gérée par le client :

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "appconfig.amazonaws.com"
    },
    "eventTime": "2023-01-03T02:22:28z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "Region",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
```

```
            "aws:appconfig:deployment:arn":
 "arn:aws:appconfig:Region:account_ID:application/application_ID/
environment/environment_ID/deployment/deployment_ID"
        },
        "keyId": "arn:aws:kms:Region:account_ID:key/key_ID",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account_ID",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "account_ID",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

Accès AWS AppConfig via un point de terminaison d'interface (AWS PrivateLink)

Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et. AWS AppConfig Vous pouvez y accéder AWS AppConfig comme s'il se trouvait dans votre VPC, sans utiliser de passerelle Internet, de périphérique NAT, de connexion VPN ou AWS Direct Connect de connexion. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour y accéder. AWS AppConfig

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par le demandeur qui servent de point d'entrée pour le trafic destiné à AWS AppConfig.

Pour plus d'informations, consultez Accès aux Services AWS via AWS PrivateLink dans le Guide AWS PrivateLink .

Considérations relatives à AWS AppConfig

Avant de configurer un point de terminaison d'interface pour AWS AppConfig, consultez les considérations du AWS PrivateLink guide.

AWS AppConfig permet de passer des appels aux `appconfigdata`services `appconfig`et via le point de terminaison de l'interface.

Création d'un point de terminaison d'interface pour AWS AppConfig

Vous pouvez créer un point de terminaison d'interface pour AWS AppConfig utiliser la console Amazon VPC ou le AWS Command Line Interface ()AWS CLI. Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide AWS PrivateLink .

Créez un point de terminaison d'interface pour AWS AppConfig utiliser les noms de service suivants :

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

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

Si vous activez le DNS privé pour le point de terminaison de l'interface, vous pouvez envoyer des demandes d'API à AWS AppConfig l'aide de son nom DNS régional par défaut. Par exemple : `appconfig.us-east-1.amazonaws.com` et `appconfigdata.us-east-1.amazonaws.com`.

Création d'une politique de point de terminaison pour votre point de terminaison d'interface

Une politique de point de terminaison est une ressource IAM que vous pouvez attacher à votre point de terminaison d'interface. La politique de point de terminaison par défaut autorise un accès complet AWS AppConfig via le point de terminaison de l'interface. Pour contrôler l'accès autorisé AWS AppConfig depuis votre VPC, associez une politique de point de terminaison personnalisée au point de terminaison de l'interface.

Une politique de point de terminaison spécifie les informations suivantes :

- Les principaux qui peuvent effectuer des actions (Comptes AWS, utilisateurs IAM et rôles IAM).
- Les actions qui peuvent être effectuées.
- La ressource sur laquelle les actions peuvent être effectuées.

Pour plus d'informations, consultez Contrôle de l'accès aux services à l'aide de politiques de point de terminaison dans le Guide AWS PrivateLink .

Exemple : politique de point de terminaison VPC pour les actions AWS AppConfig

Voici un exemple de politique de point de terminaison. Lorsque vous attachez cette politique à votre point de terminaison d'interface, elle accorde l'accès aux actions AWS AppConfig répertoriées pour tous les principaux sur toutes les ressources.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "appconfig:CreateApplication",
            "appconfig:CreateEnvironment",
            "appconfig:CreateConfigurationProfile",
            "appconfig:StartDeployment",
            "appconfig:GetLatestConfiguration"
            "appconfig:StartConfigurationSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Rotation des clés de Secrets Manager

Cette section décrit les informations de sécurité importantes concernant AWS AppConfig l'intégration avec Secrets Manager. Pour plus d'informations sur Secrets Manager, consultez Qu'est-ce que c'est AWS Secrets Manager ? dans le guide de AWS Secrets Manager l'utilisateur.

Configuration de la rotation automatique des secrets Secrets Manager déployés par AWS AppConfig

La rotation est le processus de mise à jour périodique d'un secret stocké dans Secrets Manager. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et dans la base de données ou le service. Vous pouvez configurer la rotation automatique des secrets dans Secrets Manager à l'aide d'une AWS Lambda fonction de mise à jour

du secret et de la base de données. Pour plus d'informations, voir Rotation AWS Secrets Manager des secrets dans le guide de AWS Secrets Manager l'utilisateur.

Pour activer la rotation des clés des secrets Secrets Manager déployés par AWS AppConfig, mettez à jour votre fonction Lambda de rotation et déployez le secret pivoté.

> **Note**
>
> Déployez votre profil de AWS AppConfig configuration une fois que votre secret a été modifié et entièrement mis à jour vers la nouvelle version. Vous pouvez déterminer si le secret a changé parce que le statut `VersionStage` passe de `AWSPENDING` à`AWSCURRENT`. La fin de la rotation secrète s'effectue dans le cadre de la `finish_secret` fonction Modèles de rotation de Secrets Manager.

Voici un exemple de fonction qui lance un AWS AppConfig déploiement après la rotation d'un secret.

```
import time
import boto3
client = boto3.client('appconfig')

def finish_secret(service_client, arn, new_version):
    """Finish the rotation by marking the pending secret as current
    This method finishes the secret rotation by staging the secret staged AWSPENDING
 with the AWSCURRENT stage.
    Args:
        service_client (client): The secrets manager service client
        arn (string): The secret ARN or other identifier
        new_version (string): The new version to be associated with the secret
    """
    # First describe the secret to get the current version
    metadata = service_client.describe_secret(SecretId=arn)
    current_version = None
    for version in metadata["VersionIdsToStages"]:
        if "AWSCURRENT" in metadata["VersionIdsToStages"][version]:
            if version == new_version:
                # The correct version is already marked as current, return
                logger.info("finishSecret: Version %s already marked as AWSCURRENT for
 %s" % (version, arn))
                return
            current_version = version
            break
```

```
   # Finalize by staging the secret version current
   service_client.update_secret_version_stage(SecretId=arn, VersionStage="AWSCURRENT",
MoveToVersionId=new_version, RemoveFromVersionId=current_version)

   # Deploy rotated secret
   response = client.start_deployment(
           ApplicationId='TestApp',
           EnvironmentId='TestEnvironment',
           DeploymentStrategyId='TestStrategy',
           ConfigurationProfileId='ConfigurationProfileId',
           ConfigurationVersion=new_version,
           KmsKeyIdentifier=key,
           Description='Deploy secret rotated at ' + str(time.time())
       )

   logger.info("finishSecret: Successfully set AWSCURRENT stage to version %s for
secret %s." % (new_version, arn))
```

Surveillance AWS AppConfig

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS AppConfig et des performances de vos autres AWS solutions. AWS fournit les outils de surveillance suivants pour surveiller AWS AppConfig, signaler tout problème et prendre des mesures automatiques le cas échéant :

- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos EC2 instances Amazon et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.
- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CloudTrail.
- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux à partir d' EC2 instances Amazon et d'autres sources. CloudTrail CloudWatch Les journaux peuvent surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon CloudWatch Logs.
- Amazon EventBridge peut être utilisé pour automatiser vos AWS services et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements AWS liés aux services sont diffusés EventBridge en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.

