Manual do usuário

AWS AppConfig

AWS AppConfig: Manual do usuário

Table of Contents

O que é o AWS AppConfig?

Os sinalizadores de atributos e as configurações dinâmicas do AWS AppConfig ajudam os criadores de software a ajustar, com rapidez e segurança, o comportamento do aplicativo em ambientes de produção sem implantações completas de código. O AWS AppConfig acelera a frequência de lançamento de softwares, melhora a resiliência do aplicativo e ajuda a resolver problemas emergentes com mais rapidez. Com sinalizadores de atributos, você pode liberar gradualmente novos recursos para os usuários e medir o impacto dessas mudanças antes de implantar totalmente os novos recursos para todos os usuários. Com sinalizadores operacionais e configurações dinâmicas, você pode atualizar listas de bloqueio, listas de permissões, limites de controle de utilização, verbosidade de registros em log e realizar outros ajustes operacionais para responder rapidamente a problemas nos ambientes de produção.

> **Note**
>
> O AWS AppConfig é um recurso do AWS Systems Manager.

Melhore a eficiência e libere as alterações mais rapidamente

O uso de sinalizadores de atributos com novos recursos acelera o processo de liberação de alterações nos ambientes de produção. Em vez de depender de ramificações de desenvolvimento de longa duração que exigem mesclagens complicadas antes do lançamento, os sinalizadores de atributos permitem escrever softwares usando desenvolvimento baseado em troncos. Os sinalizadores de atributos permitem implantar com segurança o código de pré-lançamento em um pipeline de CI/CD que fica oculto para os usuários. Quando estiver pronto para lançar as alterações, você poderá atualizar o sinalizador de atributos sem implantar um novo código. Depois do lançamento, o sinalizador ainda poderá funcionar como uma chave de bloqueio para desativar um novo atributo ou recursos sem a necessidade de reverter a implantação do código.

Evite alterações ou falhas não intencionais com atributos de segurança integrados

O AWS AppConfig oferece os seguintes atributos de segurança para ajudá-lo a evitar a ativação de sinalizadores de atributos ou a atualização de dados de configuração que possam causar falhas no aplicativo.

- Validadores: o validador garante que os dados de configuração estejam sintática e semanticamente corretos antes de implantar as alterações nos ambientes de produção.

- Estratégias de implantação: uma estratégia de implantação permite liberar lentamente as alterações nos ambientes de produção em questão de minutos ou horas.
- Monitoramento e reversão automática: AWS AppConfig integra-se ao Amazon CloudWatch para monitorar alterações nos aplicativos. Se o aplicativo perder a integridade por causa de uma alteração incorreta na configuração e essa alteração disparar um alarme no CloudWatch, o AWS AppConfig reverte automaticamente a alteração para minimizar o impacto sobre os usuários do aplicativo.

Implantações seguras e escaláveis para sinalizadores de atributos

O AWS AppConfig se integra ao AWS Identity and Access Management (IAM) para fornecer acesso refinado e baseado em funções ao serviço. O AWS AppConfig também se integra com o AWS Key Management Service (AWS KMS) para criptografia e ao AWS CloudTrail, para auditoria. Antes de serem lançados para clientes externos, todos os controles de segurança do AWS AppConfig foram inicialmente desenvolvidos e validados por clientes internos que usam o serviço em grande escala.

Casos de uso do AWS AppConfig

Apesar de o conteúdo da configuração do aplicativo variar muito de aplicativo para aplicativo, o AWS AppConfig oferece suporte aos seguintes casos de uso, que abrangem um amplo espectro de necessidades dos clientes:

- Sinalizadores e botões de atributos: libere com segurança novos recursos para seus clientes em um ambiente controlado. Reverta instantaneamente as alterações em caso de algum problema.
- Ajuste de aplicativos: introduza cuidadosamente as alterações nos aplicativos e, ao mesmo tempo, teste o impacto dessas mudanças com os usuários em ambientes de produção.
- Lista de permissões ou lista de bloqueio: controle o acesso a atributos premium ou bloqueie instantaneamente usuários específicos sem implantar um novo código.
- Armazenamento de configurações centralizado: mantenha seus dados de configuração organizados e consistentes em todos os workloads. Você pode usar o AWS AppConfig para implantar dados de configuração armazenados no armazenamento de configuração hospedado pelo AWS AppConfig, AWS Secrets Manager, no Systems Manager Parameter Store ou no Amazon S3.

Benefícios do uso do AWS AppConfig

O AWS AppConfig oferece os seguintes benefícios para sua organização:

- Reduza o tempo de inatividade inesperado para seus clientes

  O AWS AppConfig reduz o tempo de inatividade do aplicativo, permitindo que você crie regras para validar sua configuração. Configurações inválidas não podem ser implantadas. O AWS AppConfig oferece as duas seguintes opções para validar configurações:

  - Para validação sintática, você pode usar um esquema JSON. O AWS AppConfig valida a configuração usando o esquema JSON para garantir que as alterações de configuração cumpram os requisitos do aplicativo.
  - Para validação semântica, o AWS AppConfig pode chamar uma função AWS Lambda que você possui para validar os dados em sua configuração.
- Implante rapidamente alterações em um conjunto de destinos

  O AWS AppConfig simplifica a administração de aplicativos em escala implantando alterações de configuração em um local central. O AWS AppConfig oferece suporte a configurações armazenadas no armazenamento de configuração hospedado pelo AWS AppConfig, no Systems Manager Parameter Store, nos documentos do Systems Manager (SSM) e no Amazon S3. Você pode usar o AWS AppConfig com aplicativos hospedados em instâncias do EC2, AWS Lambda, contêineres, aplicativos móveis ou dispositivos de IoT.

  Os destinos não precisam ser configurados com o Systems Manager SSM Agent nem com o perfil de instância do IAM exigido por outros recursos do Systems Manager. Isso significa que o AWS AppConfig funciona com instâncias não gerenciadas.
- Atualizar aplicativos sem interrupções

  O AWS AppConfig implanta alterações de configuração em seus destinos no tempo de execução sem um processo de compilação pesado ou sem colocar seus destinos fora de serviço.
- Controlar a implantação de alterações em seu aplicativo

  Ao implantar alterações de configuração em seus destinos, o AWS AppConfig permite minimizar os riscos usando uma estratégia de implantação. As estratégias de implantação permitem implementar lentamente as alterações de configuração em sua frota. Em caso de algum problema durante a implantação, você poderá reverter a alteração na configuração antes que ela atinja a maioria dos seus hosts.

Como o AWS AppConfig funciona

Esta seção fornece uma descrição geral de como o AWS AppConfig funciona e quais são os primeiros passos.

1. Identificar os valores de configuração no código que você deseja gerenciar na nuvem

   Antes de começar a criar artefatos para o AWS AppConfig, recomendamos identificar os dados de configuração no código que você deseja gerenciar dinamicamente usando o AWS AppConfig. Bons exemplos incluem sinalizadores ou botões de atributos, listas de permissões e bloqueios, verbosidade de registros em log, limites de serviço e regras de controle de utilização, para citar alguns.

   Se seus dados de configuração já existirem na nuvem, você poderá aproveitar os atributos de validação, implantação e extensão do AWS AppConfig para simplificar ainda mais o gerenciamento dos dados de configuração.

2. Criar um namespace para o aplicativo

   Para criar um namespace, é preciso criar um artefato do AWS AppConfig chamado aplicativo. Um aplicativo é simplesmente uma estrutura organizacional, como uma pasta.

3. Criar ambientes

   Para cada aplicativo do AWS AppConfig, defina um ou mais ambientes. Um ambiente é um agrupamento lógico de destinos, como aplicativos em um ambiente `Beta` ou de `Production`, funções AWS Lambda ou contêineres. Também é possível definir ambientes para subcomponentes de aplicativos, como os componentes `Web`, `Mobile` e `Back-end`.

   Você pode configurar alarmes do Amazon CloudWatch para cada ambiente. O sistema monitora os alarmes durante uma implantação de configuração. Se um alarme for acionado, o sistema reverterá a configuração.

4. Criar um perfil de configuração

   Um perfil de configuração inclui, entre outras coisas, um URI que permite que o AWS AppConfig localize seus dados de configuração em seu local armazenado e um tipo de perfil. O AWS AppConfig suporta dois tipos de perfil de configuração: sinalizadores de atributos e configurações de formato livre. Os perfis de configuração dos sinalizadores de atributos armazenam seus dados no armazenamento de configuração hospedado pelo AWS AppConfig, e o URI é simplesmente `hosted`. Para perfis de configuração de formato livre, você pode armazenar seus dados no

armazenamento de configuração hospedado pelo AWS AppConfig ou em qualquer serviço da AWS que se integre ao AWS AppConfig, conforme descrito em Criando um perfil de configuração de formato livre no AWS AppConfig.

Um perfil de configuração também pode incluir validadores opcionais para garantir que seus dados de configuração estejam corretos dos pontos de vista sintático e semântico. O AWS AppConfig executa uma verificação usando os validadores quando você inicia uma implantação. Se algum erro for detectado, a implantação será revertida para os dados de configuração anteriores.

5. Implantar dados de configuração

   Ao criar uma nova implantação, é preciso especificar:

   - O ID do aplicativo
   - O ID do perfil de configuração
   - A versão da configuração
   - O ID do ambiente no qual você deseja implantar os dados de configuração
   - O ID da estratégia de implantação que define a rapidez com que você deseja que as alterações entrem em vigor

   Quando você chama a ação StartDeployment da API, o AWS AppConfig executa as seguintes tarefas:

   1. Recupera os dados de configuração do armazenamento de dados subjacente usando o URI de localização no perfil de configuração.
   2. Verifica se os dados de configuração estão sintática e semanticamente corretos usando os validadores especificados ao criar o perfil de configuração.
   3. Armazena em cache uma cópia dos dados para que estejam prontos para serem recuperados pelo seu aplicativo. Essa cópia em cache é chamada de dados implantados.

6. Recuperar a configuração

   Você pode configurar o AWS AppConfig Agent como host local e fazer com que o agente pesquise o AWS AppConfig em busca de atualizações da configuração. O agente chama as ações StartConfigurationSession e GetLatestConfiguration da API e armazena seus dados de configuração em cache localmente. Para recuperar os dados, seu aplicativo faz uma chamada HTTP para o servidor localhost. AWS AppConfig O agente oferece suporte a vários casos de uso, conforme descrito em Métodos de recuperação simplificados.

Se o AWS AppConfig Agent não for compatível com seu caso de uso, você poderá configurar seu aplicativo para pesquisar o AWS AppConfig em busca de atualizações de configuração chamando diretamente as ações StartConfigurationSession e GetLatestConfiguration da API.

Conceitos básicos do AWS AppConfig

Os recursos a seguir podem ajudar você a trabalhar diretamente com o AWS AppConfig.

Assista a mais vídeos da AWS no Canal da Amazon Web Services no YouTube.

Os seguintes blogs podem ajudá-lo a saber mais sobre o AWS AppConfig e seus recursos:

- Uso de sinalizadores de atributos do AWS AppConfig
- Práticas recomendadas para validar sinalizadores de atributos e dados de configuração do AWS AppConfig

SDKs

Para obter informações sobre SDKs específicos da linguagem do AWS AppConfig, consulte os seguintes recursos:

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

Definição de preços do AWS AppConfig

O preço do AWS AppConfig é pago conforme o uso, com base nos dados de configuração e na recuperação do sinalizador de atributos. Recomendamos usar o AWS AppConfig Agent para ajudar a otimizar custos. Para obter mais informações, consulte Preços do AWS Systems Manager.

AWS AppConfigCotas do

As informações sobre endpoints e cotas de serviço do AWS AppConfig, juntamente com outras cotas do Systems Manager, estão no Referência geral da Amazon Web Services.

> **Note**
>
> Para obter informações sobre cotas para serviços que armazenam configurações do AWS AppConfig, consulte Sobre cotas e limitações do armazenamento de configuração.

Conf AWS AppConfig iguração

Se você ainda não tiver feito isso, inscreva-se Conta da AWS e crie um usuário administrativo.

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e digitar um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como uma prática recomendada de segurança, atribua o acesso administrativo para um usuário e use somente o usuário-raiz para executar tarefas que requerem o acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, digite sua senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Signing in as the root user (Fazer login como usuário-raiz) no Guia do usuário do Início de Sessão da AWS .

2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .

2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso para usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferrament as e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de

Qual usuário precisa de acesso programático?	Para	Por
		usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

(Opcional) Configurar permissões para reversão com base em alarmes CloudWatch

Você pode configurar AWS AppConfig para reverter para uma versão anterior de uma configuração em resposta a um ou mais CloudWatch alarmes da Amazon. Ao configurar uma implantação para responder aos CloudWatch alarmes, você especifica uma função AWS Identity and Access Management (IAM). AWS AppConfig requer essa função para que possa monitorar CloudWatch os alarmes.

> **Note**
>
> O perfil do IAM deve pertencer à sua conta atual. Por padrão, só é AWS AppConfig possível monitorar alarmes de propriedade da conta corrente. Se você quiser configurar AWS AppConfig para reverter implantações em resposta às métricas de uma conta diferente, deverá configurar alarmes entre contas. Para obter mais informações, consulte CloudWatch Console multiregional entre contas no Guia CloudWatch do usuário da Amazon.

Use os procedimentos a seguir para criar uma função do IAM que permita AWS AppConfig a reversão com base em CloudWatch alarmes. Esta seção inclui os seguintes procedimentos.

1. Etapa 1: criar a política de permissão para reversão com base em alarmes CloudWatch
2. Etapa 2: criar a função do IAM para reversão com base em alarmes CloudWatch
3. Etapa 3: Adicionar uma relação de confiança

Etapa 1: criar a política de permissão para reversão com base em alarmes CloudWatch

Use o procedimento a seguir para criar uma política do IAM que dê AWS AppConfig permissão para chamar a ação da `DescribeAlarms` API.

Para criar uma política de permissão do IAM para reversão com base em alarmes CloudWatch

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas e, em seguida, Criar política.
3. Na página Criar política, escolha a guia JSON.
4. Substitua o conteúdo padrão na guia JSON pela política de permissão a seguir e, em seguida, escolha Próximo: Tags.

> **Note**
>
> Para retornar informações sobre alarmes CloudWatch compostos, é necessário atribuir * permissões à operação da DescribeAlarmsAPI, conforme mostrado aqui. Você não pode retornar informações sobre alarmes compostos se `DescribeAlarms` tiver um escopo mais restrito.

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

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

5. Insira tags para essa função e escolha Next: Review (Próximo: revisão).
6. Na página Revisão, insira **`SSMCloudWatchAlarmDiscoveryPolicy`** no campo Nome.
7. Escolha Criar política. O sistema retorna para a página Policies (Políticas).

Etapa 2: criar a função do IAM para reversão com base em alarmes CloudWatch

Use o procedimento a seguir para criar um perfil do IAM e atribuir a ele a política que você criou no procedimento anterior.

Para criar uma função do IAM para reversão com base em alarmes CloudWatch

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Roles e Create role.
3. Em Select type of trusted entity (Selecionar o tipo de entidade confiável), escolha AWS service (serviço).
4. Imediatamente em Escolher o serviço que usará essa função, selecione O EC2 permite que instâncias do EC2 chamem os serviços da AWS em seu nome e, em seguida, escolha Próximo: permissões.
5. Na página Política de permissões anexadas, pesquise por SSM. CloudWatchAlarmDiscoveryPolicy
6. Selecione essa política e escolha Next: Tags (Próximo: tags).
7. Insira tags para essa função e escolha Next: Review (Próximo: revisão).
8. Na página Criar função, insira **`SSMCloudWatchAlarmDiscoveryRole`** no campo RNome da função e escolha Criar função.
9. Na página Roles (Funções), escolha a função recém-criada. A página Summary é aberta.

Etapa 3: Adicionar uma relação de confiança

Use o procedimento a seguir para configurar a função que você acabou de criar para confiar no AWS AppConfig.

Para adicionar uma relação de confiança para AWS AppConfig

1. Na página Summary da função que você acabou de criar, escolha a guia Trust Relationships e, em seguida, Edit Trust Relationship.
2. Edite a política para incluir somente "`appconfig.amazonaws.com`", conforme mostrado no exemplo a seguir:

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

3. Escolha Update Trust Policy.

Criação de sinalizadores de recursos e dados de configuração de formato livre no AWS AppConfig

Os tópicos desta seção ajudam você a concluir as seguintes tarefas em AWS AppConfig. Essas tarefas criam artefatos importantes para a implantação de dados de configuração.

1. Criar um namespace para o aplicativo

   Para criar um namespace de aplicativo, você cria um AWS AppConfig artefato chamado aplicativo. Um aplicativo é simplesmente uma estrutura organizacional, como uma pasta.

2. Criar ambientes

   Para cada AWS AppConfig aplicativo, você define um ou mais ambientes. Um ambiente é um grupo lógico de AWS AppConfig destinos de implantação, como aplicativos em um `Production` ambiente `Beta` OR. Também é possível definir ambientes para subcomponentes de aplicativos, como os componentes `AWS Lambda functions`, `Containers`, `Web`, `Mobile` e `Back-end`.

   Você pode configurar CloudWatch alarmes da Amazon para cada ambiente para reverter automaticamente alterações problemáticas na configuração. O sistema monitora os alarmes durante uma implantação de configuração. Se um alarme for acionado, o sistema reverterá a configuração.

3. Criar um perfil de configuração

   Um perfil de configuração inclui, entre outras coisas, um URI que permite AWS AppConfig localizar seus dados de configuração em seu local armazenado e um tipo de perfil. AWS AppConfig suporta dois tipos de perfil de configuração: sinalizadores de recursos e configurações de forma livre. Os perfis de configuração do sinalizador de recursos armazenam seus dados no armazenamento de configuração AWS AppConfig hospedado e o URI é simples`hosted`. Para perfis de configuração de formato livre, você pode armazenar seus dados no armazenamento de configuração AWS AppConfig hospedado ou em outro recurso ou AWS serviço do Systems Manager que se integre AWS AppConfig, conforme descrito em. Criando um perfil de configuração de formato livre no AWS AppConfig

   Um perfil de configuração também pode incluir validadores opcionais para garantir que seus dados de configuração estejam sintática e semanticamente corretos. AWS AppConfig executa uma verificação usando os validadores quando você inicia uma implantação. Se algum erro for

detectado, a implantação será interrompida antes de fazer qualquer alteração nos destinos da configuração.

> **Note**
>
> A menos que você tenha necessidades específicas para armazenar segredos AWS Secrets Manager ou gerenciar dados no Amazon Simple Storage Service (Amazon S3), recomendamos hospedar seus dados de configuração no AWS AppConfig armazenamento de configuração hospedado, pois ele oferece mais recursos e aprimoramentos.

Tópicos

Exemplos de configuração

Use AWS AppConfiga capacidade de criar AWS Systems Manager, gerenciar e implantar rapidamente configurações de aplicativos. Uma configuração é um conjunto de definições que influenciam o comportamento do aplicativo. Aqui estão alguns exemplos.

Configuração do sinalizador de atributos

A configuração dos sinalizadores de atributos a seguir ativa ou desativa pagamentos móveis e pagamentos padrão por região.

JSON

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

```
  "default_payments_per_region": {
    "enabled": true
  }
}
```

YAML

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

Configuração operacional

A configuração de formato livre a seguir impõe limites sobre como um aplicativo processa solicitações.

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
```

Configuração da lista de controle de acesso

A configuração de formato livre da lista de controle de acesso a seguir especifica quais usuários ou grupos podem acessar um aplicativo.

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
```

Sobre o perfil do IAM de configuração

Você pode criar a função do IAM que fornece acesso aos dados de configuração usando AWS AppConfig. Ou pode criar o perfil do IAM você mesmo. Se você criar a função usando AWS AppConfig, o sistema cria a função e especifica uma das seguintes políticas de permissões, dependendo do tipo de fonte de configuração que você escolher.

A origem da configuração é um segredo do Secrets Manager

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

A origem da configuração é um parâmetro do Parameter Store

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

```
            ],
            "Resource": [
                "arn:aws:ssm:Região da AWS:account_ID:parameter/parameter_name"
            ]
        }
    ]
    }
```

A origem da configuração é um documento do SSM

```
{

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

Se você criar a função usando AWS AppConfig, o sistema também criará a seguinte relação de confiança para a função.

```
{

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

Como criar um namespace para seu aplicativo no AWS AppConfig

Os procedimentos desta seção ajudam você a criar um AWS AppConfig artefato chamado aplicativo. Um aplicativo é simplesmente uma estrutura organizacional, como uma pasta, que identifica o namespace do aplicativo. Essa construção organizacional tem um relacionamento com uma certa unidade de código executável. Por exemplo, você pode criar um aplicativo chamado MyMobileApp para organizar e gerenciar dados de configuração para um aplicativo móvel instalado por seus usuários. Você deve criar esses artefatos antes de usá-los AWS AppConfig para implantar e recuperar sinalizadores de recursos ou dados de configuração de formato livre.

> **Note**
>
> Você pode usar AWS CloudFormation para criar AWS AppConfig artefatos, incluindo aplicativos, ambientes, perfis de configuração, implantações, estratégias de implantação e versões de configuração hospedadas. Para obter mais informações, consulte AWS AppConfig resource type reference no Guia do usuário do AWS CloudFormation .

Tópicos

Criando um AWS AppConfig aplicativo (console)

Use o procedimento a seguir para criar um AWS AppConfig aplicativo usando o AWS Systems Manager console.

Para criar um aplicativo.

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Aplicações e Criar aplicação.
3. Em Name (Nome), insira um nome para o aplicativo.
4. Em Description (Descrição), insira informações sobre o aplicativo.
5. (Opcional) Na seção Extensões, escolha uma extensão na lista. Para ter mais informações, consulte Sobre AWS AppConfig extensões.