Rubriques

- Métriques de journalisation pour les appels du plan de AWS AppConfig données
- Surveillance des déploiements pour une annulation automatique

Journalisation des appels AWS AppConfig d'API à l'aide AWS CloudTrail

AWS AppConfig est intégré à AWS CloudTrailun service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API AWS AppConfig sous forme d'événements. Les appels capturés incluent des appels provenant de la AWS AppConfig console et des appels de code vers les opérations de l' AWS AppConfig API. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite AWS AppConfig, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des informations supplémentaires.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou d'utilisateur root.
- Si la demande a été faite au nom d'un utilisateur du centre d'identité IAM.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section Utilisation de l'historique des CloudTrail événements dans le guide de AWS CloudTrail l'utilisateur. La consultation de CloudTrail l'historique des événements est gratuite.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou CloudTrailLake.

CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous ne pouvez créer un journal de suivi en une ou plusieurs régions à l'aide de l' AWS CLI. Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un journal de suivi pour une seule région, il convient de n'afficher que les événements enregistrés dans le journal de suivi pour une seule région Région AWS. Pour plus d'informations sur les journaux de suivi, consultez Créez un journal de suivi dans vos Compte AWS et Création d'un journal de suivi pour une organisation dans le AWS CloudTrail Guide de l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours dans votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification. Pour obtenir des informations sur la tarification Amazon S3, consultez Tarification Amazon S3.

CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format Apache ORC. ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des sélecteurs d'événements avancés. Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section Travailler avec AWS CloudTrail Lake dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'option de tarification que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

AWS AppConfig événements de données dans CloudTrail

Les événements de données fournissent des informations sur les opérations de ressources effectuées sur ou dans une ressource (par exemple, récupération de la dernière configuration déployée en appelant GetLatestConfiguration). Ils sont également connus sous le nom opérations de plans de données. Les événements de données sont souvent des activités dont le volume est élevé. Par défaut, CloudTrail n'enregistre pas les événements liés aux données. L'historique des CloudTrail événements n'enregistre pas les événements liés aux données.

Des frais supplémentaires s'appliquent pour les événements de données. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Vous pouvez enregistrer les événements de données pour les types de AWS AppConfig ressources à l'aide de la CloudTrail console ou AWS CLI des opérations de CloudTrail l'API. Le tableau de cette section indique les types de ressources disponibles pour AWS AppConfig.

- Pour enregistrer les événements de données à l'aide de la CloudTrail console, créez un magasin de données de suivi ou d'événement pour enregistrer les événements, ou mettez à jour un magasin de données de suivi ou d'événement existant pour enregistrer les événements de données.

  1. Choisissez Data events pour enregistrer les événements liés aux données.
  2. Dans la liste des types d'événements de données, sélectionnez AWS AppConfig.
  3. Choisissez le modèle de sélecteur de journal que vous souhaitez utiliser. Vous pouvez enregistrer tous les événements de données pour le type de ressource, consigner tous les `readOnly` événements, consigner tous les `writeOnly` événements ou créer un modèle de sélecteur de journal personnalisé pour filtrer les `resources.ARN` champs `readOnlyeventName`, et.
  4. Dans Nom du sélecteur, entrez AppConfigDataEvents. Pour plus d'informations sur l'activation d'Amazon CloudWatch Logs pour le suivi de vos événements liés aux données, consultezMétriques de journalisation pour les appels du plan de AWS AppConfig données.
- Pour enregistrer des événements de données à l'aide de AWS CLI, configurez le `--advanced-event-selectors` paramètre pour définir le `eventCategory` champ égal à la valeur du type de ressource `Data` et le `resources.type` champ égal à la valeur du type de ressource (voir le tableau). Vous pouvez ajouter des conditions pour filtrer les valeurs des `resources.ARN` champs `readOnlyeventName`, et.

- Pour configurer un journal afin de consigner les événements liés aux données, exécutez put-event-selectorscommande. Pour plus d'informations, consultez la section Enregistrement des événements de données pour les sentiers avec le AWS CLI.
- Pour configurer un magasin de données d'événements afin de consigner les événements de données, exécutez create-event-data-storecommande pour créer un nouveau magasin de données d'événements afin de consigner les événements de données, ou pour exécuter le update-event-data-storecommande pour mettre à jour un magasin de données d'événements existant. Pour plus d'informations, consultez la section Enregistrement des événements de données pour les magasins de données d'événements avec le AWS CLI.

Le tableau suivant répertorie les types de AWS AppConfig ressources. La colonne Type d'événement de données (console) indique la valeur à choisir dans la liste des types d'événements de données de la CloudTrail console. La colonne de valeur resources.type indique la **`resources.type`** valeur que vous devez spécifier lors de la configuration de sélecteurs d'événements avancés à l'aide du ou. AWS CLI CloudTrail APIs La CloudTrail colonne Données APIs enregistrées indique les appels d'API enregistrés CloudTrail pour le type de ressource.

Type d'événement de données (console)	valeur resources.type	Données APIs enregistrées sur CloudTrail *
AWS AppConfig	`AWS::AppConfig::Configuration`	• GetLatestConfiguration • StartConfigurationSession

*Vous pouvez configurer des sélecteurs d'événements avancés pour filtrer les `eventName`, `readOnly`, et des `resources.ARN` champs pour enregistrer uniquement les événements qui sont importants pour vous. Pour plus d'informations sur ces champs, voir AdvancedFieldSelector.

AWS AppConfig événements de gestion dans CloudTrail

AWS AppConfig enregistre toutes les opérations AWS AppConfig du plan de contrôle en tant qu'événements de gestion. Pour obtenir la liste des opérations du plan de AWS AppConfig contrôle auxquelles il est AWS AppConfig possible de se connecter CloudTrail, consultez la référence de l'AWS AppConfig API.