6. (Opcional) Na seção Tags, insira uma chave e um valor opcional. Você pode especificar um máximo de 50 tags para um recurso.
7. Selecione Create application (Criar aplicativo).

AWS AppConfig cria o aplicativo e, em seguida, exibe a guia Ambientes. Vá para Criar ambientes para seu aplicativo no AWS AppConfig.

Criação de um AWS AppConfig aplicativo (linha de comando)

O procedimento a seguir descreve como usar o AWS CLI (no Linux ou no Windows) ou AWS Tools for PowerShell criar um AWS AppConfig aplicativo.

Para criar um aplicativo passo a passo

1. Abra AWS CLI o.
2. Execute o seguinte comando para criar um aplicativo.

   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
   ```

O sistema retorna informações como estas.

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
```

Criar ambientes para seu aplicativo no AWS AppConfig

Para cada AWS AppConfig aplicativo, você define um ou mais ambientes. Um ambiente é um grupo lógico de AppConfig destinos de implantação, como aplicativos em um `Beta` `Production` ambiente, AWS Lambda funções ou contêineres. Também é possível definir ambientes para subcomponentes de aplicativos, como os componentes `Web`, `Mobile` e `Back-end`. Você pode configurar os CloudWatch alarmes da Amazon para cada ambiente. O sistema monitora os alarmes durante uma implantação de configuração. Se um alarme for acionado, o sistema reverterá a configuração.

Antes de começar

Se você quiser ativar AWS AppConfig a reversão de uma configuração em resposta a um CloudWatch alarme, deverá configurar uma função AWS Identity and Access Management (IAM) com permissões para permitir a resposta AWS AppConfig aos CloudWatch alarmes. Escolha essa função no procedimento a seguir. Para ter mais informações, consulte (Opcional) Configurar permissões para reversão com base em alarmes CloudWatch .

Tópicos

- Criação de um AWS AppConfig ambiente (console)
- Criação de um AWS AppConfig ambiente (linha de comando)

Criação de um AWS AppConfig ambiente (console)

Use o procedimento a seguir para criar um AWS AppConfig ambiente usando o AWS Systems Manager console.

Para criar um ambiente

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Aplicativos e, em seguida, escolha o nome de um aplicativo para abrir a página de detalhes.
3. Escolha a guia Ambientes e, em seguida, escolha Criar ambiente.
4. Em Name (Nome), insira um nome para o ambiente.
5. Em Description (Descrição), insira informações sobre o ambiente.
6. (Opcional) Na seção Monitores, escolha o campo de função do IAM e, em seguida, escolha uma função do IAM com permissão para reverter uma configuração se um alarme for acionado.
7. Na lista de CloudWatch alarmes, escolha um ou mais alarmes para monitorar. AWS AppConfig reverte a implantação da configuração se um desses alarmes entrar em estado de alarme.
8. (Opcional) Na seção Associar extensões, escolha uma extensão na lista. Para ter mais informações, consulte Sobre AWS AppConfig extensões.
9. (Opcional) Na seção Tags, insira uma chave e um valor opcional. Você pode especificar um máximo de 50 tags para um recurso.
10. Selecione Create environment (Criar ambiente).

AWS AppConfig cria o ambiente e, em seguida, exibe a página de detalhes do ambiente. Vá para Criação de um perfil de configuração no AWS AppConfig.

Criação de um AWS AppConfig ambiente (linha de comando)

O procedimento a seguir descreve como usar o AWS CLI (no Linux ou no Windows) ou AWS Tools for PowerShell criar um AWS AppConfig ambiente.

Para criar um ambiente passo a passo

1. Abra AWS CLI o.
2. Execute o seguinte comando para criar um ambiente.

   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
```

O sistema retorna informações como estas.

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
```

Vá para Criação de um perfil de configuração no AWS AppConfig.

Criação de um perfil de configuração no AWS AppConfig

Um perfil de configuração inclui, entre outras coisas, um URI que permite AWS AppConfig localizar seus dados de configuração em seu local armazenado e um tipo de configuração. AWS AppConfig suporta dois tipos de perfis de configuração: sinalizadores de recursos e configurações de forma livre. Uma configuração de sinalizador de recurso armazena dados no armazenamento de configuração AWS AppConfig hospedado e o URI é simples`hosted`. Uma configuração de formato livre pode armazenar dados no repositório de configurações AWS AppConfig hospedado, em vários recursos do Systems Manager ou em um AWS serviço que se integre a. AWS AppConfig Para ter mais informações, consulte Criando um perfil de configuração de formato livre no AWS AppConfig.

Um perfil de configuração também pode incluir validadores opcionais para garantir que seus dados de configuração estejam sintática e semanticamente corretos. AWS AppConfig executa uma verificação usando os validadores quando você inicia uma implantação. Se algum erro for detectado, a implantação será interrompida antes de fazer qualquer alteração nos destinos da configuração.

> **Note**
>
> Se possível, recomendamos hospedar seus dados de configuração no armazenamento de configuração AWS AppConfig hospedado, pois ele oferece mais recursos e aprimoramentos.

Tópicos

- Sobre validadores
- Criando um perfil de configuração de sinalizador de recurso no AWS AppConfig
- Criando um perfil de configuração de formato livre no AWS AppConfig

Sobre validadores

Ao criar um perfil de configuração, é possível especificar até dois validadores. Um validador garante que os dados da configuração estejam sintaticamente e semanticamente corretos. Se você planeja usar um validador, deve criá-lo antes de criar o perfil de configuração. AWS AppConfig suporta os seguintes tipos de validadores:

- AWS Lambda funções: Compatível com sinalizadores de recursos e configurações de formato livre.
- Esquema JSON: compatível com configurações de formato livre. (valida AWS AppConfig automaticamente os sinalizadores de recursos em relação a um esquema JSON.)

Tópicos

- Validadores da função AWS Lambda
- Validadores do esquema JSON

Validadores da função AWS Lambda

O validadores da função do Lambda devem ser configurados com o esquema de eventos a seguir. O AWS AppConfig usa esse esquema para invocar a função do Lambda. O conteúdo é uma string codificada em base64 e o URI é uma string.

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

AWS AppConfig verifica se o cabeçalho `X-Amz-Function-Error` Lambda está definido na resposta. O Lambda define esse cabeçalho se a função gerar uma exceção. Para obter mais

informações sobre `X-Amz-Function-Error`, consulte Lidar com erros e tentativas automáticas no AWS Lambda no Guia do desenvolvedor do AWS Lambda .

Veja um exemplo simples de um código de resposta do Lambda para uma validação bem-sucedida.

```
import json

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

Veja um exemplo simples de um código de resposta do Lambda para uma validação malsucedida.

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

Veja a seguir outro exemplo que valida somente se o parâmetro de configuração for um número primo.

```
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 chama sua validação de Lambda ao chamar as operações de `ValidateConfigurationActivity` API `StartDeployment` e. Você deve fornecer permissões `appconfig.amazonaws.com` para invocar seu Lambda. Para obter mais informações, consulte Concedendo acesso de funções aos AWS serviços. AWS AppConfig limita o tempo de execução da validação do Lambda a 15 segundos, incluindo a latência de inicialização.

Validadores do esquema JSON

Se você criar uma configuração em um documento do SSM, será necessário especificar ou criar um esquema JSON para essa configuração. Um esquema JSON define as propriedades permitidas para cada definição de configuração de aplicativo. O esquema JSON funciona como um conjunto de regras para garantir que definições de configuração novas ou atualizadas estejam em conformidade com as melhores práticas exigidas pelo aplicativo. Aqui está um exemplo.

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

Quando você cria uma configuração a partir de um documento SSM, o sistema verifica automaticamente se a configuração está em conformidade com os requisitos do esquema. Caso contrário, o AWS AppConfig retornará um erro de validação.

> **Important**
>
> Observe as seguintes informações importantes sobre validadores do esquema JSON:
>
> - Os dados de configuração armazenados em documentos do SSM devem ser validados em relação a um esquema JSON associado antes que você possa adicionar a configuração ao

sistema. Os parâmetros SSM não exigem um método de validação, mas recomendamos que você crie uma verificação de validação para configurações de parâmetros SSM novas ou atualizadas usando. AWS Lambda

- Uma configuração em um documento SSM usa o tipo de documento `ApplicationConfiguration`. O esquema JSON correspondente usa o tipo de documento `ApplicationConfigurationSchema`.
- AWS AppConfig suporta o esquema JSON versão 4.X para esquema embutido. Se a configuração do seu aplicativo exigir uma versão diferente do esquema JSON, será necessário criar um validador do Lambda.

Criando um perfil de configuração de sinalizador de recurso no AWS AppConfig

Você pode usar sinalizadores de recursos para ativar ou desativar recursos em seus aplicativos ou para configurar características diferentes dos recursos do seu aplicativo usando atributos de sinalizadores. AWS AppConfig armazena configurações de sinalizadores de recursos no repositório de configurações AWS AppConfig hospedado em um formato de sinalizador de recurso que contém dados e metadados sobre seus sinalizadores e os atributos do sinalizador. Para obter mais informações sobre o armazenamento de configuração AWS AppConfig hospedado, consulte a Sobre o armazenamento de configuração AWS AppConfig hospedado seção.

Tópicos

- Criação de um perfil de configuração de sinalizador de recurso (console)
- Criação de um sinalizador de atributos e um perfil de configuração de sinalizadores de atributos (linha de comando)
- Referência de tipo para AWS.AppConfig.FeatureFlags

Antes de começar

No procedimento a seguir, na seção opcional Criptografia, você pode escolher uma chave AWS Key Management Service (AWS KMS). Essa chave gerenciada pelo cliente permite que você criptografe novas versões de dados de configuração no armazenamento de configuração AWS AppConfig hospedado. Para obter mais informações sobre essa chave, consulte AWS AppConfig Suporta chaves do gerente de clientes emSegurança no AWS AppConfig.

O procedimento a seguir também oferece a opção de associar uma extensão a um perfil de configuração do sinalizador de recurso. Uma extensão aumenta sua capacidade de injetar lógica ou comportamento em diferentes pontos durante o AWS AppConfig fluxo de trabalho de criação ou implantação de uma configuração. Para ter mais informações, consulte Sobre AWS AppConfig extensões.

Por fim, na seção Atributos do sinalizador de recurso, ao inserir os detalhes do atributo de um novo sinalizador de recurso, você pode especificar restrições. As restrições garantem que quaisquer valores de atributos inesperados não sejam implantados em seu aplicativo. AWS AppConfig suporta os seguintes tipos de atributos de bandeira e suas restrições correspondentes.

Tipo	Restrição	Descrição
String	Expressão regular	Padrão Regex para a string
	Enum	Lista de valores aceitáveis para a string
Número	Mínimo	Valor numérico mínimo do atributo
	Máximo	Valor numérico máximo do atributo
Booleano	Nenhum	Nenhum
Matriz de strings	Expressão regular	Padrão Regex para os elementos da matriz
	Enum	Lista de valores aceitáveis para os elementos da matriz
Matriz numérica	Mínimo	Valor numérico mínimo para os elementos da matriz
	Máximo	Valor numérico máximo para os elementos da matriz

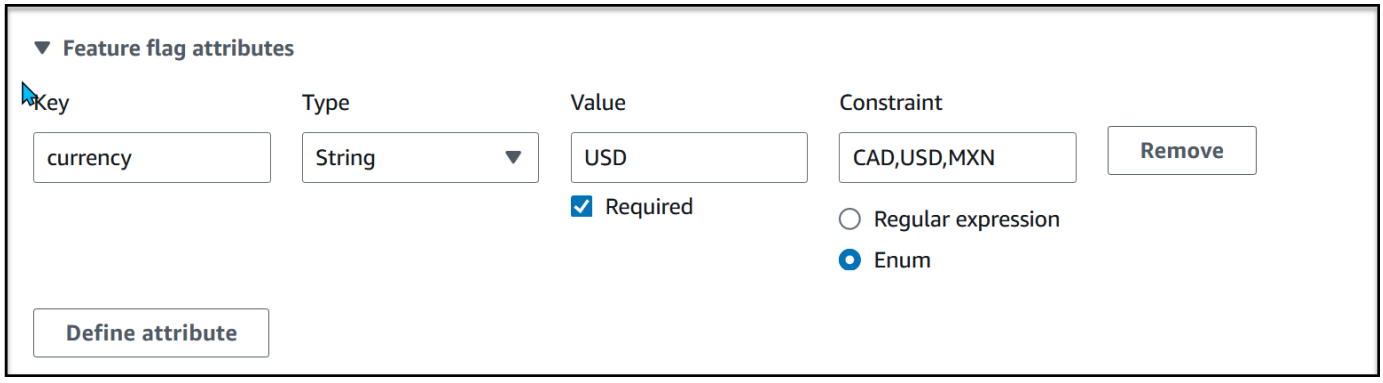
Criação de um perfil de configuração de sinalizador de recurso (console)

Use o procedimento a seguir para criar um perfil de configuração do sinalizador de AWS AppConfig recurso usando o AWS AppConfig console.

Como criar um perfil de configuração

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Aplicativos e, em seguida, escolha um aplicativo no Como criar um namespace para seu aplicativo no AWS AppConfig qual você criou.
3. Escolha a guia Perfis de configuração e sinalizadores de recursos e, em seguida, escolha Criar configuração.
4. Na seção Opções de configuração, escolha Sinalizador de recurso.
5. Role para baixo. Na seção Perfil de configuração, em Nome do perfil de configuração, insira um nome.
6. (Opcional) Expanda Descrição e insira uma descrição.
7. (Opcional) Expanda Opções adicionais e conclua o seguinte, conforme necessário.
    a. Na lista Criptografia, escolha uma chave AWS Key Management Service (AWS KMS) na lista.
    b. Na seção Associar extensões, escolha uma extensão na lista.
    c. Na seção Tags, escolha Adicionar nova tag e, em seguida, especifique uma chave e um valor opcional.
8. Selecione Next (Próximo).
9. Na seção Definição do sinalizador de recurso, em Nome do sinalizador, insira um nome.
10. Em Chave de bandeira, insira um identificador de bandeira para distinguir as bandeiras dentro do mesmo perfil de configuração. Sinalizadores dentro do mesmo perfil de configuração não podem ter a mesma chave. Depois que o sinalizador for criado, você poderá editar o nome do sinalizador, mas não a chave do sinalizador.
11. (Opcional) Expanda Descrição e insira informações sobre esse sinalizador.
12. Selecione Este é um sinalizador de curto prazo e, opcionalmente, escolha uma data em que o sinalizador deve ser desativado ou excluído. Observe que isso AWS AppConfig não desativa o sinalizador.

13. Na seção Atributos do sinalizador, escolha Definir atributo. Os atributos permitem que você forneça valores adicionais em seu sinalizador.
14. Em Chave, especifique uma chave de sinalização e escolha seu tipo na lista Tipo. Como opção, você pode validar valores de atributos com base em restrições especificadas. A imagem a seguir mostra um exemplo.



Escolha Definir atributo para adicionar atributos adicionais.

> **Note**
>
> Observe as seguintes informações:
>
> - Para nomes de atributos, a palavra “ativado” é reservada. Você não pode criar nenhum atributo de sinalizadores de atributos chamado “ativado”. Não há outras palavras reservadas.
> - Os atributos de um sinalizador de atributos só são incluídos na resposta de `GetLatestConfiguration` se esse sinalizador estiver ativado.
> - As chaves de atributos do sinalizador para um determinado sinalizador devem ser exclusivas.
> - Selecione Valor obrigatório para especificar se um valor de atributo é obrigatório.

15. Na seção Valor do sinalizador de recurso, escolha Ativado para ativar o sinalizador. Use esse mesmo botão para desativar um sinalizador quando ele atingir uma data de suspensão de uso especificada, se aplicável.
16. Selecione Next (Próximo).
17. Na página Revisar e salvar, verifique os detalhes do sinalizador e, em seguida, salve e continue com a implantação.

Vá para Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig.

Criação de um sinalizador de atributos e um perfil de configuração de sinalizadores de atributos (linha de comando)

O procedimento a seguir descreve como usar o AWS Command Line Interface (no Linux ou no Windows) ou o Tools for Windows PowerShell para criar um perfil de configuração do sinalizador de AWS AppConfig recurso. Se preferir, você pode usar AWS CloudShell para executar os comandos listados abaixo. Para obter mais informações, consulte O que é o AWS CloudShell? no Guia do usuário do AWS CloudShell .

Para criar uma configuração de sinalizadores de atributos passo a passo

1. Abra AWS CLI o.
2. Crie um perfil de configuração de sinalizadores de atributos especificando seu Tipo como `AWS.AppConfig.FeatureFlags`. O perfil de configuração deve usar `hosted` para o URI de localização.

   Linux

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

   Windows

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

   PowerShell

   ```
   New-APPCConfigurationProfile `
     -Name A_name_for_the_configuration_profile `
     -ApplicationId The_application_ID `
   ```

```
  -LocationUri hosted `
  -Type AWS.AppConfig.FeatureFlags
```

3. Crie seus dados de configuração do sinalizador de atributos. Seus dados devem estar no formato JSON e estar em conformidade com o esquema JSON `AWS.AppConfig.FeatureFlags`. Para obter mais informações sobre o esquema, consulte Referência de tipo para AWS.AppConfig.FeatureFlags.
4. Use a API `CreateHostedConfigurationVersion` para salvar seus dados de configuração do sinalizador de atributos no AWS AppConfig.

   Linux

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

   Windows

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

   PowerShell

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

   Veja um exemplo de comando do Linux.

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

O parâmetro `content` usa os dados codificados `base64` a seguir.

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

O sistema retorna informações como estas.

Linux

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

Windows

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

```
}
```

PowerShell

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

O `service_returned_content_file` contém seus dados de configuração que incluem alguns metadados AWS AppConfig gerados.

> **Note**
>
> Ao criar a versão de configuração hospedada, AWS AppConfig verifica se seus dados estão em conformidade com o esquema `AWS.AppConfig.FeatureFlags` JSON. AWS AppConfig além disso, valida se cada atributo do sinalizador de recurso em seus dados satisfaz as restrições que você definiu para esses atributos.

Referência de tipo para AWS.AppConfig.FeatureFlags

Use o esquema JSON `AWS.AppConfig.FeatureFlags` como referência para criar seus dados de configuração do sinalizador de atributos.

```
{
      "$schema": "http://json-schema.org/draft-07/schema#",
      "definitions": {
        "flagSetDefinition": {
          "type": "object",
          "properties": {
            "version": {
              "$ref": "#/definitions/flagSchemaVersions"
            },
            "flags": {
              "$ref": "#/definitions/flagDefinitions"
            },
            "values": {
              "$ref": "#/definitions/flagValues"
            }
```

```
      },
      "required": ["version", "flags"],
      "additionalProperties": false
    },
    "flagDefinitions": {
      "type": "object",
      "patternProperties": {
        "^[a-z][a-zA-Z\\d-]{0,63}$": {
          "$ref": "#/definitions/flagDefinition"
        }
      },
      "maxProperties": 100,
      "additionalProperties": false
    },
    "flagDefinition": {
      "type": "object",
      "properties": {
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "description": {
          "$ref": "#/definitions/customerDefinedDescription"
        },
        "_createdAt": {
          "type": "string"
        },
        "_updatedAt": {
          "type": "string"
        },
        "_deprecation": {
          "type": "object",
          "properties": {
            "status": {
              "type": "string",
              "enum": ["planned"]
            }
          },
         "additionalProperties": false
        },
        "attributes": {
          "$ref": "#/definitions/attributeDefinitions"
        }
      },
      "additionalProperties": false
```

```
        },
        "attributeDefinitions": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/attributeDefinition"
            }
          },
          "maxProperties": 25,
          "additionalProperties": false
        },
        "attributeDefinition": {
          "type": "object",
          "properties": {
            "description": {
              "$ref": "#/definitions/customerDefinedDescription"
            },
            "constraints": {
              "oneOf": [
                { "$ref": "#/definitions/numberConstraints" },
                { "$ref": "#/definitions/stringConstraints" },
                { "$ref": "#/definitions/arrayConstraints" },
                { "$ref": "#/definitions/boolConstraints" }
              ]
            }
          },
          "additionalProperties": false
        },
        "flagValues": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/flagValue"
            }
          },
          "maxProperties": 100,
          "additionalProperties": false
        },
        "flagValue": {
          "type": "object",
          "properties": {
            "enabled": {
              "type": "boolean"
            },
```

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

```
            "type": "string",
            "enum": ["string"]
      },
            "required": {
              "type": "boolean"
            },
            "pattern": {
              "type": "string",
              "maxLength": 1024
            },
            "enum": {
              "type": "array",
              "maxLength": 100,
              "items": {
                "oneOf": [
                  {
                    "type": "string",
                    "maxLength": 1024
                  },
                  {
                    "type": "integer"
                  }
                ]
              }
            }
          },
          "required": ["type"],
          "not": {
            "required": ["pattern", "enum"]
          },
          "additionalProperties": false
        },
        "numberConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["number"]
      },
            "required": {
              "type": "boolean"
            },
            "minimum": {
              "type": "integer"
```

```
            },
            "maximum": {
              "type": "integer"
            }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "arrayConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["array"]
      },
            "required": {
              "type": "boolean"
            },
            "elements": {
              "$ref": "#/definitions/elementConstraints"
      }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "boolConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["boolean"]
      },
            "required": {
              "type": "boolean"
            }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "elementConstraints": {
          "oneOf": [
            { "$ref": "#/definitions/numberConstraints" },
            { "$ref": "#/definitions/stringConstraints" }
          ]
```

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

> **Important**
>
> Para recuperar dados de configuração do sinalizador de atributos, seu aplicativo deve chamar a API `GetLatestConfiguration`. Não é possível recuperar dados de configuração do sinalizador de atributos chamando `GetConfiguration`, que está obsoleto. Para obter mais informações, consulte GetLatestConfiguração na Referência AWS AppConfig da API.