AWS AppConfig exemples d'événements

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un CloudTrail événement illustrant l'StartConfigurationSessionopération.


```
{
      "eventVersion": "1.09",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Administrator",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {},
          "attributes": {
            "creationDate": "2024-01-11T14:37:02Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2024-01-11T14:45:15Z",
      "eventSource": "appconfig.amazonaws.com",
      "eventName": "StartConfigurationSession",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "Boto3/1.34.11 md/Botocore#1.34.11 ua/2.0 os/macos#22.6.0
 md/arch#x86_64 lang/python#3.11.4 md/pyimpl#CPython cfg/retry-mode#legacy
 Botocore/1.34.11",
      "requestParameters": {
        "applicationIdentifier": "rrfexample",
        "environmentIdentifier": "mexampleqe0",
        "configurationProfileIdentifier": "3eexampleu1"
      },
      "responseElements": null,
      "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
      "readOnly": false,
      "resources": [
```

```
        {
          "accountId": "123456789012",
          "type": "AWS::AppConfig::Configuration",
          "ARN": "arn:aws:appconfig:us-east-1:123456789012:application/rrfexample/
environment/mexampleqe0/configuration/3eexampleu1"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "123456789012",
      "eventCategory": "Data",
      "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "appconfigdata.us-east-1.amazonaws.com"
      }
    }
```


Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir le contenu des CloudTrail enregistrements dans le Guide de AWS CloudTrail l'utilisateur.

Métriques de journalisation pour les appels du plan de AWS AppConfig données

Si vous avez configuré AWS CloudTrail pour consigner AWS AppConfig les événements liés aux données, vous pouvez activer Amazon CloudWatch Logs pour enregistrer les métriques relatives aux appels vers le plan de AWS AppConfig données. Vous pouvez ensuite rechercher et filtrer les données des CloudWatch journaux dans Logs en créant un ou plusieurs filtres métriques. Les filtres métriques définissent les termes et les modèles à rechercher dans les données du journal lorsqu'elles sont envoyées à CloudWatch Logs. CloudWatch Logs utilise des filtres métriques pour transformer les données des journaux en CloudWatch métriques numériques. Vous pouvez représenter graphiquement les métriques ou les configurer à l'aide d'une alarme.

Avant de commencer

Activez la journalisation des événements liés aux AWS AppConfig données dans AWS CloudTrail. La procédure suivante décrit comment activer la journalisation des métriques pour un suivi AWS AppConfig existant dans CloudTrail. Pour plus d'informations sur la façon d'activer la CloudTrail journalisation des appels du forfait de AWS AppConfig données, consultezAWS AppConfig événements de données dans CloudTrail.

Utilisez la procédure suivante pour permettre à CloudWatch Logs de consigner les métriques relatives aux appels vers le plan de AWS AppConfig données.

Pour permettre à CloudWatch Logs de consigner les métriques des appels vers le plan AWS AppConfig de données

1. Ouvrez la CloudTrail console à l'adresse https://console.aws.amazon.com/cloudtrail/.
2. Sur le tableau de bord, choisissez votre AWS AppConfig parcours.
3. Dans la section CloudWatch Journaux, choisissez Modifier.
4. Choisissez Enabled (Activé).
5. Pour Nom du groupe de journaux, laissez le nom par défaut ou saisissez-en un. Notez le nom. Vous choisirez le groupe de journaux dans la console CloudWatch Logs ultérieurement.
6. Pour Nom du rôle (Role name), saisissez un nom.
7. Sélectionnez Enregistrer les modifications.

Utilisez la procédure suivante pour créer une métrique et un filtre de métriques pour AWS AppConfig in CloudWatch Logs. La procédure décrit comment créer un filtre métrique pour les appels par `operation` et (éventuellement) pour les appels par `operation` et`Amazon Resource Name (ARN)`.

Pour créer une métrique et un filtre de métriques pour AWS AppConfig in CloudWatch Logs

1. Ouvrez la CloudWatch console à l'adressehttps://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation de gauche, choisissez Logs (Journaux), puis Log groups (Groupes de journaux).
3. Cochez la case à côté du groupe de AWS AppConfig journaux.
4. Choisissez Actions, puis Create metric filter (Créer un filtre de métrique).
5. Dans Nom du filtre, entrez un nom.
6. Pour Modèle de filtre, entrez ce qui suit :

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

7. (Facultatif) Dans la section Modèle de test, choisissez votre groupe de journaux dans la liste Sélectionnez les données de journal à tester. Si aucun appel CloudTrail n'a été enregistré, vous pouvez ignorer cette étape.
8. Choisissez Suivant.

9. Pour l'espace de noms métrique, entrez**`AWS AppConfig`**.
10. Pour Metric name (Nom de métrique), saisissez **`Calls`**.
11. Pour Valeur de la métrique, saisissez **`1`**.
12. Ignorez la valeur par défaut et l'unité.
13. Dans le champ Nom de la dimension, entrez**`operation`**.
14. Pour Valeur de dimension, entrez**`$.eventName`**.

    (Facultatif) Vous pouvez saisir une deuxième dimension qui inclut l'Amazon Resource Name (ARN) effectuant l'appel. Pour ajouter une deuxième dimension, saisissez le nom de la dimension**`resource`**. Pour Valeur de dimension, entrez**`$.resources[0].ARN`**.

    Choisissez Suivant.
15. Passez en revue les détails du filtre et créez un filtre métrique.

(Facultatif) Vous pouvez répéter cette procédure pour créer un nouveau filtre métrique pour un code d'erreur spécifique tel que AccessDenied. Si c'est le cas, entrez les informations suivantes :

1. Dans Nom du filtre, entrez un nom.
2. Pour Modèle de filtre, entrez ce qui suit :

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

   Par exemple

   ```
   { $.errorCode = "AccessDenied" }
   ```
3. Pour l'espace de noms métrique, entrez**`AWS AppConfig`**.
4. Pour Metric name (Nom de métrique), saisissez **`Errors`**.
5. Pour Valeur de la métrique, saisissez **`1`**.
6. Pour Valeur par défaut, entrez un zéro (0).
7. Ignorez l'unité, les dimensions et les alarmes.

Après avoir CloudTrail enregistré les appels d'API, vous pouvez consulter les métriques dans CloudWatch. Pour plus d'informations, consultez la section Affichage de vos statistiques et de vos journaux dans la console dans le guide de CloudWatch l'utilisateur Amazon. Pour plus d'informations

sur la façon de localiser une métrique que vous avez créée, voir Rechercher les métriques disponibles.

> **Note**
>
> Si vous configurez la métrique d'erreur sans dimension, comme décrit ici, vous pouvez consulter ces métriques sur la page Mesures sans dimension.

Création d'une alarme pour une CloudWatch métrique

Après avoir créé des métriques, vous pouvez créer des alarmes métriques dans CloudWatch. Par exemple, vous pouvez créer une alarme pour la métrique des AWS AppConfig appels que vous avez créée lors de la procédure précédente. Plus précisément, vous pouvez créer une alarme pour les appels à l'action d' AWS AppConfig `StartConfigurationSession`API qui dépassent un seuil. Pour plus d'informations sur la création d'une alarme pour une métrique, consultez la section Création CloudWatch d'une alarme basée sur un seuil statique dans le guide de CloudWatch l'utilisateur Amazon. Pour plus d'informations sur les limites par défaut pour les appels au plan de AWS AppConfig données, consultez la section Limites par défaut du plan de données dans le Référence générale d'Amazon Web Services.