Quando seu aplicativo chama a GetLatestConfiguração e recebe uma configuração recém-implantada, as informações que definem seus sinalizadores e atributos de recursos são removidas. O JSON simplificado contém um mapa de chaves que correspondem a cada uma das chaves de sinalizadores que você especificou. O JSON simplificado também contém valores `true` ou `false` mapeados para o atributo `enabled`. Se um sinalizador definir `enabled` como `true`, todos os atributos do sinalizador também estarão presentes. O esquema JSON a seguir descreve o formato da saída 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
    }
  }
}
```

Criando um perfil de configuração de formato livre no AWS AppConfig

Um perfil de configuração inclui, entre outras coisas, um URI que permite AWS AppConfig localizar seus dados de configuração em seu local armazenado e um tipo de perfil. AWS AppConfig suporta dois tipos de perfil de configuração: sinalizadores de recursos e configurações de forma livre. Os perfis de configuração do sinalizador de recursos armazenam seus dados no armazenamento de configuração AWS AppConfig hospedado e o URI é simples`hosted`. Para perfis de configuração de formato livre, você pode armazenar seus dados no repositório de configuração AWS AppConfig hospedado ou em qualquer um dos seguintes AWS serviços e recursos do Systems Manager:

Local	Tipos de arquivos compatíveis
AWS AppConfig armazenamento de configuração hospedado	YAML, JSON e texto, se adicionados usando o. AWS Management Console Qualquer tipo de arquivo, se adicionado usando a ação AWS AppConfig CreateHostedConfigurationVersionda API.
Amazon Simple Storage Service (Amazon S3)	Any
AWS CodePipeline	Pipeline (conforme definido pelo serviço)
AWS Secrets Manager	Segredo (conforme definido pelo serviço)
AWS Systems Manager Parameter Store	Parâmetros de string padrão e seguros (conforme definido pelo Parameter Store)

Local	Tipos de arquivos compatíveis
AWS Systems Manager armazenamento de documentos (documentos SSM)	YAML, JSON, texto

Um perfil de configuração também pode incluir validadores opcionais para garantir que seus dados de configuração estejam sintática e semanticamente corretos. AWS AppConfig executa uma verificação usando os validadores quando você inicia uma implantação. Se algum erro for detectado, a implantação será interrompida antes de fazer qualquer alteração nos destinos da configuração.

> **Note**
>
> Se possível, recomendamos hospedar seus dados de configuração no armazenamento de configuração AWS AppConfig hospedado, pois ele oferece mais recursos e aprimoramentos.

Para configurações de formato livre armazenadas no repositório de configurações AWS AppConfig hospedado ou nos documentos SSM, você pode criar a configuração de forma livre usando o console do Systems Manager ao criar um perfil de configuração. O processo está descrito posteriormente neste tópico.

Para configurações de formato livre armazenadas no Parameter Store, Secrets Manager ou Amazon S3, você deve primeiro criar o parâmetro, segredo ou objeto e armazená-lo no respectivo armazenamento de configuração. Depois de armazenar os dados de configuração, use o procedimento neste tópico para criar o perfil de configuração.

Tópicos

- Sobre cotas e limitações do armazenamento de configuração
- Sobre o armazenamento de configuração AWS AppConfig hospedado
- Sobre configurações armazenadas no Amazon S3
- Criação de uma configuração de formato livre e um perfil de configuração

Sobre cotas e limitações do armazenamento de configuração

Os repositórios de configuração suportados pelo AWS AppConfig têm as seguintes cotas e limitações.

	AWS AppConfig armazenamento de configuração hospedado	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager Document Store	AWS CodePipeline
Limite de tamanho da configuração	2 MB padrão, 4 MB no máximo	2 MB Aplicado por AWS AppConfig, não pelo S3	4 KB (nível gratuito) / 8 KB (parâmetros avançados)	64 KB	64 KB	2 MB Imposta por AWS AppConfig, não CodePipeline
Limite de armazenamento de recursos	1 GB	Ilimitado	10.000 parâmetros (nível gratuito) / 100.000 parâmetros (parâmetros avançados)	500.000	500 documentos	Limitado pelo número de perfis de configuração por aplicativo (100 perfis por aplicativo)
Criptografia do lado do servidor	Sim	SSE-S3, SSE-KMS	Sim	Sim	Não	Sim
AWS CloudFormation apoio	Sim	Não é para criar ou atualizar dados	Sim	Sim	Não	Sim

	AWS AppConfig armazenamento de configuração hospedado	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager Document Store	AWS CodePipeline
Definição de preço	Gratuito	Consulte Def de preço do Amazon S3	Consulte Definição de preço do AWS Systems Manager	Consulte Definição de preço do AWS Secrets Manager	Gratuito	Consulte Definição de preço do AWS CodePipeline

Sobre o armazenamento de configuração AWS AppConfig hospedado

AWS AppConfig inclui um armazenamento de configuração interno ou hospedado. As configurações devem ter 2 MB ou menos. O armazenamento de configuração AWS AppConfig hospedado oferece os seguintes benefícios em relação a outras opções de armazenamento de configuração.

- Não é necessário configurar outros serviços, como o Amazon Simple Storage Service (Amazon S3) ou o repositório de parâmetros.
- Você não precisa configurar permissões AWS Identity and Access Management (IAM) para usar o armazenamento de configurações.
- É possível armazenar configurações em YAML, JSON ou como documentos de texto.
- Não há custo para usar o armazenamento.
- É possível criar uma configuração e adicioná-la ao armazenamento ao criar um perfil de configuração.

Sobre configurações armazenadas no Amazon S3

Você pode armazenar configurações em um bucket do Amazon Simple Storage Service (Amazon S3). Ao criar o perfil de configuração, você especifica o URI para um único objeto do S3 em um bucket. Você também especifica o Amazon Resource Name (ARN) de uma função AWS Identity and

Access Management (IAM) que dá AWS AppConfig permissão para obter o objeto. Antes de criar um perfil de configuração para um objeto do Amazon S3, lembre-se das restrições a seguir.

Restrição	Detalhes
Tamanho	As configurações armazenadas como objetos do S3 podem ter um tamanho máximo de 1 MB.
Object encryption	Um perfil de configuração pode ter como alvo objetos criptografados SSE-S3 e SSE-KMS.
Classes de armazenamento	AWS AppConfig suporta as seguintes classes de armazenamento S3: `STANDARDINTELLIGENT_TIERING` ,`REDUCED_REDUNDANCY` ,`STANDARD_IA` , e. `ONEZONE_IA` As classes a seguir não são compatíveis: todas as classes do S3 Glacier (`GLACIER` e `DEEP_ARCHIVE`).
Versionamento	AWS AppConfig exige que o objeto S3 use o controle de versão.

Configuração de permissões para uma configuração armazenada como um objeto do Amazon S3

Ao criar um perfil de configuração para uma configuração armazenada como um objeto do S3, você deve especificar um ARN para uma função do IAM que AWS AppConfig dê permissão para obter o objeto. A função deve incluir as seguintes permissões:

Permissões para acessar o objeto do S3

- s3: GetObject
- s3: Versão GetObject

Permissões para listar buckets do S3

s3: ListAll MyBuckets

Permissões para acessar o bucket do S3 em que o objeto está armazenado

- s3: Localização GetBucket
- s3: GetBucket Controle de versão
- s3: ListBucket
- s3: Versões ListBucket

Conclua o procedimento a seguir para criar uma função que permita AWS AppConfig obter uma configuração armazenada em um objeto do S3.

Criação da política do IAM para acessar um objeto do S3

Use o procedimento a seguir para criar uma política do IAM que permita AWS AppConfig obter uma configuração armazenada em um objeto do S3.

Para criar uma política do IAM para acessar um objeto do S3

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas e, em seguida, Criar política.
3. Na página Criar política, escolha a guia JSON.
4. Atualize a política de exemplo a seguir com informações sobre o bucket do S3 e o objeto de configuração. Depois, cole a política no campo de texto na guia JSON . Substitua os *valores do espaço reservado* por suas próprias informações.

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

```
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

5. Escolha Revisar política.
6. Na página Review policy (Revisar política), digite um nome na caixa Name (Nome) e, depois, uma descrição.
7. Escolha Criar política. O sistema faz com que você retorne para a página Roles.

Criação do perfil do IAM para acessar um objeto do S3

Use o procedimento a seguir para criar uma função do IAM que permita AWS AppConfig obter uma configuração armazenada em um objeto do S3.

Para criar um perfil do IAM para acessar um objeto do Amazon S3

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Roles e Create role.
3. Na seção Selecionar tipo de entidade confiável, escolha Serviço da AWS .
4. Na seção Choose a case use (Escolher um caso de uso) em Common use cases (Casos de uso comuns), escolha EC2 e, depois, escolha Next: Permissions (Próximo: Permissões).
5. Na página Attach permissions policy (Anexar política de permissões), na caixa de pesquisa, insira o nome da política criada no procedimento anterior.
6. Selecione essa política e escolha Next: Tags (Próximo: Tags).
7. Na página Adicionar tags (opcional), insira uma chave e um valor opcional e escolha Próximo: revisão.

8. Na página Review (Revisar), digite um nome na caixa Role name (Nome da função) e, depois, uma descrição.
9. Selecione Create role (Criar função). O sistema faz com que você retorne para a página Roles.
10. Na página Roles (Funções), escolha a função que você acabou de criar para abrir a página Summary (Resumo). Anote os valores de Role Name (Nome da função) e Role ARN (ARN da função). Você especificará o ARN da função ao criar o perfil de configuração posteriormente neste tópico.

Criar um relacionamento de confiança

Use o procedimento a seguir para configurar a função que você acabou de criar para confiar no AWS AppConfig.

Como adicionar um relacionamento de confiança

1. Na página Summary da função que você acabou de criar, escolha a guia Trust Relationships e, em seguida, Edit Trust Relationship.
2. Exclua `"ec2.amazonaws.com"` e adicione `"appconfig.amazonaws.com"`, conforme mostrado no exemplo a seguir.

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

3. Escolha Update Trust Policy.

Criação de uma configuração de formato livre e um perfil de configuração

Esta seção descreve como criar uma configuração de formato livre e um perfil de configuração. Antes de começar, anote as informações a seguir.

- O procedimento a seguir requer que você especifique um perfil de serviço do IAM para que o AWS AppConfig possa acessar os dados de configuração no armazenamento de configuração escolhido. Essa função não é necessária se você usar o armazenamento de configuração AWS AppConfig hospedado. Se você escolher o S3, o Parameter Store ou o armazenamento de documentos do Systems Manager, será necessário escolher um perfil do IAM existente ou a opção para que o sistema crie automaticamente o perfil para você. Para obter mais informações sobre essa função, consulte Sobre o perfil do IAM de configuração.
- O procedimento a seguir também oferece a opção de associar uma extensão a um perfil de configuração do sinalizador de recurso. Uma extensão aumenta sua capacidade de injetar lógica ou comportamento em diferentes pontos durante o AWS AppConfig fluxo de trabalho de criação ou implantação de uma configuração. Para ter mais informações, consulte Sobre AWS AppConfig extensões.
- Se desejar criar um perfil de configuração para configurações armazenadas no S3, será necessário configurar permissões. Para obter mais informações sobre permissões e outros requisitos para usar o S3 como um armazenamento de configuração, consulte Sobre configurações armazenadas no Amazon S3.
- Se você deseja usar validadores, revise os detalhes e os requisitos para usá-los. Para ter mais informações, consulte Sobre validadores.

Tópicos

Criação de um perfil AWS AppConfig de configuração de formato livre (console)

Use o procedimento a seguir para criar um perfil de configuração de AWS AppConfig forma livre e (opcionalmente) uma configuração de forma livre usando o console. AWS Systems Manager

Para criar um perfil de configuração de formato livre

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Aplicativos e, em seguida, escolha um aplicativo no Como criar um namespace para seu aplicativo no AWS AppConfig qual você criou.
3. Escolha a guia Perfis de configuração e sinalizadores de recursos e, em seguida, escolha Criar configuração.

4. Na seção Opções de configuração, escolha Configuração de forma livre.
5. Em Nome do perfil de configuração, insira um nome para o perfil de configuração.
6. (Opcional) Expanda Descrição e insira uma descrição.
7. (Opcional) Expanda Opções adicionais e conclua o seguinte, conforme necessário.

   a. Na seção Associar extensões, escolha uma extensão na lista.

   b. Na seção Tags, escolha Adicionar nova tag e, em seguida, especifique uma chave e um valor opcional.
8. Selecione Next (Próximo).
9. Na página Especificar dados de configuração, na seção Definição de configuração, escolha uma opção.
10. Preencha os campos da opção selecionada, conforme descrito na tabela a seguir.

Opção selecionada	Detalhes
AWS AppConfig configuração hospedada	Escolha Texto, JSON ou YAML e insira sua configuração no campo. Vá para a Etapa 12 neste procedimento.
Objeto Amazon S3	Insira o URI do objeto no campo Fonte do objeto S3 e vá para a Etapa 11 neste procedimento.
AWS CodePipeline	Escolha Avançar e vá para a Etapa 12 neste procedimento.
Segredo do Secrets Manager	Escolha o segredo na lista, vá para a Etapa 11 neste procedimento.
AWS Systems Manager parameter	Escolha o parâmetro na lista e vá para a Etapa 11 neste procedimento.
AWS Systems Manager document	1. Escolha um documento na lista ou escolha Criar novo documento. 2. Se você escolher Criar novo documento , em Nome do documento, insira um nome. Opcionalmente, expanda Nome da

Opção selecionada	Detalhes
	versão e insira um nome para a versão do documento. 3. Para Esquema de configuração do aplicativo, escolha o esquema JSON na lista ou escolha Criar esquema. Se escolher Criar esquema, o Systems Manager abrirá a página Criar esquema. Insira os detalhes do esquema e escolha Criar esquema de configuração do aplicativo. 4. Na seção Content (Conteúdo) escolha YAML ou JSON e insira os dados de configuração no campo.

11. Na seção Função de serviço, escolha Nova função de serviço para AWS AppConfig criar a função do IAM que fornece acesso aos dados de configuração. AWS AppConfig preenche automaticamente o campo Nome da função com base no nome que você inseriu anteriormente. Ou escolha Função de serviço existente. Escolha a função usando a lista Role ARN (ARN de função).
12. Opcionalmente, na página Adicionar validadores, escolha Esquema JSON ou. AWS Lambda Se você escolher JSON Schema (Esquema JSON), insira o esquema JSON no campo. Se você escolher AWS Lambda, escolha o nome de recurso da Amazon (ARN) da função e a versão na lista.

> **Important**
>
> Os dados de configuração armazenados em documentos do SSM devem ser validados em relação a um esquema JSON associado antes que você possa adicionar a configuração ao sistema. Os parâmetros SSM não exigem um método de validação, mas recomendamos que você crie uma verificação de validação para configurações de parâmetros SSM novas ou atualizadas usando. AWS Lambda

13. Selecione Next (Próximo).
14. Na página Revisar e salvar, escolha Salvar e continuar com a implantação.

> **Important**
>
> Se você criou um perfil de configuração para AWS CodePipeline, deverá criar um pipeline CodePipeline que especifique AWS AppConfig como o provedor de implantação. Você não precisa executar Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig. No entanto, você deve configurar um cliente para receber atualizações de configuração do aplicativo conforme descrito em Recuperação de configurações chamando diretamente as APIs. Para obter informações sobre a criação de um pipeline que especifica AWS AppConfig como provedor de implantação, consulte Tutorial: Criar um pipeline usado AWS AppConfig como provedor de implantação no Guia do AWS CodePipeline usuário.

Vá para Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig.

Criação de um perfil AWS AppConfig de configuração de formato livre (linha de comando)

O procedimento a seguir descreve como usar o AWS CLI (no Linux ou no Windows) ou AWS Tools for PowerShell criar um perfil de configuração de AWS AppConfig formato livre. Se preferir, você pode usar AWS CloudShell para executar os comandos listados abaixo. Para obter mais informações, consulte O que é o AWS CloudShell? no Guia do usuário do AWS CloudShell .

> **Note**
>
> Para configurações de formato livre hospedadas no repositório de configurações AWS AppConfig hospedado, você especifica o `hosted` URI de localização.

Para criar um perfil de configuração usando o AWS CLI

1. Abra AWS CLI o.
2. Execute o comando a seguir para criar um perfil de configuração de formato livre.

   Linux

```
aws appconfig create-configuration-profile \
  --application-id The_application_ID \
  --name A_name_for_the_configuration_profile \
  --description A_description_of_the_configuration_profile \
  --location-uri A_URI_to_locate_the_configuration or hosted \
```

```
  --retrieval-role-
arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified
 \
  --tags User_defined_key_value_pair_metadata_of_the_configuration_profile \
  --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Windows

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

PowerShell

```
New-APPCConfigurationProfile `
  -Name A_name_for_the_configuration_profile `
  -ApplicationId The_application_ID `
  -Description Description_of_the_configuration_profile `
  -LocationUri A_URI_to_locate_the_configuration or hosted `
  -
RetrievalRoleArn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_
 `
  -
Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_configuration_profile
 `
  -Validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Important

Observe as seguintes informações importantes:

- Se você criou um perfil de configuração para AWS CodePipeline, deverá criar um pipeline CodePipeline que especifique AWS AppConfig como o provedor de implantação. Você não precisa executar Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig. No entanto, você deve configurar um cliente para receber atualizações de configuração do aplicativo conforme descrito em Recuperação de configurações chamando diretamente as APIs. Para obter informações sobre a criação de um pipeline que especifica AWS AppConfig como provedor de implantação, consulte Tutorial: Criar um pipeline usado AWS AppConfig como provedor de implantação no Guia do AWS CodePipeline usuário.
- Se você criou uma configuração no armazenamento de configuração AWS AppConfig hospedado, poderá criar novas versões da configuração usando as operações da CreateHostedConfigurationVersionAPI. Para ver AWS CLI detalhes e exemplos de comandos dessa operação de API, consulte create-hosted-configuration-version na Referência de comandos.AWS CLI

Vá para Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig.

Outras fontes de dados de configuração

Este tópico inclui informações sobre outros AWS serviços que se integram com AWS AppConfig o.

AWS AppConfig integração com AWS Secrets Manager

O Secrets Manager ajuda você a criptografar, armazenar e recuperar credenciais com segurança para bancos de dados e outros serviços. Em vez de codificar credenciais em seus aplicativos, você pode fazer chamadas ao Secrets Manager para recuperar suas credenciais sempre que necessário. O Secrets Manager ajuda você a proteger o acesso aos seus recursos e dados de TI, permitindo que você alterne e gerencie o acesso aos seus segredos.

Ao criar um perfil de configuração de formato livre, você pode escolher o Secrets Manager como fonte de seus dados de configuração. Você deve se integrar ao Secrets Manager e criar um segredo antes de criar o perfil de configuração. Para obter mais informações sobre o Secrets Manager, consulte O que é AWS Secrets Manager? no Guia do AWS Secrets Manager usuário. Para obter informações sobre a criação de um perfil de configuração que usa o Secrets Manager, consulte Criação de sinalizadores de recursos e dados de configuração de formato livre no AWS AppConfig.

Implantação de sinalizadores de atributos e dados de configuração no AWS AppConfig

Depois de criar os artefatos necessários para trabalhar com sinalizadores de atributos e dados de configuração de formato livre, você pode criar uma nova implantação. Ao criar uma nova implantação, você especifica as seguintes informações:

- O ID do aplicativo
- O ID do perfil de configuração
- A versão da configuração
- O ID do ambiente no qual você deseja implantar os dados de configuração
- O ID da estratégia de implantação que define a rapidez com que você deseja que as alterações entrem em vigor
- Um ID de chave AWS Key Management Service (AWS KMS) para criptografar os dados usando uma chave gerenciada pelo cliente.

Quando você chama a ação StartDeploymentda API, AWS AppConfig executa as seguintes tarefas:

1. Recupera os dados de configuração do armazenamento de dados subjacente usando o URI de localização no perfil de configuração.
2. Verifica se os dados de configuração estão sintática e semanticamente corretos usando os validadores especificados ao criar o perfil de configuração.
3. Armazena em cache uma cópia dos dados para que estejam prontos para serem recuperados pelo seu aplicativo. Essa cópia em cache é chamada de dados implantados.

AWS AppConfig se integra à Amazon CloudWatch para monitorar implantações. Se uma implantação acionar um alarme CloudWatch, reverte AWS AppConfig automaticamente a implantação para minimizar o impacto nos usuários do aplicativo.

Tópicos

- Como trabalhar com estratégias de implantação
- Implantar uma configuração
- AWS AppConfig integração de implantação com CodePipeline

Como trabalhar com estratégias de implantação

Uma estratégia de implantação permite que você libere lentamente as alterações nos ambientes de produção em questão de minutos ou horas. Uma estratégia AWS AppConfig de implantação define os seguintes aspectos importantes de uma implantação de configuração.