Surveillance des déploiements pour une annulation automatique

Au cours d'un déploiement, vous pouvez atténuer les situations dans lesquelles des données de configuration mal formées ou incorrectes sont à l'origine d'erreurs dans votre application en combinant des stratégies de AWS AppConfig déploiement et des annulations automatiques basées sur les alarmes Amazon CloudWatch . Une fois configurée, si une ou plusieurs CloudWatch alarmes passent à l'`INSUFFICIENT_DATA`état `ALARM` OR pendant un déploiement, vos données de configuration AWS AppConfig sont automatiquement rétablies à la version précédente, évitant ainsi les pannes ou les erreurs des applications.

> **Note**
>
> Un déploiement n'est pas automatiquement annulé si des actions ont été désactivées dans une CloudWatch alarme associée.

Vous pouvez désactiver et activer les alarmes à l'aide des actions DisableAlarmActionsset de l'EnableAlarmActionsAPI ou disable-alarm-actionsdes enable-alarm-actionscommandes et du AWS CLI.

Vous pouvez également annuler une configuration en appelant l'opération StopDeploymentAPI alors qu'un déploiement est toujours en cours.

Important

Pour les déploiements réussis, il est AWS AppConfig également possible de rétablir les données de configuration à une version précédente en utilisant le `AllowRevert` paramètre avec l'opération d'StopDeploymentAPI. Pour certains clients, le retour à une configuration précédente après un déploiement réussi garantit que les données seront les mêmes qu'avant le déploiement. Le retour en arrière ignore également les moniteurs d'alarme, ce qui peut empêcher la progression d'une application en cas d'urgence. Pour de plus amples informations, veuillez consulter Rétablir une configuration.

Pour configurer les annulations automatiques, vous devez spécifier le nom de ressource Amazon (ARN) d'une ou de plusieurs CloudWatch métriques dans le champ des CloudWatch alarmes lorsque vous créez (ou modifiez) un AWS AppConfig environnement. Pour de plus amples informations, veuillez consulter Création d'environnements pour votre application dans AWS AppConfig.

Note

Si vous utilisez une solution de surveillance tierce (Datadog, par exemple), vous pouvez créer une AWS AppConfig extension qui vérifie la présence d'alarmes au point `AT_DEPLOYMENT_TICK` d'action et, à titre de garde-fou, annule le déploiement s'il déclenche une alarme. Pour plus d'informations sur les AWS AppConfig extensions, consultezExtension des AWS AppConfig workflows à l'aide d'extensions. Pour plus d'informations sur les extensions personnalisées, consultezProcédure pas à pas : création d'extensions personnalisées AWS AppConfig. Pour consulter un exemple de code d'une AWS AppConfig extension qui utilise le point `AT_DEPLOYMENT_TICK` d'action pour s'intégrer à Datadog, consultez aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub

Mesures recommandées à surveiller pour une annulation automatique

Les indicateurs que vous choisissez de surveiller dépendent du matériel et des logiciels utilisés par vos applications. AWS AppConfig les clients surveillent souvent les indicateurs suivants. Pour obtenir la liste complète des métriques recommandées groupées par ordre Service AWS, consultez la section Alarmes recommandées dans le guide de CloudWatch l'utilisateur Amazon.

Après avoir déterminé les métriques que vous souhaitez surveiller, utilisez-les CloudWatch pour configurer les alarmes. Pour plus d'informations, consultez la section Utilisation des CloudWatch alarmes Amazon.

Service	Métrique	Détails
Amazon API Gateway	4 XXError	Cette alarme détecte un taux élevé d'erreurs côté client. Cela peut indiquer un problème dans les paramètre s d'autorisation ou de requête client. Cela peut également signifier qu'une ressource a été supprimée ou qu'un client en demande une qui n'existe pas. Envisagez d'activer Amazon CloudWatch Logs et de vérifier l'absence d'erreurs susceptibles d'être à l'origine des erreurs 4XX. En outre, pensez à activer CloudWatc h les métriques détaillées pour afficher cette métrique par ressource et par méthode et affiner la source des erreurs. Des erreurs peuvent également être causées par le dépassement de la limite configurée.

Service	Métrique	Détails
Amazon API Gateway	5 XXError	Cette alarme permet de détecter un taux élevé d'erreurs côté serveur. Cela peut indiquer qu'il y a un problème sur le backend de l'API, le réseau ou l'intégration entre la passerelle d'API et l'API du backend.
Amazon API Gateway	Latence	Cette alarme détecte une latence élevée dans une phase. Recherchez la valeur de la métrique `IntegrationLatency` pour vérifier la latence du backend de l'API. Si les deux métriques sont globalement alignées, le backend de l'API est à l'origine d'une latence plus élevée et vous devriez examiner les problèmes à ce niveau. Envisagez également d'activer CloudWatch les journaux et de vérifier les erreurs susceptibles d'être à l'origine de cette latence élevée.

Service	Métrique	Détails
Amazon EC2 Auto Scaling	GroupInServiceCapacity	Cette alarme permet de détecter lorsque la capacité du groupe est inférieure à la capacité souhaitée pour votre charge de travail. Pour résoudre le problème, vérifiez vos activités de dimension nement pour détecter les échecs de lancement et confirmez que la configura tion de capacité souhaitée est correcte.
Amazon EC2	CPUUtilization	Cette alarme permet de surveiller l'utilisation du processeur d'une EC2 instance. Selon l'applica tion, des niveaux d'utilisation élevés et constants peuvent être normaux. Mais si les performances sont dégradées et que l'application n'est pas limitée par les E/S du disque, la mémoire ou les ressource s réseau, un processeur au maximum peut indiquer un goulot d'étranglement des ressources ou des problèmes de performance de l'applica tion.

Service	Métrique	Détails
Amazon ECS	CPUReservation	Cette alarme vous aide à détecter une réserve de CPU élevée dans le cluster ECS. Une réserve de processeur élevée peut indiquer que le cluster est à court d'unités enregistrées CPUs pour la tâche.
Amazon ECS	HTTPCode_TARGET_5XX_Count	Cette alarme vous aide à détecter un nombre élevé d'erreurs côté serveur pour le service ECS. Cela peut indiquer que des erreurs empêchent le serveur de répondre aux requêtes.
Amazon EKS avec Container Insights	node_cpu_utilization	Cette alarme permet de détecter une utilisation élevée du processeur dans les nœuds de travail du cluster Amazon EKS. Si le taux d'utilisation est constamment élevé, cela peut indiquer la nécessité de remplacer vos composants master par des instances dotées d'un processeur plus puissant ou de mettre à l'échelle le système horizontalement.