<table>
<tr><th>Configuração</th><th>Descrição</th></tr>
<tr><td>Tipo de implantação</td><td>O tipo de implantação define como a configuração é implantada ou implementada. AWS AppConfig suporta tipos de implantação linear e exponencial.
Linear: para esse tipo, AWS AppConfig processa a implantação por incrementos do fator de crescimento distribuídos uniformemente pela implantação. Veja um exemplo de cronograma para uma implantação de 10 horas que usa 20% de crescimento linear:<table><tr><th>Tempo decorrido</th><th>Progresso da implantação</th></tr><tr><td>0 hora</td><td>0%</td></tr><tr><td>2 horas</td><td>20%</td></tr><tr><td>4 horas</td><td>40%</td></tr><tr><td>6 horas</td><td>60%</td></tr><tr><td>8 horas</td><td>80%</td></tr><tr><td>10 horas</td><td>100%</td></tr></table>
Exponencial: para esse tipo, o AWS AppConfig processa a implantação exponencialmente usando a seguinte fórmula: <code>G*(2^N)</code>. Nessa fórmula, <code>G</code> é a</td></tr>
</table>

Configuração	Descrição

	porcentagem de etapa especificada pelo usuário e N é o número de etapas até que a configuração seja implantada em todos os destinos. Por exemplo, se você especific ar um fator de crescimento de 2, o sistema implementará a configuração da seguinte maneira:
	
	`2*(2^0)`
	`2*(2^1)`
	`2*(2^2)`
	
	Expresso numericamente, a implantação é feita da seguinte forma: 2% dos alvos, 4% dos alvos, 8% dos alvos e continua até que a configuração tenha sido implantada em todos os destinos.

Percentagem de etapa (fator de crescimento)	Essa configuração especifica a porcentagem de chamadores a serem direcionados durante cada etapa da implantação. **Note** No SDK e na Referência da API do AWS AppConfig, `step percentage` é chamado de `growth factor`.
Tempo de implantação	Essa configuração especifica o período de tempo durante o qual é AWS AppConfig implantado nos hosts. Isso não é um valor de tempo limite. É uma janela de tempo durante a qual a implantação é processada em intervalos.

Configuração	Descrição
Tempo de incorporação	Essa configuração especifica a quantidade de tempo que AWS AppConfig monitora os CloudWatch alarmes da Amazon após a configuração ter sido implantada em 100% de suas metas, antes de considerar que a implantação foi concluída. Se um alarme for acionado durante esse período, o AWS AppConfig reverterá a implantação. Você deve configurar as permissões AWS AppConfig para reverter com base nos CloudWatch alarmes. Para ter mais informações, consulte (Opcional) Configurar permissões para reversão com base em alarmes CloudWatch .

Você pode escolher uma estratégia predefinida incluída AWS AppConfig ou criar a sua própria.

Tópicos

- Estratégias de implantação predefinidas
- Criar uma estratégia de implantação

Estratégias de implantação predefinidas

AWS AppConfig inclui estratégias de implantação predefinidas para ajudá-lo a implantar rapidamente uma configuração. Em vez de criar suas próprias estratégias, é possível escolher uma das seguintes opções ao implantar uma configuração.

Estratégia de implantação	Descrição
AppConfig. PercentEvery Linear 20 6 minutos	AWS recomendado: Essa estratégia implanta a configuração para 20% de todos os destinos a cada seis minutos para uma implantação de 30 minutos. O sistema monitora os CloudWatch alarmes da

Estratégia de implantação	Descrição
	Amazon por 30 minutos. Se nenhum alarme for recebido nesse período, a implantação será concluída. Se um alarme for acionado durante esse período, AWS AppConfig reverte a implantação. Recomendamos usar essa estratégia para implantações de produção porque ela se alinha às AWS melhores práticas e inclui ênfase adicional na segurança da implantação devido à sua longa duração e tempo de cozimento.
AppConfig. Canário 10 por cento 20 minutos	AWS recomendado: Essa estratégia processa a implantação exponencialmente usando um fator de crescimento de 10% durante 20 minutos. O sistema monitora CloudWatch os alarmes por 10 minutos. Se nenhum alarme for recebido nesse período, a implantação será concluída . Se um alarme for acionado durante esse período, AWS AppConfig reverte a implantaç ão. Recomendamos usar essa estratégia para implantações de produção porque ela se alinha às AWS melhores práticas para implantações de configuração.

Estratégia de implantação	Descrição
AppConfig.AllAtOnce	Rápida: Essa estratégia implanta a configuração em todos os destinos de forma imediata. O sistema monitora CloudWatch os alarmes por 10 minutos. Se nenhum alarme for recebido nesse período, a implantação será concluída. Se um alarme for acionado durante esse período, o AWS AppConfig reverterá a implantação.
AppConfig. PercentEvery Linear 50 30 segundos	Teste/demonstração: Essa estratégia implanta a configuração para metade de todos os destinos a cada 30 segundos para uma implantação de um minuto. O sistema monitora os CloudWatch alarmes da Amazon por 1 minuto. Se nenhum alarme for recebido nesse período, a implantação será concluída. Se um alarme for acionado durante esse período, AWS AppConfig reverte a implantação. Recomendamos usar esta estratégia apenas para fins de teste ou demonstração, pois ela tem uma curta duração e tempo de incorpora ção.

Criar uma estratégia de implantação

Se você não quiser usar uma das estratégias de implantação predefinidas, você pode criar a sua própria. Você pode criar um máximo de 20 estratégias de implantação. Ao implantar uma configuração, você pode escolher a estratégia de implantação mais adequada para o aplicativo e o ambiente.

Criação de uma estratégia de implantação do AWS AppConfig (console)

Use o procedimento a seguir para criar uma estratégia de AWS AppConfig implantação usando o AWS Systems Manager console.

Como criar uma estratégia de implantação

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Estratégias de implantação e, em seguida, escolha Criar estratégia de implantação.
3. Em Name (Nome), insira um nome para a estratégia de implantação.
4. Em Description (Descrição), insira informações sobre a estratégia de implantação.
5. Em Deployment type (Tipo de implantação), selecione um tipo.
6. Em Step percentage (Porcentagem de etapa), escolha a porcentagem de chamadores a serem direcionados durante cada etapa da implantação.
7. Em Deployment time (Tempo de implantação), insira a duração total da implantação em minutos ou horas.
8. Em Bake time, insira o tempo total, em minutos ou horas, para monitorar CloudWatch os alarmes da Amazon antes de prosseguir para a próxima etapa de uma implantação ou antes de considerar que a implantação foi concluída.
9. Na seção Tags, insira uma chave e um valor opcional. Você pode especificar um máximo de 50 tags para um recurso.
10. Escolha Create deployment strategy (Criar estratégia de implantação).

> **Important**
>
> Se você criou um perfil de configuração para AWS CodePipeline, deverá criar um pipeline CodePipeline que especifique AWS AppConfig como o provedor de implantação. Você não precisa executar Implantar uma configuração. No entanto, você deve configurar um cliente para receber atualizações de configuração do aplicativo conforme descrito em Recuperação de configurações chamando diretamente as APIs. Para obter informações sobre a criação de um pipeline que especifica AWS AppConfig como provedor de implantação, consulte Tutorial: Criar um pipeline usado AWS AppConfig como provedor de implantação no Guia do AWS CodePipeline usuário.

Vá para Implantar uma configuração.

Criar uma estratégia de implantação do AWS AppConfig (linha de comando)

O procedimento a seguir descreve como usar o AWS CLI (no Linux ou no Windows) ou AWS Tools for PowerShell criar uma estratégia de AWS AppConfig implantação.

Para criar uma estratégia de implantação passo a passo

1. Abra AWS CLI o.
2. Execute o seguinte comando para criar uma estratégia de implantação.

   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
```

O sistema retorna informações como estas.

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
```

Implantar uma configuração

Depois de criar os artefatos necessários para trabalhar com sinalizadores de recursos e dados de configuração de formato livre, você pode criar uma nova implantação usando o AWS Management Console, o ou o AWS CLI SDK. Iniciar uma implantação em AWS AppConfig chama a operação StartDeploymentda API. Essa chamada inclui os IDs do aplicativo do AWS AppConfig , do ambiente, do perfil de configuração e (opcionalmente) a versão de dados de configuração a ser implantada. A chamada também inclui o ID da estratégia de implantação a ser usada, que determina como os dados de configuração são implantados.

Se você implantar segredos armazenados em AWS Secrets Manager objetos do Amazon Simple Storage Service (Amazon S3) criptografados com uma chave gerenciada pelo cliente ou parâmetros de cadeia de caracteres seguros armazenados AWS Systems Manager no Parameter Store criptografados com uma chave gerenciada pelo cliente, você deverá especificar um valor para o parâmetro. `KmsKeyIdentifier` Se sua configuração não estiver criptografada ou estiver criptografada com um Chave gerenciada pela AWS, não é necessário especificar um valor para o `KmsKeyIdentifier` parâmetro.

> **Note**
>
> O valor especificado para `KmsKeyIdentifier` deve ser uma chave gerenciada pelo cliente. Não precisa ser a mesma chave que você usou para criptografar sua configuração. Quando você inicia uma implantação com um`KmsKeyIdentifier`, a política de permissão anexada ao seu diretor AWS Identity and Access Management (IAM) deve permitir a `kms:GenerateDataKey` operação.

AWS AppConfig monitora a distribuição para todos os hosts e relata o status. Se uma distribuição falhar, AWS AppConfig reverte a configuração.

> **Note**
>
> Você só pode implantar uma única configuração por vez em determinado ambiente. No entanto, pode implantar uma única configuração em ambientes diferentes ao mesmo tempo.

Implantar uma configuração (console)

Use o procedimento a seguir para implantar uma AWS AppConfig configuração usando o AWS Systems Manager console.

Como implantar uma configuração usando o console

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha Aplicativos e escolha um aplicativo no Como criar um namespace para seu aplicativo no AWS AppConfig qual você criou.
3. Na guia Ambientes, preencha o botão de rádio de um ambiente e escolha Exibir detalhes.
4. Selecione Iniciar implantação.
5. Em Configuration (Configuração), escolha uma configuração na lista.
6. Dependendo da origem da sua configuração, use a lista de versões para escolher a versão que você deseja implantar.
7. Em Deployment strategy (Estratégia de implantação), escolha uma estratégia da lista.
8. (Opcional) Em Descrição da implantação, insira uma descrição.
9. Para opções adicionais de criptografia, escolha uma AWS Key Management Service chave na lista.
10. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor opcional. Você pode especificar um máximo de 50 tags para um recurso.
11. Selecione Iniciar implantação.

Implantar uma configuração (linha de comando)

O procedimento a seguir descreve como usar o AWS CLI (no Linux ou no Windows) ou AWS Tools for PowerShell implantar uma AWS AppConfig configuração.

Para implantar uma configuração passo a passo

1. Abra AWS CLI o.
2. Execute o comando a seguir para implantar uma configuração.

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
```

O sistema retorna informações como estas.

Linux

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
```

```
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
   "DeploymentNumber": The sequence number of the deployment,
   "ConfigurationName": "The name of the configuration",
   "ConfigurationLocationUri": "Information about the source location of the
 configuration",
   "ConfigurationVersion": "The configuration version that was deployed",
   "Description": "The description of the deployment",
   "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
   "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
   "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
   "State": "The state of the deployment",

   "EventLog": [
      {
         "Description": "A description of the deployment event",
         "EventType": "The type of deployment event",
         "OccurredAt": The date and time the event occurred,
         "TriggeredBy": "The entity that triggered the deployment event"
      }
   ],

   "PercentageComplete": The percentage of targets for which the deployment is
 available,
   "StartedAt": The time the deployment started,
   "CompletedAt": The time the deployment completed
}
```

Windows

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
```

```
  "DeploymentNumber": The sequence number of the deployment,
  "ConfigurationName": "The name of the configuration",
  "ConfigurationLocationUri": "Information about the source location of the
 configuration",
  "ConfigurationVersion": "The configuration version that was deployed",
  "Description": "The description of the deployment",
  "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
  "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
  "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
  "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
  "State": "The state of the deployment",

  "EventLog": [
     {
        "Description": "A description of the deployment event",
        "EventType": "The type of deployment event",
        "OccurredAt": The date and time the event occurred,
        "TriggeredBy": "The entity that triggered the deployment event"
     }
  ],

  "PercentageComplete": The percentage of targets for which the deployment is
 available,
  "StartedAt": The time the deployment started,
  "CompletedAt": The time the deployment completed
}
```

PowerShell

```
ApplicationId               : The ID of the application that was deployed
CompletedAt                 : The time the deployment completed
ConfigurationLocationUri    : Information about the source location of the
 configuration
ConfigurationName           : The name of the configuration
ConfigurationProfileId      : The ID of the configuration profile that was
 deployed
ConfigurationVersion        : The configuration version that was deployed
ContentLength               : Runtime of the deployment
DeploymentDurationInMinutes : Total amount of time the deployment lasted
DeploymentNumber            : The sequence number of the deployment
```

```
DeploymentStrategyId        : The ID of the deployment strategy that was
 deployed
Description                 : The description of the deployment
EnvironmentId               : The ID of the environment that was deployed
EventLog                    : {Description : A description of the deployment
 event, EventType : The type of deployment event, OccurredAt : The date and time
 the event occurred,
         TriggeredBy : The entity that triggered the deployment event}
FinalBakeTimeInMinutes      : Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete
GrowthFactor                : The percentage of targets to receive a deployed
 configuration during each interval
GrowthType                  : The linear or exponential algorithm used to define
 how percentage grew over time
HttpStatusCode              : HTTP Status of the runtime
PercentageComplete          : The percentage of targets for which the deployment
 is available
ResponseMetadata            : Runtime Metadata
StartedAt                   : The time the deployment started
State                       : The state of the deployment
```

AWS AppConfig integração de implantação com CodePipeline

AWS AppConfig é uma ação de implantação integrada para AWS CodePipeline (CodePipeline). CodePipeline é um serviço de entrega contínua totalmente gerenciado que ajuda você a automatizar seus pipelines de lançamento para atualizações rápidas e confiáveis de aplicativos e infraestrutura. CodePipeline automatiza as fases de criação, teste e implantação do seu processo de lançamento sempre que houver uma alteração no código, com base no modelo de lançamento que você define. Para ter mais informações, consulte O que é o AWS CodePipeline?

A integração AWS AppConfig com CodePipeline oferece os seguintes benefícios:

- Os clientes que CodePipeline costumavam gerenciar a orquestração agora têm um meio leve de implantar alterações de configuração em seus aplicativos sem precisar implantar toda a base de código.
- Os clientes que desejam usar AWS AppConfig para gerenciar implantações de configuração, mas estão limitados porque AWS AppConfig não oferecem suporte ao código ou armazenamento de configuração atual, agora têm opções adicionais. CodePipeline suporta AWS CodeCommit, GitHub, e BitBucket (para citar alguns).

> **Note**
>
> AWS AppConfig a integração com só CodePipeline é suportada Regiões da AWS onde CodePipeline está disponível.

Como funciona a integração

Você começa instalando e configurando. CodePipeline Isso inclui adicionar sua configuração a um armazenamento CodePipeline de código compatível. Em seguida, você configura seu AWS AppConfig ambiente executando as seguintes tarefas:

- Crie um namespace e um perfil de configuração
- Escolha uma estratégia de implantação predefinida ou crie sua própria

Depois de concluir essas tarefas, você cria um pipeline CodePipeline que especifica AWS AppConfig como o provedor de implantação. Em seguida, você pode fazer uma alteração na sua configuração e enviá-la para o seu armazenamento de CodePipeline código. O upload da nova configuração inicia automaticamente uma nova implantação em CodePipeline. Depois da conclusão da implantação, você pode verificar as alterações. Para obter informações sobre a criação de um pipeline que especifica AWS AppConfig como provedor de implantação, consulte Tutorial: Criar um pipeline usado AWS AppConfig como provedor de implantação no Guia do AWS CodePipeline usuário.

Recuperação de sinalizadores de atributos e dados de configuração no AWS AppConfig

Seu aplicativo recupera sinalizadores de recursos e dados de configuração de formato livre estabelecendo uma sessão de configuração usando o serviço de AWS AppConfig dados. Se você usar um dos métodos de recuperação simplificados descritos nesta seção, a extensão do Agent Lambda AWS AppConfig ou o AWS AppConfig Agent gerencia uma série de chamadas de API e tokens de sessão em seu nome. Você configura o AWS AppConfig Agente como um host local e faz com que o agente faça uma pesquisa AWS AppConfig para atualizações de configuração. O agente chama as ações da API de StartConfigurationsessão e GetLatestconfiguração e armazena seus dados de configuração em cache localmente. Para recuperar os dados, seu aplicativo faz uma chamada HTTP para o servidor localhost. AWS AppConfig O agente oferece suporte a vários casos de uso, conforme descrito emMétodos de recuperação simplificados.

Se preferir, você pode chamar manualmente essas ações da API para recuperar uma configuração. O processo da API funciona da seguinte maneira:

Seu aplicativo estabelece uma sessão de configuração usando a ação `StartConfigurationSession` da API. O cliente da sua sessão então faz chamadas periódicas para `GetLatestConfiguration` para verificar e recuperar os dados mais recentes disponíveis.

Ao chamar`StartConfigurationSession`, seu código envia identificadores (ID ou nome) de um AWS AppConfig aplicativo, ambiente e perfil de configuração que a sessão rastreia.

Em resposta, AWS AppConfig fornece um `InitialConfigurationToken` a ser entregue ao cliente da sessão e usado na primeira vez em que ele liga `GetLatestConfiguration` para essa sessão.

Ao chamar `GetLatestConfiguration`, seu código de cliente envia o valor `ConfigurationToken` mais recente que ele tem e recebe em resposta:

- `NextPollConfigurationToken`: o valor `ConfigurationToken` a ser usado na próxima chamada para `GetLatestConfiguration`.
- A configuração: os dados mais recentes destinados à sessão. Este pode estar vazio se o cliente já tiver a versão mais recente da configuração.

Esta seção inclui as seguintes informações:

Conteúdo

Sobre o serviço AWS AppConfig de plano de dados

Em 18 de novembro de 2021, AWS AppConfig lançou um novo serviço de plano de dados. Esse serviço substitui o processo anterior de recuperação de dados de configuração usando a ação `GetConfiguration` da API. O serviço de plano de dados usa duas novas ações de API, StartConfigurationSessão e GetLatestConfiguração. O serviço de plano de dados também usa novos endpoints.

Se você começou a usar AWS AppConfig antes de 28 de janeiro de 2022, o serviço pode estar chamando a ação da `GetConfiguration` API diretamente ou pode estar usando um cliente fornecido por AWS, como a extensão do AWS AppConfig Agente Lambda, para chamar essa ação da API. Se você chamar a ação `GetConfiguration` da API diretamente, tome medidas para usar as ações `StartConfigurationSession` e `GetLatestConfiguration` da API. Se você estiver usando a extensão do AWS AppConfig Agent Lambda, consulte a seção intitulada Como essa alteração afeta a extensão do Agent AWS AppConfig Lambda posteriormente neste tópico.

As novas ações da API do plano de dados oferecem os seguintes benefícios em relação à ação `GetConfiguration` da API, que agora está obsoleta.

1. Não é necessário gerenciar um parâmetro `ClientID`. Com o serviço de plano de dados, `ClientID` é gerenciado internamente pelo token de sessão criado por `StartConfigurationSession`.
2. Você não precisa mais incluir `ClientConfigurationVersion` para indicar a versão em cache dos seus dados de configuração. Com o serviço de plano de dados, `ClientConfigurationVersion` é gerenciado internamente pelo token de sessão criado por `StartConfigurationSession`.
3. O novo endpoint especial para chamadas da API do plano de dados melhora a estrutura do código ao separar as chamadas do ambiente de gerenciamento e do plano de dados.
4. O novo serviço de plano de dados melhora a extensibilidade futura para operações de plano de dados. Ao utilizar uma sessão de configuração que gerencia a recuperação de dados de configuração, a equipe do AWS AppConfig pode criar aprimoramentos mais poderosos no futuro.

Migração do **`GetConfiguration`** para o **`GetLatestConfiguration`**

Para começar a usar o novo serviço de plano de dados, você precisa atualizar seu código que chama a ação `GetConfiguration` da API. Para iniciar uma sessão de configuração, use a ação `StartConfigurationSession` da API e, em seguida, chame a ação `GetLatestConfiguration` da API para recuperar os dados de configuração. Para melhorar o desempenho, recomendamos armazenar em cache os dados de configuração localmente. Para ter mais informações, consulte Recuperação de configurações chamando diretamente as APIs.

Como essa alteração afeta a extensão do AWS AppConfig Agent Lambda

Essa alteração não tem impacto direto no funcionamento da extensão do AWS AppConfig Agente Lambda. Versões mais antigas da extensão do AWS AppConfig Agent Lambda chamavam de ação de `GetConfiguration` API em seu nome. As versões mais recentes chamam as ações da API do plano de dados. Se você estiver usando a extensão do Lambda do AWS AppConfig , recomendamos atualizar sua extensão para o nome do recurso da Amazon (ARN) mais recente e atualizar as permissões para as novas chamadas da API. Para ter mais informações, consulte Recuperação de dados de configuração usando a extensão AWS AppConfig Agent Lambda.

Métodos de recuperação simplificados

AWS AppConfig oferece vários métodos simplificados para recuperar dados de configuração. Se você usar sinalizadores de AWS AppConfig recursos ou dados de configuração de formato livre em uma AWS Lambda função, poderá usar a extensão AWS AppConfig Agent Lambda para recuperar as configurações. Se você tiver aplicativos em execução nas instâncias do Amazon EC2, poderá usar o AWS AppConfig Agent para recuperar as configurações. AWS AppConfig O Agent também oferece suporte a aplicativos executados em imagens de contêiner do Amazon Elastic Kubernetes Service (Amazon EKS) ou do Amazon Elastic Container Service (Amazon ECS).

Depois de concluir a configuração inicial, esses métodos de recuperação de dados de configuração são mais simples do que chamar AWS AppConfig diretamente as APIs. Eles implementam automaticamente as melhores práticas e podem reduzir seu custo de uso AWS AppConfig como resultado de menos chamadas de API para recuperar configurações.

Tópicos

- Recuperação de dados de configuração usando a extensão AWS AppConfig Agent Lambda
- Recuperação de dados de configuração de instâncias do Amazon EC2
- Recuperação de dados de configuração do Amazon ECS e do Amazon EKS

- Recursos adicionais de recuperação
- AWS AppConfig Agente de desenvolvimento local

Recuperação de dados de configuração usando a extensão AWS AppConfig Agent Lambda

Uma AWS Lambda extensão é um processo complementar que aumenta os recursos de uma função Lambda. Uma extensão pode ser iniciada antes de uma função ser invocada, executada paralelamente a uma função e continuar sendo executada após o processamento de uma invocação da função. Em essência, uma extensão do Lambda é como um cliente que é executado em paralelo a uma invocação do Lambda. Esse cliente paralelo pode interagir com a função a qualquer momento durante o seu ciclo de vida.

Se você usa sinalizadores de AWS AppConfig recursos ou outros dados de configuração dinâmica em uma função Lambda, recomendamos que você adicione a extensão AWS AppConfig Agent Lambda como uma camada à sua função Lambda. Isso simplifica a chamada de sinalizadores de recursos, e a própria extensão inclui as melhores práticas que simplificam o uso e reduzem os AWS AppConfig custos. Custos reduzidos resultam de menos chamadas de API para o AWS AppConfig serviço e menores tempos de processamento da função Lambda. Para obter mais informações sobre extensões do Lambda, consulte Extensões do Lambda no Guia do desenvolvedor do AWS Lambda .

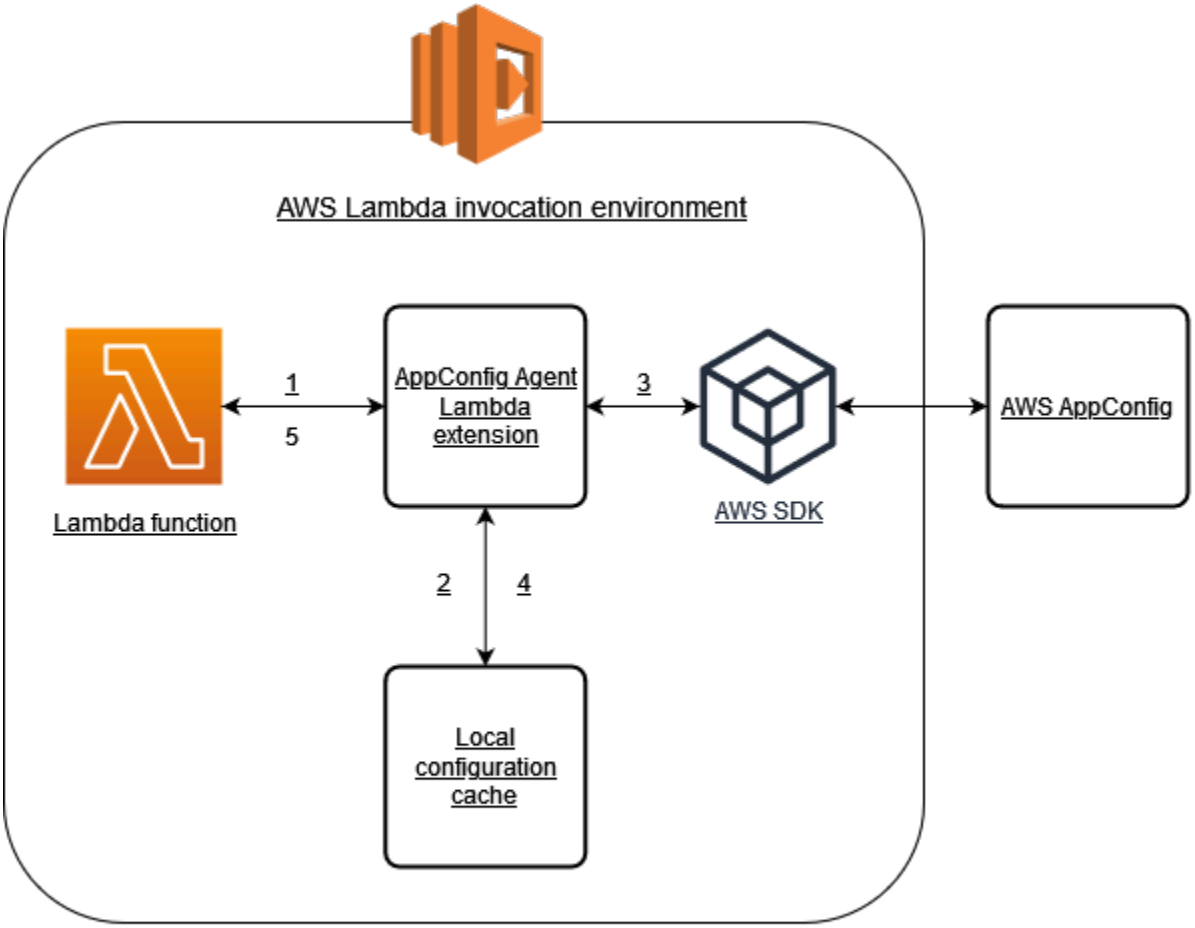
> **Note**
>
> AWS AppConfig o preço é baseado no número de vezes que uma configuração é chamada e recebida. Os custos aumentarão se o Lambda realizar várias inicializações a frio e recuperar novos dados de configuração com frequência.

Este tópico inclui informações sobre a extensão do AWS AppConfig Agent Lambda e o procedimento de como configurar a extensão para funcionar com sua função Lambda.

Como funciona

Se você usa AWS AppConfig para gerenciar configurações para uma função do Lambda sem extensões do Lambda, deve configurar sua função do Lambda para receber atualizações de configuração por meio da integração com as ações da API de sessão e configuração. StartConfiguration GetLatest

A integração da extensão do AWS AppConfig Agent Lambda com sua função Lambda simplifica esse processo. A extensão se encarrega de chamar o AWS AppConfig serviço, gerenciar um cache local de dados recuperados, rastrear os tokens de configuração necessários para as próximas chamadas de serviço e verificar periodicamente as atualizações de configuração em segundo plano. O diagrama a seguir mostra como funciona.



1. Você configura a extensão do AWS AppConfig Agent Lambda como uma camada da sua função Lambda.
2. Para acessar seus dados de configuração, sua função chama a AWS AppConfig extensão em um endpoint HTTP em `localhost:2772` execução.
3. A extensão mantém um cache local dos dados de configuração. Se os dados não estiverem no cache, a extensão chamará AWS AppConfig para obter os dados de configuração.
4. Ao receber a configuração do serviço, a extensão a armazena no cache local e a transmite para a função do Lambda.

5. AWS AppConfig A extensão Agent Lambda verifica periodicamente se há atualizações em seus dados de configuração em segundo plano. Sempre que sua função do Lambda é invocada, a extensão verifica o tempo decorrido desde que recuperou uma configuração. Se o tempo decorrido for maior que o intervalo de pesquisa configurado, a extensão liga AWS AppConfig para verificar os dados recém-implantados, atualiza o cache local se houver alguma alteração e redefine o tempo decorrido.

> **Note**
>
> - O Lambda cria instâncias separadas correspondentes ao nível de simultaneidade requerido por sua função. Cada instância é isolada e mantém o próprio cache local dos dados de configuração. Para obter mais informações sobre instâncias e simultaneidade do Lambda, consulte Gerenciamento da simultaneidade para uma função do Lambda.
> - O tempo necessário para que uma alteração de configuração apareça em uma função Lambda, após a implantação de uma configuração atualizada AWS AppConfig, depende da estratégia de implantação usada para a implantação e do intervalo de pesquisa configurado para a extensão.

Antes de começar

Antes de ativar a extensão do AWS AppConfig Agent Lambda, faça o seguinte:

- Organize as configurações em sua função do Lambda para que você possa externalizá-las no AWS AppConfig.
- Crie AWS AppConfig artefatos e dados de configuração, incluindo sinalizadores de recursos ou dados de configuração de formato livre. Para ter mais informações, consulte Criação de sinalizadores de recursos e dados de configuração de formato livre no AWS AppConfig.
- Adicione `appconfig:StartConfigurationSession` e `appconfig:GetLatestConfiguration` à política AWS Identity and Access Management (IAM) usada pela função de execução da função Lambda. Para obter mais informações, consulte perfil do IAM para execução do AWS Lambda no Guia do desenvolvedor do AWS Lambda . Para obter mais informações sobre permissões do AWS AppConfig , consulte Ações, recursos e chaves de condição do AWS AppConfig na Referência de autorização do serviço.

Adicionando a extensão AWS AppConfig Agent Lambda

Para usar a extensão AWS AppConfig Agent Lambda, você precisa adicionar a extensão ao seu Lambda. Isso pode ser feito adicionando a extensão do AWS AppConfig Agent Lambda à sua função Lambda como uma camada ou habilitando a extensão em uma função Lambda como uma imagem de contêiner.

> **Note**
>
> A AWS AppConfig extensão é independente do tempo de execução e oferece suporte a todos os tempos de execução.

Adição da extensão do Lambda do AWS AppConfig Agent usando uma camada e um ARN

Para usar a extensão AWS AppConfig Agent Lambda, você adiciona a extensão à sua função Lambda como uma camada. Para obter informações sobre como adicionar uma camada à sua função, consulte Configuração de extensões no Guia do desenvolvedor do AWS Lambda . O nome da extensão no AWS Lambda console é AWS- AppConfig -Extension. Observe também que, ao adicionar a extensão como uma camada ao Lambda, é necessário especificar um nome do recurso da Amazon (ARN). Escolha um ARN em uma das listas a seguir que corresponda à plataforma e Região da AWS onde você criou o Lambda.

- Plataforma x86-64
- Plataforma ARM64

Se quiser testar a extensão antes de adicioná-la à sua função, você pode verificar se ela funciona usando o exemplo de código a seguir.

```
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
```

Para testá-la, crie uma nova função do Lambda para Python, adicione a extensão e execute a função do Lambda. Depois de executar a função Lambda, a função AWS AppConfig Lambda retorna a configuração especificada para o caminho http://localhost:2772. Para obter informações sobre como criar uma função do Lambda, consulte Criar uma função do Lambda com o console, no Guia do desenvolvedor do AWS Lambda .

Para adicionar a extensão do AWS AppConfig Agent Lambda como uma imagem de contêiner, consulte. Usando uma imagem de contêiner para adicionar a extensão do AWS AppConfig Agent Lambda

Configuração da extensão do Lambda do AWS AppConfig Agent

Você pode configurar a extensão alterando as seguintes variáveis de AWS Lambda ambiente. Para obter mais informações, consulte Usando variáveis de AWS Lambda ambiente no Guia do AWS Lambda desenvolvedor.

Pré-busca de dados de configuração

A variável de ambiente `AWS_APPCONFIG_EXTENSION_PREFETCH_LIST` pode melhorar o tempo de inicialização da sua função. Quando a extensão do AWS AppConfig Agent Lambda é inicializada, ela recupera a configuração especificada antes de o AWS AppConfig Lambda começar a inicializar sua função e invocar seu manipulador. Em alguns casos, os dados de configuração já estão disponíveis no cache local antes que sua função os solicite.

Para usar o recurso de pré-busca, defina o valor da variável de ambiente para o caminho correspondente aos seus dados de configuração. Por exemplo, se sua configuração corresponder a um aplicativo, ambiente e perfil de configuração denominados respectivamente “my_application”, “my_environment” e “my_configuration_data”, o caminho será `/applications/my_application/environments/my_environment/configurations/my_configuration_data`. Você pode especificar vários itens de configuração listando-os como uma lista separada por vírgulas (se você tiver um nome de recurso que inclua uma vírgula, use o valor do ID do recurso em vez do nome).

Acesso a dados de configuração a partir de outra conta

A extensão do AWS AppConfig Agent Lambda pode recuperar dados de configuração de outra conta especificando uma função do IAM que concede permissões aos dados. Para configurá-la, siga estas etapas:

1. Na conta usada para gerenciar os dados de configuração, crie uma função com uma política de confiança que conceda à conta que executa a função Lambda acesso às

`appconfig:GetLatestConfiguration` ações `appconfig:StartConfigurationSession` e, junto com os ARNs parciais ou completos correspondentes aos AWS AppConfig recursos de configuração. AWS AppConfig

2. Na conta que executa a função do Lambda, adicione a variável de ambiente `AWS_APPCONFIG_EXTENSION_ROLE_ARN` à função do Lambda com o ARN da função criada na etapa 1.
3. (Opcional) Se necessário, um ID externo pode ser especificado usando a variável de ambiente `AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`. Da mesma forma, um nome de sessão pode ser configurado usando a variável de ambiente `AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`.

> **Note**
>
> Observe as seguintes informações:
>
> - A extensão do AWS AppConfig Agent Lambda só pode recuperar dados de uma conta. Se você especificar um perfil do IAM, a extensão não poderá recuperar dados de configuração da conta na qual a função do Lambda está sendo executada.
> - AWS Lambda registra informações sobre a extensão do AWS AppConfig Agent Lambda e a função Lambda usando o Amazon Logs. CloudWatch

Variável de ambiente	Detalhes	Valor padrão
`AWS_APPCONFIG_EXTENSION_HTTP_PORT`	Esta variável de ambiente especifica a porta na qual é executado o servidor HTTP local que hospeda a extensão.	2772
`AWS_APPCONFIG_EXTENSION_LOG_LEVEL`	Essa variável de ambiente especifica quais registros AWS AppConfig específicos da extensão são enviados ao Amazon CloudWatch Logs para uma função. Os valores válidos que não diferenci	`info`

Variável de ambiente	Detalhes	Valor padrão
	am maiúsculas e minúscula s são: `debug`, `info`, `warn`, `error` e `none`. A depuração inclui informações detalhada s, inclusive informações de tempo, sobre a extensão.	
`AWS_APPCONFIG_EXTE NSION_MAX_CONNECTI ONS`	Esta variável de ambiente configura o número máximo de conexões que a extensão usa para recuperar as configurações do AWS AppConfig.	3
`AWS_APPCONFIG_EXTE NSION_POLL_INTERVA L_SECONDS`	Essa variável de ambiente controla a frequência com que a extensão pesquisa AWS AppConfig uma configuração atualizada em segundos.	45
`AWS_APPCONFIG_EXTE NSION_POLL_TIMEOUT _MILLIS`	Essa variável de ambiente controla o tempo máximo, em milissegundos, do qual a extensão espera por uma resposta AWS AppConfig ao atualizar os dados no cache. Se AWS AppConfig não responder no período de tempo especificado, a extensão ignora esse intervalo de pesquisa e retorna os dados em cache atualizados anteriormente.	3000

Variável de ambiente	Detalhes	Valor padrão
AWS_APPCONFIG_EXTENSION_PREFETCH_LIST	Esta variável de ambiente especifica os dados de configuração que a extensão começa a recuperar antes que a função seja inicializada e o manipulador seja executado. Ela pode reduzir significativamente o tempo de inicialização a frio da função.	Nenhum
AWS_APPCONFIG_EXTENSION_PROXY_HEADERS	Esta variável de ambiente especifica os cabeçalhos exigidos pelo proxy referenciado na variável de ambiente AWS_APPCONFIG_EXTENSION_PROXY_URL. O valor é uma lista de cabeçalhos separados por vírgula. Cada cabeçalho usa o formato a seguir: `"header: value"`	Nenhum
AWS_APPCONFIG_EXTENSION_PROXY_URL	Essa variável de ambiente especifica a URL do proxy a ser usada para conexões da AWS AppConfig extensão a. Serviços da AWSHTTPSe HTTP URLs são compatíveis.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`AWS_APPCONFIG_EXTENSION_ROLE_ARN`	Essa variável de ambiente especifica o ARN da função do IAM correspondente a uma função que deve ser assumida pela extensão para AWS AppConfig recuperar a configuração.	Nenhum
`AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID`	Esta variável de ambiente especifica o ID externo a ser usado em conjunto com o ARN da função assumida.	Nenhum
`AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME`	Esta variável de ambiente especifica o nome da sessão a ser associado às credenciais do perfil do IAM assumido.	Nenhum
`AWS_APPCONFIG_EXTENSION_SERVICE_REGION`	Essa variável de ambiente especifica uma região alternativa que a extensão deve usar para chamar o AWS AppConfig serviço. Quando indefinida, a extensão usa o endpoint na região atual.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`AWS_APPCONFIG_EXTENSION_MANIFEST`	Essa variável de ambiente configura o AWS AppConfig Agente para aproveitar os recursos adicionais por configuração, como recuperações de várias contas e salvamento da configuração em disco. Você pode inserir um dos seguintes valores: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Para obter mais informações sobre esses recursos, consulte Recursos adicionais de recuperação.	verdadeiro
`AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST`	Essa variável de ambiente configura o AWS AppConfig Agente para esperar até que o manifesto seja processado antes de concluir a inicialização.	verdadeiro

Recuperação de um ou mais sinalizadores de uma configuração de sinalizadores de atributos

Para configurações de sinalizadores de atributos (configurações do tipo `AWS.AppConfig.FeatureFlags`), a extensão do Lambda permite que você recupere um único sinalizador ou um subconjunto de sinalizadores em uma configuração. A recuperação de um ou dois

sinalizadores é útil se o Lambda precisar usar apenas alguns sinalizadores do perfil de configuração. Os exemplos a seguir usam Python.

> **Note**
>
> A capacidade de chamar um único sinalizador de recurso ou um subconjunto de sinalizadores em uma configuração só está disponível na versão 2.0.45 e superior AWS AppConfig da extensão do Agent Lambda.

Você pode recuperar dados AWS AppConfig de configuração de um endpoint HTTP local. Para acessar um sinalizador específico ou uma lista de sinalizadores, use o parâmetro de consulta ? `flag=`*`flag_name`* para um perfil de configuração do AWS AppConfig .

Para acessar um único sinalizador e seus atributos

```
import urllib.request

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

Para acessar vários sinalizadores e seus atributos

```
import urllib.request

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

Visualizando os AWS AppConfig registros da extensão do Agent Lambda

Você pode visualizar os dados de registro da extensão do AWS AppConfig Agent Lambda AWS Lambda nos registros. As entradas de registro são prefaciadas com`appconfig agent`. Aqui está um exemplo.