Service	Métrique	Détails
Amazon EKS avec Container Insights	node_memory_utilization	Cette alarme permet de détecter une utilisation élevée de la mémoire dans les nœuds de travail du cluster Amazon EKS. Si le taux d'utilisation est constamment élevé, cela peut indiquer la nécessité de mettre à l'échelle le nombre de réplicas de pods ou d'optimiser votre application.
Amazon EKS avec Container Insights	pod_cpu_utilization_over_pod_limit	Cette alarme permet de détecter une utilisation élevée du processeur dans les pods du cluster Amazon EKS. Si le taux d'utilisation est constamment élevé, cela peut indiquer la nécessité d'augmenter la limite du processeur pour le pod concerné.
Amazon EKS avec Container Insights	pod_memory_utilization_over_pod_limit	Cette alarme permet de détecter une utilisation élevée du processeur dans les pods du cluster Amazon EKS. Si le taux d'utilisation est constamment élevé, cela peut indiquer la nécessité d'augmenter la limite du processeur pour le pod concerné.

Service	Métrique	Détails
AWS Lambda	Erreurs	Cette alarme détecte un nombre élevé d'erreurs. Les erreurs de fonction incluent les exceptions levées par votre code et par l'environnement d'exécution Lambda.
AWS Lambda	Throttles	Cette alarme détecte un nombre élevé de demandes d'invocation limitées. La limitation se produit lorsqu'aucune simultanéité n'est disponible pour une augmentation.
Informations sur Lambda	memory_utilization	Cette alarme est utilisée pour détecter si l'utilisation de la mémoire d'une fonction lambda se rapproche de la limite configurée.
Amazon S3	4xxErrors	Cette alarme nous permet de signaler le nombre total de codes d'état d'erreur 4xx créés en réponse aux demandes des clients. 403 codes d'erreur peuvent indiquer une politique IAM incorrecte, et 404 codes d'erreur peuvent indiquer un mauvais comportement de l'application client, par exemple.

Service	Métrique	Détails
Amazon S3	5xxErrors	Cette alarme vous permet de détecter un grand nombre d'erreurs côté serveur. Ces erreurs indiquent qu'un client a émis une requête que le serveur n'a pas pu traiter. Cela peut vous aider à établir une corrélation entre le problème auquel votre application est confrontée à cause de S3.

AWS AppConfig Historique du document du guide de l'utilisateur

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version de AWS AppConfig.

Version actuelle de l'API : 2019-10-09

Modification	Description	Date
IPv6 soutien	Tous sont AWS AppConfig APIs désormais entièrement pris en charge IPv4 et IPv6 appels. Pour plus d'informations, consultez la section Comprendre IPv6 le support.	23 avril 2025
Nouveau sujet : Sauvegarde d'une version précédente d'un indicateur de fonctionnalité dans une nouvelle version	Lorsque vous mettez à jour un indicateur de fonctionnalité, vos modifications AWS AppConfig sont automatiquement enregistrées dans une nouvelle version. Si vous souhaitez utiliser une version précédente de l'indicateur de fonctionnalité, vous devez la copier dans une version brouillon, puis l'enregistrer. Vous ne pouvez pas modifier et enregistrer les modifications apportées à une version précédente d'un drapeau sans l'enregistrer dans une nouvelle version. Pour plus d'informations, voir Enregistrer une version précédente d'un	15 avril 2025

	indicateur de fonctionnalité dans une nouvelle version.	
Nouveau sujet : Exemples d'indicateurs de fonctionnalités pour le mode de développement local de l'AWS AppConfig agent	AWS AppConfig L'agent prend en charge un mode de développement local. Si vous activez le mode de développement local, l'agent lit les données de configuration depuis un répertoire spécifique sur le disque. Il ne récupère pas les données de configuration à partir de AWS AppConfig. Pour vous aider à mieux comprendre comment utiliser le mode de développement local, ce guide contient désormais une rubrique contenant des exemples d'indicateurs de fonctionnalités. Pour plus d'informations, consultez la section Exemples d'indicateurs de fonctionnalité pour le mode de développement local de l'AWS AppConfig agent.	18 février 2025

Nouveau sujet : Création d'un profil de configuration pour les sources de données non natives	Cette rubrique décrit le processus de haut niveau d'utilisation d'une AWS AppConfig extension pour récupérer des données de configuration à partir de sources qui ne sont pas prises en charge de manière native, notamment d'autres AWS services tels qu'Amazon RDS et Amazon DynamoDB, ainsi que de sources tierces telles que, ou un dépôt local. GitHub GitLab Pour plus d'informations, voir Création d'un profil de configuration pour les sources de données non natives	19 décembre 2024
Sujet mis à jour : correction de l'expression régulière dans la référence du type d'indicateurs de fonctionnalité	Le schéma json dans la référence du type d'indicateur de fonctionnalité affichait précédemment le modèle regex suivant à divers endroits :`"^[a-z][a-zA-Z\\d-_]{0,63}$"` . Le modèle de regex correct est`"^[a-z][a-zA-Z\\d_-]{0,63}$"` . Le trait d'union apparaît après le trait de soulignement. Pour plus d'informations, consultez Comprendre la référence de type pour AWS. AppConfig. FeatureFlags	18 décembre 2024

Rubriques mises à jour : ajout d'exemples de variables d'environnement	Les tableaux décrivant les variables d'environnement dans les rubriques suivantes ont été mis à jour pour inclure des exemples : • (Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon ECS et Amazon EKS • (Facultatif) Utilisation de variables d'environnement pour configurer AWS AppConfig l'agent pour Amazon EC2 • Configuration de l'extension AWS AppConfig Agent Lambda	12 décembre 2024
Nouvelle section : Comprendre l'opérateur de division	Une nouvelle section utilise des exemples pour expliquer comment l'`split`opérateur fonctionne pour une règle d'indicateur de fonctionnalité à plusieurs variantes. Pour plus d'informations, voir Comprendre les règles relatives aux indicateurs de fonctionnalités à variantes multiples.	22 novembre 2024