```
[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)
```

Versões disponíveis da extensão AWS AppConfig Agent Lambda

Este tópico inclui informações sobre as versões da extensão AWS AppConfig do Agent Lambda. A extensão do Lambda do AWS AppConfig Agent oferece suporte às funções do Lambda desenvolvidas para as plataformas x86-64 e ARM64 (Graviton2). Para funcionar corretamente, sua função Lambda deve ser configurada para usar o Amazon Resource Name (ARN) específico para o Região da AWS local em que está atualmente hospedada. Você pode visualizar Região da AWS os detalhes do ARN posteriormente nesta seção.

Important

Observe os seguintes detalhes importantes sobre a extensão do AWS AppConfig Agent Lambda.

- A ação `GetConfiguration` da API tornou-se obsoleta em 28 de janeiro de 2022. Em vez disso, as chamadas para receber dados de configuração devem usar as APIs `StartConfigurationSession` e `GetLatestConfiguration`. Se você estiver usando uma versão da extensão do AWS AppConfig Agent Lambda criada antes de 28 de janeiro de 2022, talvez seja necessário configurar a permissão para as novas APIs. Para ter mais informações, consulte Sobre o serviço AWS AppConfig de plano de dados.
- AWS AppConfig suporta todas as versões listadas emVersões de extensão mais antigas. É recomendável atualizar periodicamente para a versão mais recente para aproveitar os aprimoramentos das extensões.

Tópicos

- Versões de extensão mais antigas

Notas de versão da extensão do Lambda do AWS AppConfig Agent

A tabela a seguir descreve as alterações feitas nas versões recentes da extensão AWS AppConfig Lambda.

Version (Versão)	Data de lançamento	Observações
2.0.358	12/01/2023	Foi adicionado suporte para os seguintes recursos de recuperação: • Recuperação de várias contas: use o AWS AppConfig Agente de uma conta primária ou de recuperação Conta da AWS para recuperar dados de configuração de várias contas de fornecedores. • Gravar cópia da configura ção no disco: use o AWS AppConfig Agente para gravar dados de configura ção no disco. Esse recurso permite que os clientes com aplicativos que leem dados de configuração do disco se integrem AWS AppConfig.
2.0.181	14/08/2023	Foi adicionado suporte para o Região da AWS il-central-1 de Israel (Tel Aviv).
2.0.165	21/02/2023	Correções de erros secundári as. Não restringe mais o uso da extensão a versões

Version (Versão)	Data de lançamento	Observações
		específicas de tempo de execução por meio do AWS Lambda console. Suporte adicionado para as seguintes Regiões da AWS: • Oriente Médio (EAU) me-central-1 • Ásia-Pacífico (Hyderabad) ap-south-2 • Ásia-Pacífico (Melbourne) ap-southeast-4 • Europa (Espanha) eu-south-2 • Europa (Zurique) eu-central-2
2.0.122	23/08/2022	Foi adicionado suporte para um proxy de tunelamento, que pode ser configurado com as variáveis de ambiente `AWS_APPCONFIG_EXTENSION_PROXY_URL` e `AWS_APPCONFIG_EXTENSION_PROXY_HEADERS`. Foi adicionado o .NET 6 como runtime. Para obter mais informações sobre variáveis de ambiente, consulte Configuração da extensão do Lambda do AWS AppConfig Agent.

Version (Versão)	Data de lançamento	Observações
2.0.58	05/03/2022	Suporte aprimorado para processadores Graviton2 (ARM64) no Lambda.
2.0.45	15/03/2022	Foi adicionado suporte para chamar um único sinalizador de atributos. Anteriormente, os clientes chamavam sinalizadores de atributos agrupados em um perfil de configuração e precisavam analisar a resposta do lado do cliente. Com essa versão, os clientes podem usar um parâmetro `flag=<flag-name>` ao chamar o endpoint de localhost HTTP para obter o valor de um único sinalizador. Foi também adicionado suporte inicial para processadores Graviton2 (ARM64).

Como encontrar o número da versão da sua extensão do Lambda

Use o procedimento a seguir para localizar o número da versão da sua extensão do AWS AppConfig Agente Lambda atualmente configurada. Para funcionar corretamente, sua função Lambda deve ser configurada para usar o Amazon Resource Name (ARN) específico para o Região da AWS local em que está atualmente hospedada.

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a função do Lambda na qual deseja adicionar a camada `AWS-AppConfig-Extension`.
3. Na seção Camadas, escolha Adicionar uma camada.

4. Na seção Escolha uma camada, escolha AWS- AppConfig -Extensão na lista de AWS camadas.
5. Use a lista Versão para escolher um número de versão.
6. Escolha Adicionar.
7. Use a guia Testar para testar a função.
8. Depois que o teste for concluído, visualize a saída de logs. Localize a versão da extensão do AWS AppConfig Agent Lambda na seção Detalhes da execução. Essa versão deve corresponder aos URLs necessários para essa versão.

Plataforma x86-64

Ao adicionar a extensão como uma camada ao seu Lambda, você deve especificar um ARN. Escolha um ARN na tabela a seguir que corresponda ao local em Região da AWS que você criou o Lambda. Esses ARNs são para funções do Lambda desenvolvidas para a plataforma x86-64.

Versão 2.0.358

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93`

Região	ARN
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106`
Europa (Zurique)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93`
Europa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98`
Europa (Estocolmo)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159`
Europa (Milão)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83`
Europa (Espanha)	`arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44`
China (Pequim)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:76`

Região	ARN
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:76`
Ásia-Pacífico (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98`
Ásia-Pacífico (Seul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108`
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79`
Ásia-Pacífico (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20`

Região	ARN
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107`
Ásia-Pacífico (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:83`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:22`
Oriente Médio (Emirados Árabes Unidos)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54`

Plataforma ARM64

Ao adicionar a extensão como uma camada ao seu Lambda, você deve especificar um ARN. Escolha um ARN na tabela a seguir que corresponda ao local em Região da AWS que você criou o Lambda. Esses ARNs são para funções do Lambda desenvolvidas para a plataforma ARM64.

Versão 2.0.358

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49`
Europa (Zurique)	`arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5`

Região	ARN
Europa (Irlanda)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63
Europa (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45
Europa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17
Europa (Estocolmo)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18
Europa (Milão)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11
Europa (Espanha)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5
Ásia-Pacífico (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11
Ásia-Pacífico (Tóquio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51
Ásia-Pacífico (Seul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16

Região	ARN
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16`
Ásia-Pacífico (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49`
Ásia-Pacífico (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11`

Região	ARN
Oriente Médio (Emirados Árabes Unidos)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:5`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:13`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:5`

Versões de extensão mais antigas

Esta seção lista os ARNs e Regiões da AWS as versões mais antigas da extensão AWS AppConfig Lambda. Esta lista não contém informações de todas as versões anteriores da extensão do Lambda do AWS AppConfig Agent, mas será atualizada quando novas versões forem lançadas.

Versões de extensão mais antigas (plataforma x86-64)

As tabelas a seguir listam os ARNs e as Regiões da AWS versões mais antigas da extensão AWS AppConfig Agent Lambda desenvolvida para a plataforma x86-64.

Data substituída pela nova extensão: 12/01/2023

Versão 2.0.181

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:113`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:81`

Região	ARN
Oeste dos EUA (N. da Califórnia)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124
Oeste dos EUA (Oregon)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146
Canadá (Central)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81
Europa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93
Europa (Zurique)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32
Europa (Irlanda)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110
Europa (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81
Europa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82
Europa (Estocolmo)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142

Região	ARN
Europa (Milão)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73
Europa (Espanha)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29
China (Pequim)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:68
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:68
Ásia-Pacífico (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73
Ásia-Pacífico (Tóquio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84
Ásia-Pacífico (Seul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93
Asia Pacific (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86
Ásia-Pacífico (Singapura)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91

Região	ARN
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64`
Ásia-Pacífico (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94`
Ásia-Pacífico (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:73`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:7`
Oriente Médio (Emirados Árabes Unidos)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34`

Região	ARN
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46`

Data de substituição pela nova extensão: 14/08/2023

Versão 2.0.165

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143`

Região	ARN
Canadá (Central)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79
Europa (Frankfurt)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91
Europa (Zurique)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29
Europa (Irlanda)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108
Europa (Londres)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79
Europa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80
Europa (Estocolmo)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139
Europa (Milão)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71
Europa (Espanha)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26

Região	ARN
China (Pequim)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:66`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:66`
Ásia-Pacífico (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82`
Ásia-Pacífico (Seul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91`
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60`

Região	ARN
Ásia-Pacífico (Melbourne)	`arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92`
Ásia-Pacífico (Hyderabad)	`arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71`
Oriente Médio (Emirados Árabes Unidos)	`arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44`

Data de substituição pela nova extensão: 21/02/2023

Versão 2.0.122

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59`

Região	ARN
Europa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60`
Europa (Estocolmo)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111`
Europa (Milão)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54`
China (Pequim)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:52`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:52`
Ásia-Pacífico (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62`
Ásia-Pacífico (Seul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70`
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59`

Região	ARN
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29`

Data de substituição pela nova extensão: 23/08/2022

Versão 2.0.58

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50`

Região	ARN
Europa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51
Europa (Estocolmo)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98
Europa (Milão)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47
China (Pequim)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46
Ásia-Pacífico (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47
Ásia-Pacífico (Tóquio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49
Ásia-Pacífico (Seul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59
Asia Pacific (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46

Região	ARN
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23`

Data de substituição pela nova extensão: 21/04/2022

Versão 2.0.45

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49`

Região	ARN
Europa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50`
Europa (Estocolmo)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97`
Europa (Milão)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46`
China (Pequim)	`arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45`
China (Ningxia)	`arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45`
Ásia-Pacífico (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48`
Ásia-Pacífico (Seul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58`
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45`

Região	ARN
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22`

Data de substituição pela nova extensão: 15/03/2022

Versão 2.0.30

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47`

Região	ARN
Europa (Paris)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48
Europa (Estocolmo)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86
Europa (Milão)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44
China (Pequim)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43
Ásia-Pacífico (Hong Kong)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44
Ásia-Pacífico (Tóquio)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45
Ásia-Pacífico (Osaka)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42
Ásia-Pacífico (Seul)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54

Região	ARN
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44`
AWS GovCloud (Leste dos EUA)	`arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20`
AWS GovCloud (Oeste dos EUA)	`arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20`

Versões de extensão mais antigas (plataforma ARM64)

As tabelas a seguir listam os ARNs e Regiões da AWS as versões mais antigas da extensão AWS AppConfig Agent Lambda desenvolvida para a plataforma ARM64.

Data substituída pela nova extensão: 12/01/2023

Versão 2.0.181

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:46`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:33`
Oeste dos EUA (N. da Califórnia)	`arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:1`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:48`
Canadá (Central)	`arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:36`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:48`

Região	ARN
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33`
Europa (Paris)	`arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Estocolmo)	`arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Milão)	`arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Hong Kong)	`arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37`
Ásia-Pacífico (Seul)	`arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1`
Asia Pacific (Osaka)	`arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43`

Região	ARN
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36`
Ásia-Pacífico (Jacarta)	`arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36`
América do Sul (São Paulo)	`arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1`
África (Cidade do Cabo)	`arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1`
Oriente Médio (Barém)	`arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1`

Data de substituição pela nova extensão: 30/03/2023

Versão 2.0.165

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43`

Região	ARN
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34`

Data de substituição pela nova extensão: 21/02/2023

Versão 2.0.122

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:15`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:11`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:16`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:13`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:20`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:11`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:15`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:16`

Região	ARN
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:13`

Data de substituição pela nova extensão: 23/08/2022

Versão 2.0.58

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2`
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7`

Região	ARN
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2`

Data de substituição pela nova extensão: 21/04/2022

Versão 2.0.45

Região	ARN
Leste dos EUA (Norte da Virgínia)	`arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1`
Leste dos EUA (Ohio)	`arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1`

Região	ARN
Oeste dos EUA (Oregon)	`arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2`
Europa (Frankfurt)	`arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1`
Europa (Irlanda)	`arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6`
Europa (Londres)	`arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Tóquio)	`arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Singapura)	`arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2`
Ásia-Pacífico (Sydney)	`arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1`
Ásia-Pacífico (Mumbai)	`arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1`

Usando uma imagem de contêiner para adicionar a extensão do AWS AppConfig Agent Lambda

Você pode empacotar sua extensão do AWS AppConfig Agent Lambda como uma imagem de contêiner para carregá-la em seu registro de contêiner hospedado no Amazon Elastic Container Registry (Amazon ECR).

Para adicionar a extensão AWS AppConfig Agent Lambda como uma imagem de contêiner Lambda

1. Insira o Região da AWS e o Amazon Resource Name (ARN) no AWS Command Line Interface (AWS CLI) conforme mostrado abaixo. Substitua o valor da região e do ARN pela sua região e pelo ARN correspondente para recuperar uma cópia da camada Lambda. AWS AppConfig fornece ARNs para plataformas x86-64 e ARM64.

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

   Aqui está um exemplo.

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

   A resposta é semelhante à seguinte:

   ```
   {
     "LayerVersionArn": "arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-
   Extension:128",
     "Description": "AWS AppConfig extension: Use dynamic configurations deployed via
    AWS AppConfig for your AWS Lambda functions",
     "CreatedDate": "2021-04-01T02:37:55.339+0000",
     "LayerArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension",

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

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

2. Na resposta acima, o valor retornado para `Location` é o URL do bucket do Amazon Simple Storage Service (Amazon S3) que contém a extensão do Lambda. Cole o URL em seu navegador para baixar o arquivo .zip da extensão do Lambda.

> **Note**
>
> O URL do bucket do Amazon S3 fica disponível por apenas 10 minutos.

(Opcional) Como alternativa, você também pode usar o comando `curl` a seguir para baixar a extensão do Lambda.

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

(Opcional) Como alternativa, você pode combinar a Etapa 1 e a Etapa 2 para recuperar o ARN e baixar o arquivo de extensão .zip de uma só vez.

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

3. Adicione as linhas a seguir ao seu `Dockerfile` para adicionar a extensão à imagem do contêiner.

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

4. Certifique-se de que a função de execução da função Lambda tenha o conjunto de permissões appconfig:. GetConfiguration

Exemplo

Esta seção inclui um exemplo para habilitar a extensão do AWS AppConfig Agent Lambda em uma função Python Lambda baseada em imagem de contêiner.

1. Crie um `Dockerfile` semelhante ao seguinte:

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

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

2. Baixe a camada de extensão no mesmo diretório do `Dockerfile`.

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

3. Crie um arquivo Python nomeado `index.py` no mesmo diretório do `Dockerfile`.

```
import urllib.request

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

def get_configuration(app, env, config):
    url = f'http://localhost:2772/applications/{app}/environments/{env}/
configurations/{config}'
```

```
    return urllib.request.urlopen(url).read()
```

4. Execute as seguintes etapas para criar a imagem do `docker` e carregá-la no Amazon ECR.

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

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

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

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

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

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

5. Use a imagem do Amazon ECR que você criou acima para criar a função do Lambda. Para obter mais informações sobre uma função do Lambda como contêiner, consulte Criar uma função do Lambda definida como uma imagem de contêiner.
6. Certifique-se de que a função de execução da função Lambda tenha o conjunto de permissões appconfig:. GetConfiguration

Integração com a OpenAPI

Você pode usar a seguinte especificação YAML para que a OpenAPI crie um SDK usando uma ferramenta como o OpenAPI Generator. Você pode atualizar essa especificação para incluir valores codificados para aplicativo, ambiente ou configuração. Você também pode adicionar caminhos adicionais (se tiver vários tipos de configuração) e incluir esquemas de configuração para gerar

modelos com tipos específicos de configuração para seus clientes SDK. Para obter mais informações sobre a OpenAPI (também conhecida como Swagger), consulte a especificação da OpenAPI.

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

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

Recuperação de dados de configuração de instâncias do Amazon EC2

Você pode se integrar AWS AppConfig com aplicativos executados em suas instâncias Linux do Amazon Elastic Compute Cloud (Amazon EC2) usando o Agent. AWS AppConfig O agente aprimora o processamento e o gerenciamento de aplicativos das seguintes maneiras:

- O agente liga AWS AppConfig em seu nome usando uma função AWS Identity and Access Management (IAM) e gerenciando um cache local de dados de configuração. Ao extrair dados de configuração do cache local, seu aplicativo exige menos atualizações de código para gerenciar dados de configuração, recupera dados de configuração em milissegundos e não é afetado por problemas de rede que podem afetar as chamadas para esses dados.*
- O agente oferece uma experiência nativa para recuperar e resolver sinalizadores de AWS AppConfig recursos.
- Pronto para uso, o agente fornece as práticas recomendadas para estratégias de armazenamento em cache, intervalos de pesquisa e disponibilidade de dados de configuração locais, enquanto rastreia os tokens de configuração necessários para chamadas de serviço subsequentes.
- Durante a execução em segundo plano, o agente consulta periodicamente o plano de AWS AppConfig dados para atualizações de dados de configuração. Seu aplicativo pode recuperar os dados conectando-se ao localhost na porta 2772 (um valor de porta padrão personalizável) e chamando HTTP GET para recuperar os dados.

* O AWS AppConfig agente armazena os dados em cache na primeira vez que o serviço recupera seus dados de configuração. Por esse motivo, a primeira chamada para recuperar dados é mais lenta que as chamadas subsequentes.

Tópicos

- Etapa 5: (obrigatória) recuperação de dados de configuração
- Etapa 6 (opcional, mas recomendada): automatizar as atualizações do Agente AWS AppConfig

Etapa 1: (obrigatória) crie recursos e configure permissões

Para se integrar AWS AppConfig com aplicativos executados em suas instâncias do Amazon EC2, você deve criar AWS AppConfig artefatos e dados de configuração, incluindo sinalizadores de recursos ou dados de configuração de formato livre. Para ter mais informações, consulte Criação de sinalizadores de recursos e dados de configuração de formato livre no AWS AppConfig.

Para recuperar dados de configuração hospedados por AWS AppConfig, seus aplicativos devem ser configurados com acesso ao plano de AWS AppConfig dados. Para dar acesso aos seus aplicativos, atualize a política de permissões do IAM atribuída à função de instância do Amazon EC2. Especificamente, você deve adicionar as ações `appconfig:StartConfigurationSession` e `appconfig:GetLatestConfiguration` à política. Exemplo:

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

Para obter informações sobre como adicionar permissões a uma política, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

Etapa 2: (Obrigatório) Instalar e iniciar o AWS AppConfig agente nas instâncias do Amazon EC2

AWS AppConfig O agente está hospedado em um bucket do Amazon Simple Storage Service (Amazon S3) que é gerenciado por. AWS Use o procedimento a seguir para instalar a versão mais recente do agente em sua instância do Linux. Se seu aplicativo estiver distribuído entre várias instâncias, você deverá executar este procedimento em cada instância que hospeda o aplicativo.

> **Note**
>
> Observe as seguintes informações:
>
> - AWS AppConfig O agente está disponível para sistemas operacionais Linux que executam a versão 4.15 ou superior do kernel. Sistemas baseados em Debian, como o Ubuntu, não são suportados.
> - O agente suporta arquiteturas x86_64 e ARM64.
> - Para aplicativos distribuídos, recomendamos adicionar os comandos de instalação e inicialização aos dados de usuário do Amazon EC2 do seu grupo do Auto Scaling. Se fizer isso, cada instância executará os comandos automaticamente. Para obter mais informações, consulte Executar comandos em sua instância Linux na inicialização no Guia do usuário do Amazon EC2. Além disso, consulte Tutorial: configurar dados do usuário para recuperar o estado do ciclo de vida do destino por meio de metadados da instância no Guia do usuário do Amazon EC2 Auto Scaling.
> - Os procedimentos deste tópico descrevem como realizar ações como instalar o agente fazendo login na instância para executar o comando. Você pode executar os comandos em uma máquina cliente local e direcionar uma ou mais instâncias usando o Run Command, que é um recurso do AWS Systems Manager. Para obter mais informações, consulte Comando AWS Systems Manager Run no Guia do usuário do AWS Systems Manager .
> - AWS AppConfig O agente nas instâncias Linux do Amazon EC2 é um `systemd` serviço.

Para instalar e iniciar o AWS AppConfig Agente em uma instância

1. Faça login na sua instância do Linux.
2. Abra um terminal e execute o seguinte comando com permissões de administrador para arquiteturas x86_64:

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

   Para arquiteturas ARM64, execute o seguinte comando:

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

Se você quiser instalar uma versão específica do AWS AppConfig Agente, `latest` substitua a URL por um número de versão específico. Veja um exemplo para 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. Para iniciar o agente, execute o seguinte comando:

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

4. Execute o seguinte comando para verificar se o agente está em execução:

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

Se houver êxito, o comando retornará informações como as seguintes:

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

> **Note**
>
> Para interromper o agente, execute o seguinte comando:
>
> ```
> sudo systemctl stop aws-appconfig-agent
> ```

Etapa 3: (opcional, mas recomendado) Enviar arquivos de log para o CloudWatch Logs

Por padrão, o AWS AppConfig Agente publica registros no STDERR. O Systemd redireciona STDOUT e STDERR para todos os serviços em execução na instância Linux para o diário do systemd. Você pode visualizar e gerenciar dados de log no diário do systemd se estiver executando

o AWS AppConfig Agent em apenas uma ou duas instâncias. Uma solução melhor, uma solução altamente recomendada para aplicativos distribuídos, é gravar arquivos de log em disco e depois usar o CloudWatch agente da Amazon para carregar os dados de log na AWS nuvem. Além disso, você pode configurar o CloudWatch agente para excluir arquivos de log antigos da sua instância, o que evita que ela fique sem espaço em disco.

Para habilitar o log em disco, você deve definir a variável de ambiente `LOG_PATH`, conforme descrito em Etapa 4: (opcional) Usando variáveis de ambiente para configurar o AWS AppConfig agente para o Amazon EC2.

Para começar a usar o CloudWatch agente, consulte Coletar métricas e registros de instâncias do Amazon EC2 e servidores locais com o CloudWatch agente no Guia do usuário da Amazon CloudWatch . Você pode usar o Quick Setup, um recurso do Systems Manager para instalar rapidamente o CloudWatch agente. Para obter mais informações, consulte Gerenciamento do host de Configuração Rápida no Guia do usuário do AWS Systems Manager .

> **Warning**
>
> Se você optar por gravar arquivos de log no disco sem usar o CloudWatch agente, deverá excluir os arquivos de log antigos. AWS AppConfig O agente gira automaticamente os arquivos de log a cada hora. Se você não excluir os arquivos de log antigos, sua instância poderá ficar sem espaço em disco.

Depois de instalar o CloudWatch agente na sua instância, crie um arquivo de configuração do CloudWatch agente. O arquivo de configuração instrui o CloudWatch agente sobre como trabalhar com os arquivos de log do AWS AppConfig agente. Para obter mais informações sobre a criação de um arquivo de configuração do CloudWatch agente, consulte Criar o arquivo de configuração do CloudWatch agente.

Adicione a `logs` seção a seguir ao arquivo de configuração do CloudWatch agente na instância e salve suas alterações:

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

Se o valor de `auto_removal` for `true`, o CloudWatch agente excluirá automaticamente os arquivos de log do AWS AppConfig Agente rotacionados.

Etapa 4: (opcional) Usando variáveis de ambiente para configurar o AWS AppConfig agente para o Amazon EC2

Você pode configurar o AWS AppConfig Agente para o Amazon EC2 usando variáveis de ambiente. Para definir variáveis de ambiente para um serviço do `systemd`, crie um arquivo de unidade drop-in. O exemplo a seguir mostra como criar um arquivo de unidade drop-in para definir o nível de registro do AWS AppConfig agente como. `DEBUG`

Exemplo de como criar um arquivo de unidade drop-in para variáveis de ambiente

1. Faça login na sua instância do Linux.
2. Abra um terminal ou execute o comando a seguir com permissões de administrador. O comando cria um diretório de configuração:

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

3. Execute o comando a seguir para criar o arquivo de unidade drop-in. Substitua *`file_name`* por um nome para o arquivo. A extensão deve ser `.conf`:

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

4. Insira as informações no arquivo da unidade drop-in. O exemplo a seguir adiciona uma seção `Service` que define uma variável de ambiente. O exemplo define o nível de log do AWS AppConfig Agente como `DEBUG`.

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

5. Execute o comando a seguir para recarregar a configuração do systemd:

```
sudo systemctl daemon-reload
```

6. Execute o comando a seguir para reiniciar o AWS AppConfig Agente:

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

Você pode configurar o AWS AppConfig Agent for Amazon EC2 especificando as seguintes variáveis de ambiente em um arquivo de unidade drop-in.