Nouveau point d'action d'extension : AT_DEPLOYMENT_TICK	AWS AppConfig a lancé un nouveau point d'action pour les utilisateurs qui créent des extensions personnalisées. Le point `AT_DEPLOYMENT_TICK` d'action prend en charge l'intégration de la surveillance par des tiers. `AT_DEPLOYMENT_TICK` est invoqué lors de l'orchestration du traitement du déploiement de la configuration. Si vous utilisez une solution de surveillance tierce (Datadog, par exemple), vous pouvez créer une AWS AppConfig extension qui vérifie la présence d'alarmes au point `AT_DEPLOYMENT_TICK` d'action et, à titre de garde-fou, annule le déploiement s'il déclenche une alarme. Pour plus d'informations sur les AWS AppConfig extensions, consultez la section Extension des AWS AppConfig flux de travail à l'aide d'extensions. Pour plus d'informations sur les extensions personnalisées, voir Procédure pas à pas : création d' AWS AppConfig extensions personnalisées. Pour consulter un exemple de code d'une AWS AppConfig extension qui utilise le point `AT_DEPLOYMENT_TICK`	22 novembre 2024

	d'action pour s'intégrer à Datadog, consultez aws-samples/-for-datadog on. aws-appconfig-tick-extn GitHub	
Nouveau sujet : considérations relatives à l'utilisation des AWS AppConfig appareils mobiles	Une nouvelle rubrique de ce guide décrit les points importants à prendre en compte lors de l'utilisation des indicateurs de AWS AppConfig fonctionnalité sur les appareils mobiles. Pour plus d'informations, consultez la section Considérations relatives à l'utilisation des AWS AppConfig appareils mobiles.	21 novembre 2024
Nouvelle fonctionnalité : protection contre la AWS AppConfig suppression	AWS AppConfig fournit désormais un paramètre de compte pour empêcher les utilisateurs de supprimer par inadvertance des environnements et des profils de configuration utilisés activement. Pour plus d'informations, consultez la section Configuration de la protection contre les AWS AppConfig suppressions.	28 août 2024

Nouvelle version de l'extension AWS AppConfig Agent Lambda	L'agent a été mis à jour avec des améliorations mineures et des corrections de bogues. Pour consulter les nouveaux Amazon Resource Names (ARNs) associés à l'extension, consultez la section Versions disponibles de l'extension AWS AppConfig Agent Lambda.	9 août 2024
Nouveaux exemples de code pour récupérer les variantes de drapeaux	Pour plus d'informations, consultez la section Utilisation de l' AWS AppConfig agent pour récupérer un indicateur de fonctionnalité avec des variantes.	9 août 2024
Nouvelle version de l'extension AWS AppConfig Agent Lambda	L'agent a été mis à jour pour prendre en charge les cibles, les variantes et les divisions des indicateurs de fonctionnalités. Pour consulter les nouveaux Amazon Resource Names (ARNs) associés à l'extension, consultez la section Versions disponibles de l'extension AWS AppConfig Agent Lambda.	23 juillet 2024

Nouvelle fonctionnalité : drapeaux de fonctionnalités à variantes multiples	Les indicateurs de fonctionnalités à variantes multiples vous permettent de définir un ensemble de valeurs d'indicateurs possibles à renvoyer pour une demande. Vous pouvez également configurer différents statuts (activé ou désactivé) pour les indicateurs à variantes multiples. Lorsque vous demandez un indicateur configuré avec des variantes, votre application fournit un contexte qui est AWS AppConfig évalué par rapport à un ensemble de règles définies par l'utilisateur. En fonction du contexte spécifié dans la demande et des règles définies pour la variante, AWS AppConfig renvoie différentes valeurs d'indicateur à l'application. Pour plus d'informations, consultez la section Création d'indicateurs de fonctionnalités à variantes multiples.	23 juillet 2024

Nouvelle version de l'extension AWS AppConfig Agent Lambda	L'agent a été mis à jour avec des améliorations mineures et des corrections de bogues. Pour consulter les nouveaux Amazon Resource Names (ARNs) associés à l'extension, consultez la section Versions disponibles de l'extension AWS AppConfig Agent Lambda.	28 février 2024
AWS AppConfig exemples d'extensions personnalisées	La rubrique Procédure pas à pas : création d'AWS AppConfig extensions personnalisées inclut désormais des liens vers les exemples d'extensions suivants sur GitHub : • Exemple d'extension qui empêche les déploiements avec un calendrier de `blocked day` moratoire à l'aide de Systems Manager Change Calendar • Exemple d'extension qui empêche les secrets de s'infiltrer dans les données de configuration à l'aide de git-secrets • Exemple d'extension empêchant la fuite d'informations personnelles (PII) dans les données de configuration à l'aide d'Amazon Comprehend	28 février 2024

Nouveau sujet : Journalisation des appels AWS AppConfig d'API à l'aide AWS CloudTrail	AWS AppConfig est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans AWS AppConfig. CloudTrail capture tous les appels d'API AWS AppConfig sous forme d'événements. Cette nouvelle rubrique fournit un contenu AWS AppConfig spécifique plutôt que des liens vers le contenu correspondant dans le guide de l'AWS Systems Manager utilisateur. Pour plus d'informations, consultez la section Journalisation des appels AWS AppConfig d'API à l'aide de AWS CloudTrail.	18 janvier 2024

AWS AppConfig prend désormais en charge AWS PrivateLink	Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et. AWS AppConfig Vous pouvez y accéder AWS AppConfig comme s'il se trouvait dans votre VPC, sans utiliser de passerelle Internet, de périphérique NAT, de connexion VPN ou AWS Direct Connect de connexion . Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour y accéder. AWS AppConfig Pour plus d'informations, consultez Accès à AWS AppConfig l'aide d'un point de terminaison d'interface (AWS PrivateLink).	6 décembre 2023

Fonctionnalités supplémentaires de récupération de l' AWS AppConfig agent et nouveau mode de développement local	AWS AppConfig L'agent propose les fonctionnalités supplémentaires suivantes pour vous aider à récupérer les configurations de vos applications. Fonctionnalités de récupération supplémentaires • Récupération multi-comptes : utilisez l' AWS AppConfig agent d'un compte principal ou d'une extraction Compte AWS pour récupérer les données de configuration de plusieurs comptes fournisseurs. • Écrire une copie de configuration sur le disque : utilisez l' AWS AppConfig agent pour écrire les données de configuration sur le disque. Cette fonctionnalité permet aux clients utilisant des applications qui lisent les données de configuration sur disque de s'y intégrer AWS AppConfig. Note L'écriture de la configuration sur	1er décembre 2023

disque n'est pas conçue comme une fonctionnalité de sauvegarde de configuration. AWS AppConfig L'agent ne lit pas les fichiers de configuration copiés sur le disque. Si vous souhaitez sauvegarder des configurations sur disque, consultez les variables d'`PRELOAD_BACKUP` environnement `BACKUP_DIRECTORY` et relatives à l'utilisation de l'AWS AppConfig agent avec Amazon EC2 ou à l'utilisation de l'AWS AppConfig agent avec Amazon ECS et Amazon EKS.