Variável de ambiente	Detalhes	Valor padrão

`ACCESS_TOKEN`	Esta variável de ambiente define um token que deve ser fornecido ao solicitar dados de configuração do servidor HTTP do agente. O valor do token deve ser definido no cabeçalho de autorização da solicitação HTTP com um tipo de autorização `Bearer`. Aqui está um exemplo.
	`GET /applications/my_app/...`
	`Host: localhost:2772`
	`Authorization: Bearer <token value>`
		Nenhum

`BACKUP_DIRECTORY`	Essa variável de ambiente permite que o AWS AppConfig Agente salve um backup de cada configuração recuperada no diretório especificado.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
	Important As configurações copiadas em disco não são criptografadas. Se sua configuração contiver dados confidenciais, recomendamos AWS AppConfig que você pratique o princípio do menor privilégio com as permissões do sistema de arquivos. Para ter mais informações, consulte Segurança no AWS AppConfig.	
`HTTP_PORT`	Esta variável de ambiente especifica a porta na qual o servidor HTTP do agente é executado.	2772

Variável de ambiente	Detalhes	Valor padrão
`LOG_LEVEL`	Esta variável de ambiente especifica o nível de detalhes que o agente registra. Cada nível inclui o nível atual e todos os níveis superiores. As variáveis diferenciam maiúsculas de minúsculas. Do mais detalhado ao menos detalhado, os níveis de log são: `debug`, `info`, `warn`, `error` e `none`. O `Debug` inclui informações detalhadas, incluindo informações de tempo, sobre o agente.	`info`
`LOG_PATH`	O local do disco em que os logs são gravados. Se não especificado, os logs serão gravados em stderr.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`MANIFEST`	Essa variável de ambiente configura o AWS AppConfig Agente para aproveitar os recursos adicionais por configuração, como recuperaç ões de várias contas e salvamento da configuração em disco. Você pode inserir um dos seguintes valores: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Para obter mais informaçõ es sobre esses recursos, consulte Recursos adicionais de recuperação.	verdadeiro
`MAX_CONNECTIONS`	Esta variável de ambiente configura o número máximo de conexões que o agente usa para recuperar configurações do AWS AppConfig.	3

Variável de ambiente	Detalhes	Valor padrão
`POLL_INTERVAL`	Essa variável de ambiente controla a frequência com que o agente pesquisa dados AWS AppConfig de configuração atualizados. É possível especificar um número de segundos para o intervalo. Você também pode especificar um número com uma unidade de tempo: s para segundos, m para minutos, e h para horas. Se nenhuma unidade for especificada, o agente usará segundos como padrão. Por exemplo, 60, 60 s e 1 min resultam no mesmo intervalo de pesquisa.	45 segundos
`PREFETCH_LIST`	Essa variável de ambiente especifica os dados de configuração que o agente solicita AWS AppConfig assim que é iniciado.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
PRELOAD_BACKUPS	Se definido como`true`, o AWS AppConfig Agente carrega os backups de configuração encontrados BACKUP_DIRECTORY na memória e verifica imediatam ente se existe uma versão mais recente do serviço. Se definido como`false`, o AWS AppConfig Agente só carrega o conteúdo de um backup de configuração se não conseguir recuperar dados de configura ção do serviço, por exemplo, se houver um problema com sua rede.	verdadeiro
PROXY_HEADERS	Esta variável de ambiente especifica cabeçalhos que são exigidos pelo proxy referenci ado na variável de ambiente PROXY_URL . O valor é uma lista de cabeçalhos separados por vírgula. Cada cabeçalho usa o formato a seguir: `"header: value"`	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`PROXY_URL`	Essa variável de ambiente especifica a URL do proxy a ser usada para conexões do agente com Serviços da AWS, inclusive AWS AppConfig . `HTTPS`e `HTTP` URLs são compatíveis.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`REQUEST_TIMEOUT`	Essa variável de ambiente controla a quantidade de tempo do qual o agente espera por uma resposta. AWS AppConfig Se o serviço não responder, a solicitação falhará. Se a solicitação for para a recuperação inicial de dados, o agente retornará um erro ao seu aplicativo. Se o tempo limite ocorrer durante uma verificação de dados atualizados em segundo plano, o agente registrará o erro e tentará novamente após um pequeno atraso. Você pode especificar o número de milissegundos para o tempo limite. Você pode também especificar um número com uma unidade de tempo: ms, para milissegu ndos, e s, para segundos. Se nenhuma unidade for especificada, o agente usará milissegundos como padrão. Por exemplo, 5000, 5000 ms e 5 s resultam no mesmo valor de tempo limite da solicitação.	3000 milissegundos

Variável de ambiente	Detalhes	Valor padrão
ROLE_ARN	Essa variável de ambiente especifica o Amazon Resource Name (ARN) de uma função do IAM. AWS AppConfig O agente assume essa função para recuperar os dados de configuração.	Nenhum
ROLE_EXTERNAL_ID	Esta variável de ambiente especifica o ID externo a ser usado com o ARN da função assumida.	Nenhum
ROLE_SESSION_NAME	Esta variável de ambiente especifica o nome da sessão a ser associado às credenciais do perfil do IAM assumido.	Nenhum
SERVICE_REGION	Essa variável de ambiente especifica uma alternativa Região da AWS que o AWS AppConfig Agente usa para chamar o AWS AppConfig serviço. Se não for definida, o agente tentará determina r a região atual. Se não for possível, o agente não iniciará.	Nenhum
WAIT_ON_MANIFEST	Essa variável de ambiente configura o AWS AppConfig Agente para esperar até que o manifesto seja processad o antes de concluir a inicializ ação.	verdadeiro

Etapa 5: (obrigatória) recuperação de dados de configuração

Você pode recuperar dados de configuração do AWS AppConfig Agente usando uma chamada HTTP localhost. Os exemplos a seguir usam `curl` com um cliente HTTP. Você pode chamar o agente usando qualquer cliente HTTP disponível compatível com a linguagem do aplicativo ou com as bibliotecas disponíveis, incluindo um AWS SDK.

Para recuperar o conteúdo completo de qualquer configuração implantada

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

Para recuperar um único sinalizador e seus atributos de uma configuração do AWS AppConfig do tipo **`Feature Flag`**

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

Para acessar vários sinalizadores e seus atributos a partir de uma configuração do AWS AppConfig do tipo **`Feature Flag`**

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

Etapa 6 (opcional, mas recomendada): automatizar as atualizações do Agente AWS AppConfig

AWS AppConfig O agente é atualizado periodicamente. Para garantir que você esteja executando a versão mais recente do AWS AppConfig Agent em suas instâncias, recomendamos adicionar os comandos a seguir aos seus dados de usuário do Amazon EC2. Você pode adicionar os comandos aos dados do usuário na instância ou no grupo do Auto Scaling do EC2. O script instala e inicia a versão mais recente do agente sempre que uma instância é iniciada ou reinicializada.

```
#!/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
```

Recuperação de dados de configuração do Amazon ECS e do Amazon EKS

Você pode se integrar AWS AppConfig ao Amazon Elastic Container Service (Amazon ECS) e ao Amazon Elastic Kubernetes Service (Amazon EKS) usando o Agent. AWS AppConfig O agente funciona como um contêiner auxiliar executado junto com seus aplicativos de contêiner Amazon ECS e Amazon EKS. O agente aprimora o processamento e o gerenciamento de aplicativos em contêineres das seguintes maneiras:

- O agente liga AWS AppConfig em seu nome usando uma função AWS Identity and Access Management (IAM) e gerenciando um cache local de dados de configuração. Ao extrair dados de configuração do cache local, seu aplicativo exige menos atualizações de código para gerenciar dados de configuração, recupera dados de configuração em milissegundos e não é afetado por problemas de rede que podem afetar as chamadas para esses dados.*
- O agente oferece uma experiência nativa para recuperar e resolver sinalizadores de AWS AppConfig recursos.
- Pronto para uso, o agente fornece as práticas recomendadas para estratégias de armazenamento em cache, intervalos de pesquisa e disponibilidade de dados de configuração local, enquanto rastreia os tokens de configuração necessários para chamadas de serviço subsequentes.
- Durante a execução em segundo plano, o agente consulta periodicamente o plano de AWS AppConfig dados para atualizações de dados de configuração. Seu aplicativo em contêiner pode recuperar os dados conectando-se ao localhost na porta 2772 (um valor de porta padrão personalizável) e chamando HTTP GET para recuperar os dados.
- AWS AppConfig O agente atualiza os dados de configuração em seus contêineres sem precisar reiniciá-los ou reciclá-los.

* O AWS AppConfig agente armazena os dados em cache na primeira vez que o serviço recupera seus dados de configuração. Por esse motivo, a primeira chamada para recuperar dados é mais lenta que as chamadas subsequentes.

Tópicos

Antes de começar

Para se integrar AWS AppConfig aos seus aplicativos de contêiner, você deve criar AWS AppConfig artefatos e dados de configuração, incluindo sinalizadores de recursos ou dados de configuração de formato livre. Para ter mais informações, consulte Criação de sinalizadores de recursos e dados de configuração de formato livre no AWS AppConfig.

Para recuperar dados de configuração hospedados por AWS AppConfig, seus aplicativos de contêiner devem ser configurados com acesso ao plano de AWS AppConfig dados. Para dar acesso aos seus aplicativos, atualize a política de permissões do IAM que é usada pelo perfil do IAM do seu serviço de contêiner. Especificamente, você deve adicionar as ações `appconfig:StartConfigurationSession` e `appconfig:GetLatestConfiguration` à política. Os perfis do IAM de serviço de contêineres incluem o seguinte:

- A função de tarefa do Amazon ECS
- A função do nó do Amazon EKS
- A função de execução do AWS Fargate (Fargate) pod (se seus contêineres do Amazon EKS usarem o Fargate para processamento computacional)

Para obter informações sobre como adicionar permissões a uma política, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

Como iniciar o AWS AppConfig Agent para integração com o Amazon ECS

O contêiner auxiliar do AWS AppConfig agente está disponível automaticamente em seu ambiente Amazon ECS. Para usar o contêiner auxiliar do AWS AppConfig Agent, você deve iniciá-lo.

Para iniciar o Amazon ECS (console)

1. Abra o console em https://console.aws.amazon.com/ecs/v2.
2. No painel de navegação, escolha Task definitions (Definições de tarefa).
3. Escolha a definição de tarefa para seu aplicativo e, em seguida, selecione a revisão mais recente.
4. Escolha Criar revisão, Criar revisão.
5. Escolha Adicionar mais contêineres.
6. Em Nome, insira um nome exclusivo para o contêiner do AWS AppConfig agente.
7. Em URI da imagem, insira: **`public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x`**
8. Em Contêiner essencial, escolha Sim.
9. Na seção Mapeamentos de portas, escolha Adicionar mapeamento de portas.
10. Em Porta do contêiner, insira **`2772`**.

> **Note**
>
> AWS AppConfig O agente é executado na porta 2772, por padrão. Você pode especificar uma porta diferente.

11. Selecione Create (Criar). O Amazon ECS cria uma nova revisão do contêiner e exibe os detalhes.
12. No painel de navegação, escolha Clusters e, em seguida, escolha o cluster do aplicativo na lista.
13. Na guia Serviços, selecione o serviço para seu aplicativo.
14. Selecione Atualizar.
15. Em Configuração de implantação, em Revisão, escolha a revisão mais recente.
16. Selecione Atualizar. O Amazon ECS implanta a definição de tarefa mais recente.
17. Após a conclusão da implantação, você pode verificar se o AWS AppConfig Agente está sendo executado na guia Configuração e tarefas. Na guia Tarefas, escolha a tarefa em execução.
18. Na seção Contêineres, verifique se o contêiner do AWS AppConfig agente está listado.
19. Para verificar se o AWS AppConfig agente foi iniciado, escolha a guia Registros. Localize uma declaração como a seguinte para o contêiner do AWS AppConfig Agente: `[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Você pode ajustar o comportamento padrão do AWS AppConfig Agente inserindo ou alterando variáveis de ambiente. Para mais informações sobre as variáveis de ambiente disponíveis, consulte Usando variáveis de ambiente para configurar o AWS AppConfig Agente para Amazon ECS e Amazon EKS. Para obter informações sobre como alterar variáveis de ambiente no Amazon ECS, consulte Passar variáveis de ambiente para um contêiner no Guia do desenvolvedor do Amazon Elastic Container Service.

Como iniciar a integração do AWS AppConfig Agent com o Amazon EKS

O contêiner auxiliar do AWS AppConfig Agent está disponível automaticamente em seu ambiente Amazon EKS. Para usar o contêiner auxiliar do AWS AppConfig Agent, você deve iniciá-lo. O procedimento a seguir descreve como usar a ferramenta de linha de comando `kubectl` do Amazon EKS para fazer alterações no arquivo `kubeconfig` do aplicativo de contêiner. Para obter mais informações sobre como criar ou editar um arquivo `kubeconfig`, consulte Criação ou atualização de um arquivo kubeconfig para um cluster do Amazon EKS no Guia do usuário do Amazon EKS.

Para iniciar o AWS AppConfig Agent (ferramenta de linha de comando kubectl)

1. Abra seu arquivo `kubeconfig` e verifique se o aplicativo Amazon EKS está sendo executado como uma implantação de contêiner único. O conteúdo do arquivo deve ser semelhante ao seguinte:

```
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. Adicione os detalhes da definição do contêiner do AWS AppConfig Agente ao seu arquivo de implantação YAML.

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

> **Note**
>
> Observe as seguintes informações:
>
> - AWS AppConfig O agente é executado na porta 2772, por padrão. Você pode especificar uma porta diferente.
> - Você pode ajustar o comportamento padrão do AWS AppConfig Agente inserindo variáveis de ambiente. Para ter mais informações, consulte Usando variáveis de ambiente para configurar o AWS AppConfig Agente para Amazon ECS e Amazon EKS.
> - Para *SERVICE_REGION*, especifique o Região da AWS código (por exemplo,`us-west-1`) em que o AWS AppConfig Agente recupera os dados de configuração.

3. Execute o comando a seguir na ferramenta `kubectl` para aplicar as alterações ao seu cluster.

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

4. Após a conclusão da implantação, verifique se o AWS AppConfig Agente está em execução. Use o comando a seguir para visualizar o arquivo de log do pod do aplicativo.

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

Localize uma declaração como a seguinte para o contêiner do AWS AppConfig Agente:
`[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> Você pode ajustar o comportamento padrão do AWS AppConfig Agente inserindo ou alterando variáveis de ambiente. Para mais informações sobre as variáveis de ambiente disponíveis, consulte Usando variáveis de ambiente para configurar o AWS AppConfig Agente para Amazon ECS e Amazon EKS.

Usando variáveis de ambiente para configurar o AWS AppConfig Agente para Amazon ECS e Amazon EKS

Você pode configurar o AWS AppConfig Agente alterando as seguintes variáveis de ambiente para seu contêiner de agente.

Variável de ambiente	Detalhes	Valor padrão
`ACCESS_TOKEN`	Esta variável de ambiente define um token que deve ser fornecido ao solicitar dados de configuração do servidor HTTP do agente. O valor do token deve ser definido no cabeçalho de autorização da solicitação HTTP com um tipo de autorização `Bearer`. Aqui está um exemplo. `GET /applications/my_app/... Host: localhost:2772`	Nenhum

Variável de ambiente	Detalhes	Valor padrão
	`Authorization: Bearer <token value>`	
`BACKUP_DIRECTORY`	Essa variável de ambiente permite que o AWS AppConfig Agente salve um backup de cada configuração recuperada no diretório especificado. **Important** As configurações copiadas em disco não são criptografadas. Se sua configuração contiver dados confidenciais, recomendamos AWS AppConfig que você pratique o princípio do menor privilégio com as permissões do sistema de arquivos. Para ter mais informações, consulte Segurança no AWS AppConfig.	Nenhum
`HTTP_PORT`	Esta variável de ambiente especifica a porta na qual o servidor HTTP do agente é executado.	2772

Variável de ambiente	Detalhes	Valor padrão
`LOG_LEVEL`	Esta variável de ambiente especifica o nível de detalhes que o agente registra. Cada nível inclui o nível atual e todos os níveis superiores. As variáveis diferenciam maiúsculas de minúsculas. Do mais detalhado ao menos detalhado, os níveis de log são: `debug`, `info`, `warn`, `error` e `none`. O `Debug` inclui informações detalhadas, incluindo informações de tempo, sobre o agente.	`info`

Variável de ambiente	Detalhes	Valor padrão
`MANIFEST`	Essa variável de ambiente configura o AWS AppConfig Agente para aproveitar os recursos adicionais por configuração, como recuperaç ões de várias contas e salvamento da configuração em disco. Você pode inserir um dos seguintes valores: • `"app:env:manifest-config"` • `"file:/fully/qualified/path/to/manifest.json"` Para obter mais informaçõ es sobre esses recursos, consulte Recursos adicionais de recuperação.	verdadeiro
`MAX_CONNECTIONS`	Esta variável de ambiente configura o número máximo de conexões que o agente usa para recuperar configurações do AWS AppConfig.	3

Variável de ambiente	Detalhes	Valor padrão
`POLL_INTERVAL`	Essa variável de ambiente controla a frequência com que o agente pesquisa dados AWS AppConfig de configuração atualizados. É possível especificar um número de segundos para o intervalo. Você também pode especificar um número com uma unidade de tempo: s para segundos, m para minutos, e h para horas. Se nenhuma unidade for especificada, o agente usará segundos como padrão. Por exemplo, 60, 60 s e 1 min resultam no mesmo intervalo de pesquisa.	45 segundos
`PREFETCH_LIST`	Essa variável de ambiente especifica os dados de configuração que o agente solicita AWS AppConfig assim que é iniciado.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
PRELOAD_BACKUPS	Se definido como`true`, o AWS AppConfig Agente carrega os backups de configuração encontrados BACKUP_DIRECTORY na memória e verifica imediatam ente se existe uma versão mais recente do serviço. Se definido como`false`, o AWS AppConfig Agente só carrega o conteúdo de um backup de configuração se não conseguir recuperar dados de configura ção do serviço, por exemplo, se houver um problema com sua rede.	verdadeiro
PROXY_HEADERS	Esta variável de ambiente especifica cabeçalhos que são exigidos pelo proxy referenci ado na variável de ambiente PROXY_URL . O valor é uma lista de cabeçalhos separados por vírgula. Cada cabeçalho usa o formato a seguir: `"header: value"`	Nenhum

Variável de ambiente	Detalhes	Valor padrão
PROXY_URL	Essa variável de ambiente especifica a URL do proxy a ser usada para conexões do agente com Serviços da AWS, inclusive AWS AppConfig . HTTPSe HTTP URLs são compatíveis.	Nenhum

Variável de ambiente	Detalhes	Valor padrão
`REQUEST_TIMEOUT`	Essa variável de ambiente controla a quantidade de tempo do qual o agente espera por uma resposta. AWS AppConfig Se o serviço não responder, a solicitação falhará. Se a solicitação for para a recuperação inicial de dados, o agente retornará um erro ao seu aplicativo. Se o tempo limite ocorrer durante uma verificação de dados atualizados em segundo plano, o agente registrará o erro e tentará novamente após um pequeno atraso. Você pode especificar o número de milissegundos para o tempo limite. Você pode também especificar um número com uma unidade de tempo: ms, para milissegu ndos, e s, para segundos. Se nenhuma unidade for especificada, o agente usará milissegundos como padrão. Por exemplo, 5000, 5000 ms e 5 s resultam no mesmo valor de tempo limite da solicitação.	3000 milissegundos

Variável de ambiente	Detalhes	Valor padrão
ROLE_ARN	Essa variável de ambiente especifica o Amazon Resource Name (ARN) de uma função do IAM. AWS AppConfig O agente assume essa função para recuperar os dados de configuração.	Nenhum
ROLE_EXTERNAL_ID	Esta variável de ambiente especifica o ID externo a ser usado com o ARN da função assumida.	Nenhum
ROLE_SESSION_NAME	Esta variável de ambiente especifica o nome da sessão a ser associado às credenciais do perfil do IAM assumido.	Nenhum
SERVICE_REGION	Essa variável de ambiente especifica uma alternativa Região da AWS que o AWS AppConfig Agente usa para chamar o AWS AppConfig serviço. Se não for definida, o agente tentará determina r a região atual. Se não for possível, o agente não iniciará.	Nenhum
WAIT_ON_MANIFEST	Essa variável de ambiente configura o AWS AppConfig Agente para esperar até que o manifesto seja processad o antes de concluir a inicializ ação.	verdadeiro

Recuperação de dados de configuração

Você pode recuperar dados de configuração do AWS AppConfig Agente usando uma chamada HTTP localhost. Os exemplos a seguir usam `curl` com um cliente HTTP. Você pode chamar o agente usando qualquer cliente HTTP disponível compatível com a linguagem do aplicativo ou com as bibliotecas disponíveis.

> **Note**
>
> Para recuperar dados de configuração se seu aplicativo usar uma barra invertida, por exemplo, “test-backend/test-service”, você precisará usar a codificação de URL.

Para recuperar o conteúdo completo de qualquer configuração implantada

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

Para recuperar um único sinalizador e seus atributos de uma configuração do AWS AppConfig do tipo **`Feature Flag`**

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

Para acessar vários sinalizadores e seus atributos a partir de uma configuração do AWS AppConfig do tipo **`Feature Flag`**

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

Recursos adicionais de recuperação

AWS AppConfig O agente oferece os seguintes recursos adicionais para ajudá-lo a recuperar configurações para seus aplicativos.

- Recuperação de várias contas: use o AWS AppConfig Agente de uma conta primária ou de recuperação Conta da AWS para recuperar dados de configuração de várias contas de fornecedores.
- Gravar cópia da configuração em disco: use o AWS AppConfig Agente para gravar dados de configuração em disco. Esse recurso permite que os clientes com aplicativos que leem dados de configuração do disco se integrem AWS AppConfig.

Sobre manifestos de agentes

Para habilitar esses recursos AWS AppConfig do Agente, você cria um manifesto. Um manifesto é um conjunto de dados de configuração que você fornece para controlar as ações que o agente pode realizar. Um manifesto é escrito em JSON. Ele contém um conjunto de chaves de nível superior que correspondem às diferentes configurações que você implantou usando. AWS AppConfig

Um manifesto pode incluir várias configurações. Além disso, cada configuração no manifesto pode identificar um ou mais recursos do agente a serem usados na configuração especificada. O conteúdo do manifesto usa o seguinte formato:

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

Aqui está um exemplo de JSON para um manifesto com duas configurações. A primeira configuração (*MyApp*) não usa nenhum recurso do AWS AppConfig Agente. A segunda configuração (*my2ndApp*) usa a cópia da configuração de gravação em disco e os recursos de recuperação de várias contas:

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

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