Mode de développement local

AWS AppConfig L'agent prend en charge un mode de développement local. Si vous activez le mode de développement local, l'agent lit les données de configuration depuis un répertoire spécifique sur le disque. Il ne récupère pas les données

	de configuration à partir de AWS AppConfig. Vous pouvez simuler des déploiements de configuration en mettant à jour les fichiers dans le répertoire spécifié. Nous recommandons le mode de développement local pour les cas d'utilisation suivants : • Testez différentes versions de configuration avant de les déployer à l'aide de AWS AppConfig. • Testez différentes options de configuration pour une nouvelle fonctionnalité avant de valider les modifications dans votre référentiel de code. • Testez différents scénarios de configuration pour vérifier qu'ils fonctionnent comme prévu.	
Nouveau sujet relatif aux exemples de code	Ajout d'une nouvelle rubrique sur les exemples de code à ce guide. Cette rubrique inclut des exemples en Java, Python et permet JavaScript d'effectuer six actions courantes AWS AppConfig par programmation.	17 novembre 2023

Table des matières révisée pour mieux refléter le AWS AppConfig flux de travail	Le contenu de ce guide de l'utilisateur est désormais regroupé sous les rubriques Création, déploiement, récupération et extension des flux de travail. Cette organisation reflète mieux le flux de travail d'utilisation AWS AppConfig et vise à rendre le contenu plus facile à découvrir .	7 novembre 2023
Référence de charge utile ajoutée	La rubrique Création d'une fonction Lambda pour une AWS AppConfig extension personnalisée inclut désormais une référence de charge utile de demande et de réponse.	7 novembre 2023
Nouvelle stratégie de déploiement AWS prédéfinie	AWS AppConfig propose et recommande désormais la stratégie de déploiement `AppConfig.Linear20PercentEvery6Minutes` prédéfinie. Pour plus d'informations, consultez la section Stratégies de déploiement prédéfinies.	11 août 2023

AWS AppConfig intégration avec Amazon EC2	Vous pouvez intégrer AWS AppConfig des applications exécutées sur vos instances Linux Amazon Elastic Compute Cloud (Amazon EC2) à l'aide de l' AWS AppConfig agent. L'agent prend en charge x86_64 et les architectures Amazon ARM64 . EC2 Pour plus d'informations, consultez la section AWS AppConfig Intégration à Amazon EC2.	20 juillet 2023
AWS CloudFormation support pour de nouvelles AWS AppConfig ressources et exemple d'indicateur de fonctionnalité	AWS CloudFormation prend désormais en charge les AWS::AppConfig::ExtensionAssociationressources AWS::AppConfig::Extensionet les ressources nécessaires pour vous aider à démarrer avec les AWS AppConfig extensions. Les ressources AWS::AppConfig::ConfigurationProfileet AWS::AppConfig::HostedConfigurationVersion incluent désormais un exemple de création d'un profil de configuration avec indicateur de fonctionnalité dans le magasin de configuration AWS AppConfig hébergé.	12 avril 2023

AWS AppConfig intégration avec AWS Secrets Manager	AWS AppConfig s'intègre à AWS Secrets Manager. Secrets Manager vous aide à chiffrer, stocker et récupérer en toute sécurité les informations d'identification pour vos bases de données et autres services. Au lieu de coder en dur les informations d'identification dans vos applications, vous pouvez appeler Secrets Manager pour récupérer vos informations d'identification chaque fois que vous en avez besoin. Secrets Manager vous aide à protéger l'accès à vos ressources informatiques et à vos données en vous permettant d'alterner et de gérer l'accès à vos secrets. Lorsque vous créez un profil de configuration libre, vous pouvez choisir Secrets Manager en tant que source de vos données de configuration. Vous devez intégrer Secrets Manager et créer un secret avant de créer le profil de configuration. Pour plus d'informations sur Secrets Manager, consultez Qu'est-ce que c'est AWS Secrets Manager ? dans le guide de AWS Secrets Manager l'utilisateur. Pour plus d'informations	2 février 2023

sur la création d'un profil de configuration, consultez la section Création d'un profil de configuration de forme libre.

AWS AppConfig intégration avec Amazon ECS et Amazon EKS	Vous pouvez intégrer Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS) à l' AWS AppConfig aide de l'agent. AWS AppConfig L'agent fonctionne comme un conteneur annexe s'exécutant parallèlement à vos applicati ons de conteneur Amazon ECS et Amazon EKS. L'agent améliore le traitement et la gestion des applications conteneurisées de la manière suivante : • L'agent appelle AWS AppConfig en votre nom en utilisant un rôle AWS Identity and Access Management (IAM) et en gérant un cache local de données de configuration. En extrayant les données de configuration du cache local, votre application nécessite moins de mises à jour de code pour gérer les données de configuration, récupère les données de configuration en quelques millisecondes et n'est pas affectée par les problèmes réseau susceptibles de	2 décembre 2022

perturber les appels pour ces données.

- L'agent propose une expérience native pour récupérer et résoudre les indicateurs de AWS AppConfig fonctionnalités.
- Prêt à l'emploi, l'agent fournit les meilleures pratiques en matière de stratégies de mise en cache, d'intervalles d'interrogation et de disponibilité des données de configuration locales, tout en suivant les jetons de configuration nécessaires pour les appels de service suivants.
- Lorsqu'il s'exécute en arrière-plan, l'agent interroge régulièrement le plan de AWS AppConfig données pour les mises à jour des données de configuration. Votre application conteneurisée peut récupérer les données en se connectant à localhost sur le port 2772 (une valeur de port par défaut personnalisable) et en appelant HTTP GET pour récupérer les données.
- L' AWS AppConfig agent met à jour les données de configuration de vos

conteneurs sans avoir à redémarrer ou à recycler ces conteneurs.

Pour plus d'informations, consultez la section AWS AppConfig Intégration avec Amazon ECS et Amazon EKS.

Nouvelle extension : AWS AppConfig extension pour CloudWatch Evidently	Vous pouvez utiliser Amazon CloudWatch Evidently pour valider de nouvelles fonctionn alités en toute sécurité en les proposant à un pourcentage spécifique de vos utilisateurs pendant que vous déployez la fonctionnalité. Vous pouvez surveiller les performances de la nouvelle fonction afin de décider du moment où vous souhaitez augmenter le trafic vers vos utilisateurs. Ainsi, vous pouvez réduire les risques et identifier les conséquences involontaires avant de lancer pleinemen t la fonction. Vous pouvez également effectuer des A/B expériences pour prendre des décisions de conception des fonctionnalités sur la base de preuves et de données. L' AWS AppConfig extension pour CloudWatch Evidently permet à votre application d'attribuer des variations aux sessions utilisateur localemen t plutôt qu'en appelant l'EvaluateFeatureopération . Une session locale atténue les risques de latence et de disponibilité associés à un appel d'API. Pour plus d'informations sur la configura	13 septembre 2022

	tion et l'utilisation de l'extensi on, consultez la section Effectuer des lancements et A/B des expériences avec CloudWatch Evidently dans le guide de l' CloudWatch utilisateur Amazon.	
Obsolète de l'action d'API `GetConfiguration`	Le 18 novembre 2021, AWS AppConfig a lancé un nouveau service de plan de données. Ce service remplace le processus précédent de récupération des données de configuration à l'aide de l'action `GetConfiguration` API. Le service de plan de données utilise deux nouvelles actions d'API, StartConf igurationSessionet GetLatest Configuration. Le service de plan de données utilise également de nouveaux points de terminaison. Pour plus d'informations, reportez-vous à la section À propos du service de plan de AWS AppConfig données.	13 septembre 2022

Nouvelle version de l'extension AWS AppConfig Agent Lambda	La version 2.0.122 de l'extension Agent AWS AppConfig Lambda est désormais disponible. La nouvelle extension utilise différents noms de ressources Amazon (ARNs). Pour plus d'informations, consultez les notes de mise à jour de l'extension AWS AppConfig Agent Lambda.	23 août 2022
Lancement des AWS AppConfig extensions	Une extension augmente votre capacité à injecter de la logique ou du comportement à différents moments du AWS AppConfig flux de travail de création ou de déploiement d'une configuration. Vous pouvez utiliser les extensions AWS-authored ou créer les vôtres. Pour plus d'informations, consultez la section Utilisation des AWS AppConfig extensions.	12 juillet 2022

Nouvelle version de l'extension AWS AppConfig Agent Lambda	La version 2.0.58 de l'extension Agent AWS AppConfig Lambda est désormais disponible. La nouvelle extension utilise différents noms de ressources Amazon (ARNs). Pour plus d'informations, consultez la section Versions disponibles de l'extension AWS AppConfig Lambda.	3 mai 2022
AWS AppConfig intégration avec Atlassian Jira	L'intégration à Atlassian Jira permet AWS AppConfig de créer et de mettre à jour des problèmes dans la console Atlassian chaque fois que vous modifiez un indicateur de fonctionnalité dans votre formulaire spécifié. Compte AWS Région AWS Chaque problème de Jira inclut le nom du drapeau, l'ID de l'application, l'ID du profil de configuration et les valeurs du drapeau. Une fois que vous avez mis à jour, enregistré et déployé vos modifications d'indicateur, Jira met à jour les problèmes existants avec les détails de la modification. Pour plus d'informations, consultez la section AWS AppConfig Intégration à Atlassian Jira.	7 avril 2022

Disponibilité générale des indicateurs de fonctionnalité et prise en charge de l'extension Lambda pour les processeurs ARM64 (Graviton2)	Avec les indicateurs de AWS AppConfig fonctionnalité, vous pouvez développer une nouvelle fonctionnalité et la déployer en production tout en la cachant aux utilisate urs. Vous commencez par ajouter l'indicateur en AWS AppConfig tant que données de configuration. Une fois que la fonctionnalité est prête à être publiée, vous pouvez mettre à jour les données de configuration du drapeau sans déployer de code. Cette fonctionnalité améliore la sécurité de votre environne ment de développement car vous n'avez pas besoin de déployer de nouveau code pour lancer la fonctionnalité. Pour plus d'informations, consultez la section Création d'un profil de configuration d'indicateur de fonctionnalité. La disponibilité générale des indicateurs de fonctionnalité AWS AppConfig inclut les améliorations suivantes : • La console inclut une option permettant de désigner un drapeau comme indicateur à court terme. Vous pouvez filtrer et trier la liste des	15 mars 2022

drapeaux sur les drapeaux à court terme.

- Pour les clients utilisant des indicateurs de fonctionn alité AWS Lambda, la nouvelle extension Lambda vous permet d'appeler des indicateurs de fonctionn alité individuels à l'aide d'un point de terminaison HTTP. Pour plus d'informations, voir Extraction d'un ou de plusieurs indicateurs à partir d'une configuration d'indicateur de fonctionnalité.

Cette mise à jour prend également en charge les AWS Lambda extensions développées pour les processeurs ARM64 (Graviton2). Pour plus d'informations, voir Versions disponibles de l'extension AWS AppConfig Lambda.

L'action d' GetConfiguration API est obsolète	L'action d'`GetConfiguration` API est obsolète. Les appels destinés à recevoir des données de configuration doivent utiliser le `StartConfigurationSession` et à la `GetLatestConfiguration` APIs place. Pour plus d'informations à ce sujet APIs et sur leur utilisation, consultez la section Récupération de la configuration.	28 janvier 2022
Nouvel ARN de région pour l'extension AWS AppConfig Lambda	AWS AppConfig L'extension Lambda est disponible dans la nouvelle région Asie-Pacifique (Osaka). L'Amazon Resource Name (ARN) est requis pour créer un Lambda dans la région. Pour plus d'informations sur l'ARN de la région Asie-Pacifique (Osaka), consultez la section Ajout de l'extension AWS AppConfig Lambda.	4 mars 2021

AWS AppConfig Extension Lambda	Si vous avez l' AWS AppConfig habitude de gérer les configurations d'une fonction Lambda, nous vous recommandons d'ajouter l'extension Lambda AWS AppConfig . Cette extension inclut les meilleures pratiques qui simplifient l'utilisation AWS AppConfig tout en réduisant les coûts. La réduction des coûts résulte de la diminutio n du nombre d'appels d'API au AWS AppConfig service et, séparément, de la réduction des coûts résultant de la réduction des temps de traitement des fonctions Lambda. Pour plus d'informa tions, consultez la section AWS AppConfig Intégration avec les extensions Lambda.	8 octobre 2020
Nouvelle section	Ajout d'une nouvelle section qui fournit des instructions de configuration AWS AppConfig . Pour plus d'informations, consultez la section Configura tion AWS AppConfig.	30 septembre 2020

Procédures de ligne de commande ajoutées	Les procédures décrites dans ce guide de l'utilisateur incluent désormais des étapes de ligne de commande pour AWS Command Line Interface (AWS CLI) et Tools for Windows PowerShell. Pour plus d'informations, consultez la section Travailler avec AWS AppConfig.	30 septembre 2020
Lancement du guide de AWS AppConfig l'utilisateur	Utilisez AWS AppConfig un outil dans AWS Systems Manager, pour créer, gérer et déployer rapidement des configurations d'applications. AWS AppConfig prend en charge les déploiements contrôlés vers des applications de toutes tailles et inclut des contrôles de validation et une surveillance intégrés. Vous pouvez l'utiliser AWS AppConfig avec des applications hébergées sur des EC2 instances AWS Lambda, des conteneurs, des applications mobiles ou des appareils IoT.	31 juillet 2020

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.