```
                "roleSessionName": "AwsAppConfigAgent",
                "credentialsDuration": "2h"
            },
            "writeTo": {
                "path": "/tmp/aws-appconfig/my-2nd-app/beta/my-enable-payments-feature-
flag-configuration.json"
            }
        }
    }
```

Como fornecer um manifesto de agente

Você pode armazenar o manifesto como um arquivo em um local onde o AWS AppConfig Agente possa lê-lo. Ou você pode armazenar o manifesto como uma AWS AppConfig configuração e direcionar o agente para ele. Para fornecer um manifesto do agente, você deve definir uma variável de `MANIFEST` ambiente com um dos seguintes valores:

Localização do manifesto	Valor da variável de ambiente	Caso de uso
Arquivo	arquivo: /path/to/agent-manifest.json	Use esse método se seu manifesto não mudar com frequência.
AWS AppConfig configuração	*nome do aplicativo: nome do ambiente: nome da configuração*	Use esse método para atualizações dinâmicas. Você pode atualizar e implantar um manifesto armazenado o AWS AppConfig como configuração da mesma forma que armazena outras AWS AppConfig configurações.
Variável de ambiente	Conteúdo do manifesto (JSON)	Use esse método se seu manifesto não mudar com frequência. Esse método é útil em ambientes de contêiner em que é mais fácil definir uma variável de ambiente do que expor um arquivo.

Para obter mais informações sobre a configuração de variáveis para o AWS AppConfig Agent, consulte o tópico relevante para seu caso de uso:

- Configuração da extensão do Lambda do AWS AppConfig Agent
- Usando o AWS AppConfig agente com o Amazon EC2
- Usando o AWS AppConfig Agent com o Amazon ECS e o Amazon EKS

Recuperação de várias contas

Você pode configurar o AWS AppConfig Agente para recuperar configurações de várias Contas da AWS inserindo substituições de credenciais no manifesto do Agente. AWS AppConfig As substituições de credenciais incluem o Amazon Resource Name (ARN) de uma função AWS Identity and Access Management (IAM), um ID da função, um nome de sessão e a duração de quanto tempo o agente pode assumir a função.

Você insere esses detalhes em uma seção de “credenciais” no manifesto. A seção “credenciais” usa o seguinte formato:

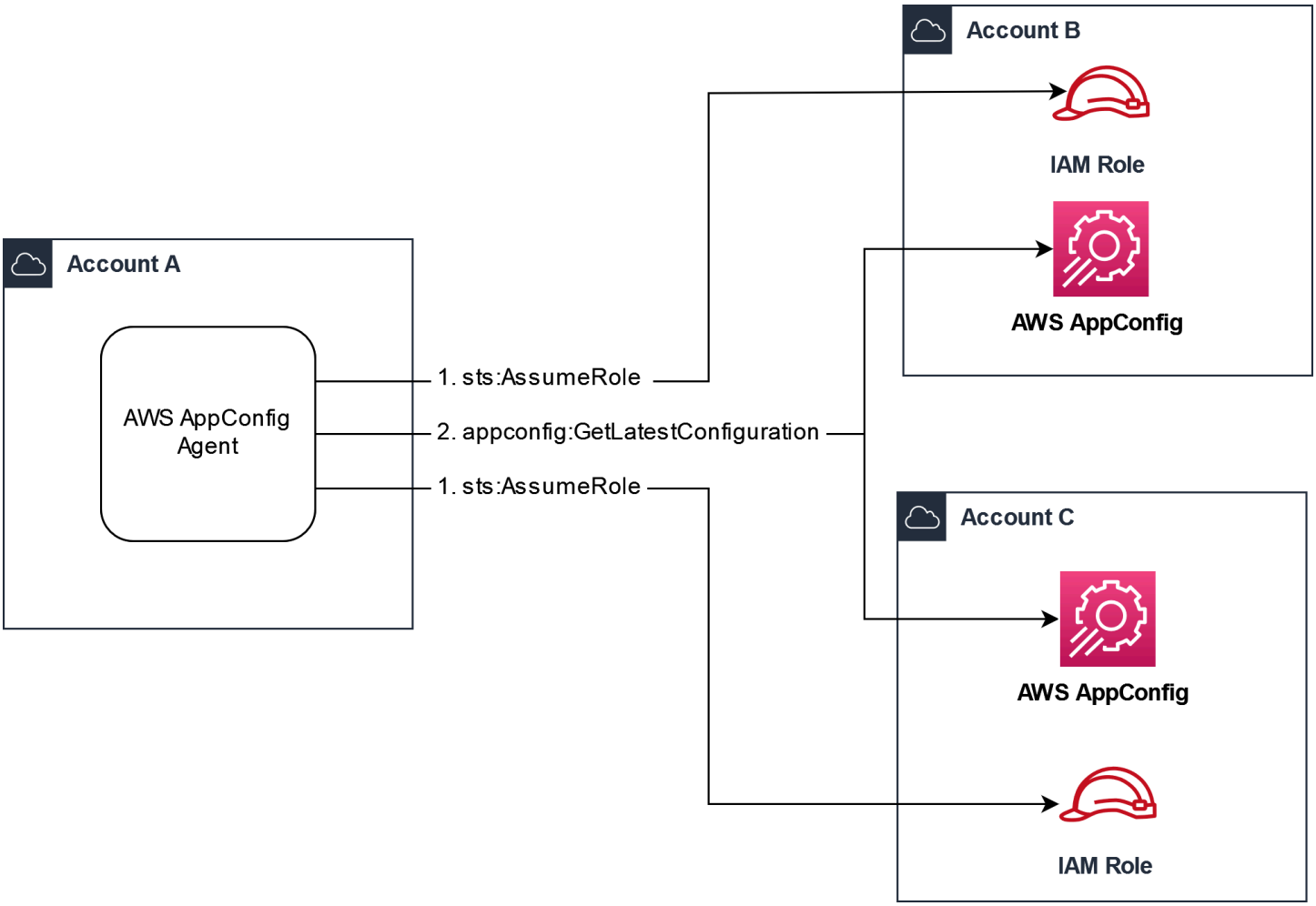
```
{
    "application_name:environment_name:configuration_name": {
        "credentials": {
            "roleArn": "arn:partition:iam::account_ID:role/roleName",
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
     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

Exemplo:

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

```
}
```

Antes de recuperar uma configuração, o agente lê os detalhes da credencial da configuração no manifesto e, em seguida, assume a função do IAM especificada para essa configuração. Você pode especificar um conjunto diferente de substituições de credenciais para configurações diferentes em um único manifesto. O diagrama a seguir mostra como o AWS AppConfig Agente, ao ser executado na Conta A (a conta de recuperação), assume funções separadas especificadas para as Contas B e C (as contas do fornecedor) e, em seguida, chama a operação da API de GetLatestConfiguração para recuperar os dados de configuração da AWS AppConfig execução nessas contas:



Configure permissões para recuperar dados de configuração das contas do fornecedor

AWS AppConfig O agente em execução na conta de recuperação precisa de permissão para recuperar dados de configuração das contas do fornecedor. Você concede permissão ao agente criando uma função AWS Identity and Access Management (IAM) em cada uma das contas do fornecedor. AWS AppConfig O agente na conta de recuperação assume essa função para obter

dados das contas do fornecedor. Conclua os procedimentos nesta seção para criar uma política de permissões do IAM, uma função do IAM e adicionar substituições de agente ao manifesto.

Antes de começar

Colete as informações a seguir antes de criar uma política de permissão e uma função no IAM.

- Os IDs de cada um Conta da AWS. A conta de recuperação é a conta que chamará outras contas para obter dados de configuração. As contas do fornecedor são as contas que fornecerão os dados de configuração para a conta de recuperação.
- O nome da função do IAM usada AWS AppConfig na conta de recuperação. Aqui está uma lista das funções usadas por AWS AppConfig, por padrão:
  - Para o Amazon Elastic Compute Cloud (Amazon EC2) AWS AppConfig , usa a função de instância.
  - Para AWS Lambda, AWS AppConfig usa a função de execução do Lambda.
  - Para o Amazon Elastic Container Service (Amazon ECS) e o Amazon Elastic Kubernetes Service (Amazon EKS), usa a função de contêiner. AWS AppConfig

  Se você configurou o AWS AppConfig Agente para usar uma função diferente do IAM especificando a variável de `ROLE_ARN` ambiente, anote esse nome.

Crie a política de permissões

Use o procedimento a seguir para criar uma política de permissões usando o console do IAM. Conclua o procedimento em cada um deles Conta da AWS que fornecerá dados de configuração para a conta de recuperação.

Para criar uma política do IAM

1. Faça login AWS Management Console na conta de um fornecedor.
2. Abra o console do IAM em https://console.aws.amazon.com/iam/.
3. No painel de navegação, escolha Políticas e, em seguida, Criar política.
4. Escolha a opção JSON.
5. No editor de políticas, substitua o JSON padrão pela seguinte declaração de política. Atualize cada *`exemplo de espaço reservado de recurso`* com detalhes da conta do fornecedor.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "appconfig:StartConfigurationSession",
            "appconfig:GetLatestConfiguration"
        ],
        "Resource":
 "arn:partition:appconfig:region:vendor_account_ID:application/
vendor_application_ID/environment/vendor_environment_ID/
configuration/vendor_configuration_ID"
    }
   ]
}
```

Veja um exemplo abaixo:

```
{
    "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. Selecione Next (Próximo).
7. No campo Nome da política, insira um nome.
8. (Opcional) Em Adicionar tags, adicione um ou mais pares de valores de chave de tag para organizar, rastrear ou controlar o acesso a essa política.
9. Escolha Criar política. O sistema retorna para a página Policies (Políticas).
10. Repita esse procedimento em cada uma delas Conta da AWS que fornecerá dados de configuração para a conta de recuperação.

Crie o perfil do IAM

Use o procedimento a seguir para criar uma função do IAM usando o console do IAM. Conclua o procedimento em cada um deles Conta da AWS que fornecerá dados de configuração para a conta de recuperação.

Para criar um perfil do IAM

1. Faça login AWS Management Console na conta de um fornecedor.
2. Abra o console IAM em https://console.aws.amazon.com/iam/.
3. No painel de navegação, escolha Funções e, em seguida, escolha Criar política.
4. Em Tipo de Entidade Confiável, escolha Conta da AWS.
5. Na Conta da AWSseção, escolha Outro Conta da AWS.
6. No campo ID da conta, insira o ID da conta de recuperação.
7. (Opcional) Como prática recomendada de segurança para essa função assumida, escolha Exigir ID externa e insira uma string.
8. Selecione Next (Próximo).
9. Na página Adicionar permissões, use o campo Pesquisar para localizar a política que você criou no procedimento anterior. Marque a caixa de seleção ao lado do nome.
10. Selecione Next (Próximo).
11. Em Nome do perfil, insira um nome.
12. (Opcional) Em Description (Descrição), insira uma descrição.
13. Para Etapa 1: Selecionar entidades confiáveis, escolha Editar. Substitua a política de confiança padrão do JSON pela política a seguir. Atualize cada *exemplo de espaço reservado de recurso* com informações da sua conta de recuperação.

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

```
}
```

14. (Opcional) em Tags, adicione um ou mais pares de valores tag-chave para organizar, monitorar ou controlar acesso para essa função.
15. Selecione Create role (Criar função). O sistema faz com que você retorne para a página Roles.
16. Pesquise a função que você acabou de criar. Escolha-o. Na seção ARN, copie o ARN. Você especificará essas informações no próximo procedimento.

Adicione substituições de credenciais ao manifesto

Depois de criar a função do IAM em sua conta de fornecedor, atualize o manifesto na conta de recuperação. Especificamente, adicione o bloco de credenciais e o ARN da função do IAM para recuperar dados de configuração da conta do fornecedor. Aqui está o formato 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"
        }
    }
}
```

Exemplo:

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

Valide se a recuperação de várias contas está funcionando

Você pode validar se esse agente é capaz de recuperar dados de configuração de várias contas revisando os registros do AWS AppConfig agente. O registro `INFO` de nível dos dados iniciais recuperados para 'YourApplicationName:YourEnvironmentName:YourConfigurationName' é o melhor indicador para recuperações bem-sucedidas. Se as recuperações estiverem falhando, você deverá ver um registro de `ERROR` nível indicando o motivo da falha. Aqui está um exemplo de uma recuperação bem-sucedida de uma conta de fornecedor:

```
[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
```

Gravar cópia da configuração em disco

Você pode configurar o AWS AppConfig Agente para armazenar automaticamente uma cópia de uma configuração em disco em texto simples. Esse recurso permite que os clientes com aplicativos que leem dados de configuração do disco se integrem AWS AppConfig.

Esse recurso não foi projetado para ser usado como um recurso de backup de configuração. AWS AppConfig O agente não lê os arquivos de configuração copiados para o disco. Se você quiser fazer backup das configurações em disco, consulte as variáveis de `PRELOAD_BACKUP` ambiente `BACKUP_DIRECTORY` e de uso do agente com o Amazon EC2 ou AWS AppConfig Uso do AWS AppConfig agente com o Amazon ECS e o Amazon EKS.

> **Warning**
>
> Observe as seguintes informações importantes sobre esse recurso:
>
> - As configurações salvas em disco são armazenadas em texto simples e podem ser lidas por humanos. Não habilite esse recurso para configurações que incluam dados confidenciais.
> - Esse recurso grava no disco local. Use o princípio do privilégio mínimo para permissões do sistema de arquivos. Para ter mais informações, consulte Implemente o acesso de privilégio mínimo.

Para habilitar a configuração de gravação, copie em disco

1. Edite o manifesto.

2. Escolha a configuração que você AWS AppConfig deseja gravar no disco e adicione um `writeTo` elemento. Exemplo:

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

Exemplo:

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

3. Salve as alterações. O arquivo configuration.json será atualizado sempre que novos dados de configuração forem implantados.

Valide se a cópia da configuração de gravação no disco está funcionando

Você pode validar se cópias de uma configuração estão sendo gravadas em disco examinando os registros do AWS AppConfig agente. A entrada de `INFO` registro com a frase “INFO escreveu a configuração '*aplicativo*: *ambiente*: *configuração*' em *file_path*" indica que o AWS AppConfig Agente grava cópias da configuração no disco.

Exemplo:

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MobileApp:Beta:EnableMobilePayments' in XX.Xms
[appconfig agent] 2023/11/13 17:05:49 INFO wrote configuration
 'MobileApp:Beta:EnableMobilePayments' to /tmp/configs/your-app/your-env/your-
config.json
```

AWS AppConfig Agente de desenvolvimento local

AWS AppConfig O agente suporta um modo de desenvolvimento local. Se você ativar o modo de desenvolvimento local, o agente lerá os dados de configuração de um diretório especificado no disco. Ele não recupera dados de configuração de AWS AppConfig. Você pode simular implantações de configuração atualizando arquivos no diretório especificado. Recomendamos o modo de desenvolvimento local para os seguintes casos de uso:

- Teste diferentes versões de configuração antes de implantá-las usando o. AWS AppConfig
- Teste diferentes opções de configuração para um novo recurso antes de confirmar as alterações no seu repositório de código.
- Teste diferentes cenários de configuração para verificar se eles funcionam conforme o esperado.

> **Warning**
>
> Não use o modo de desenvolvimento local em ambientes de produção. Esse modo não oferece suporte a recursos de AWS AppConfig segurança importantes, como validação de implantação e reversões automatizadas.

Use o procedimento a seguir para configurar o AWS AppConfig Agente para o modo de desenvolvimento local.

Para configurar o AWS AppConfig Agente para o modo de desenvolvimento local

1. Instale o agente usando o método descrito para seu ambiente computacional. AWS AppConfig O agente trabalha com o seguinte Serviços da AWS:
   - AWS Lambda
   - Amazon EC2
   - Amazon ECS e Amazon EKS
2. Se o agente estiver em execução, pare-o.
3. Adicione `LOCAL_DEVELOPMENT_DIRECTORY` à lista de variáveis de ambiente. Especifique um diretório no sistema de arquivos que forneça ao agente permissões de leitura. Por exemplo, `/tmp/local_configs`.
4. Crie um arquivo no diretório. O nome do arquivo deve usar o seguinte formato:

```
application_name:environment_name:configuration_profile_name
```

Exemplo:

```
Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
```

> **Note**
>
> (Opcional) Você pode controlar o tipo de conteúdo que o agente retorna para seus dados de configuração com base na extensão fornecida ao arquivo. Por exemplo, se você nomear o arquivo com uma extensão.json, o agente retornará um tipo de conteúdo de `application/json` quando seu aplicativo o solicitar. Se você omitir a extensão, o agente usará `application/octet-stream` para o tipo de conteúdo. Se precisar de um controle preciso, você pode fornecer uma extensão no formato `.type%subtype`. O agente retornará um tipo de conteúdo de `.type/subtype`.

5. Execute o comando a seguir para reiniciar o agente e solicitar os dados de configuração.

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

O agente verifica as alterações no arquivo local no intervalo de pesquisa especificado para o agente. Se o intervalo da pesquisa não for especificado, o agente usará o intervalo padrão de 45 segundos. Essa verificação no intervalo da pesquisa garante que o agente se comporte da mesma forma em um ambiente de desenvolvimento local e quando configurado para interagir com o serviço. AWS AppConfig

> **Note**
>
> Para implantar uma nova versão de um arquivo de configuração de desenvolvimento local, atualize o arquivo com novos dados.

Recuperação de configurações chamando diretamente as APIs

Seu aplicativo recupera os dados de configuração estabelecendo primeiro uma sessão de configuração usando a operação da API de StartConfigurationsessão. O cliente da sua sessão então faz chamadas periódicas para a GetLatestConfiguração para verificar e recuperar os dados mais recentes disponíveis.

Ao chamar `StartConfigurationSession`, seu código envia as seguintes informações:

- Identificadores (ID ou nome) de um AWS AppConfig aplicativo, ambiente e perfil de configuração que a sessão rastreia.
- (Opcional) O tempo mínimo que o cliente da sessão deve esperar entre chamadas para `GetLatestConfiguration`.

Em resposta, AWS AppConfig fornece um `InitialConfigurationToken` a ser entregue ao cliente da sessão e usado na primeira vez em que ele liga `GetLatestConfiguration` para essa sessão.

> **Important**
>
> Esse token só deve ser usado uma vez em sua primeira chamada para `GetLatestConfiguration`. Você deve usar o novo token na resposta `GetLatestConfiguration` (`NextPollConfigurationToken`) em cada chamada subseqüente para `GetLatestConfiguration`. Para oferecer suporte a casos de uso de pesquisas longas, os tokens são válidos por até 24 horas. Se uma chamada `GetLatestConfiguration` usar um token expirado, o sistema retornará `BadRequestException`.

Ao chamar `GetLatestConfiguration`, seu código de cliente envia o valor `ConfigurationToken` mais recente que ele tem e recebe em resposta:

- `NextPollConfigurationToken`: o valor `ConfigurationToken` a ser usado na próxima chamada para `GetLatestConfiguration`.
- `NextPollIntervalInSeconds`: o tempo que o cliente deve esperar até fazer sua próxima chamada para `GetLatestConfiguration`.

- A configuração: os dados mais recentes destinados à sessão. Este pode estar vazio se o cliente já tiver a versão mais recente da configuração.

> **Important**
>
> Observe as seguintes informações importantes:
>
> - A API de StartConfigurationsessão só deve ser chamada uma vez por aplicativo, ambiente, perfil de configuração e cliente para estabelecer uma sessão com o serviço. Isso geralmente é feito no startup do aplicativo ou imediatamente antes da primeira recuperação de uma configuração.
> - Se sua configuração for implantada usando um `KmsKeyIdentifier`, sua solicitação para receber a configuração deverá incluir permissão para chamar `kms:Decrypt`. Para obter mais informações, consulte Descriptografar na Referência da API do AWS Key Management Service .
> - A operação da API usada anteriormente para recuperar dados de configuração, `GetConfiguration`, está obsoleta. A operação `GetConfiguration` da API não é compatível com configurações criptografadas.

Exemplo de recuperação de uma configuração

O AWS CLI exemplo a seguir demonstra como recuperar dados de configuração usando as operações de AWS AppConfig dados `StartConfigurationSession` e `GetLatestConfiguration` API. O primeiro comando inicia uma sessão de configuração. Essa chamada inclui as IDs (ou nomes) do AWS AppConfig aplicativo, do ambiente e do perfil de configuração. A API retorna um `InitialConfigurationToken` usado para buscar seus dados de configuração.

```
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
```

O sistema responde com informações no seguinte formato.

```
{
```

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

Depois de iniciar uma sessão, use o InitialConfigurationToken para chamar a GetLatestConfiguração para buscar seus dados de configuração. Os dados de configuração são salvos no arquivo `mydata.json`.

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

A primeira chamada para `GetLatestConfiguration` usa o `ConfigurationToken` obtido de `StartConfigurationSession`. As informações a seguir são retornadas.

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

As chamadas subsequentes para `GetLatestConfiguration` devem fornecer `NextPollConfigurationToken` da resposta anterior.

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

Important

Observe os detalhes importantes a seguir sobre a operação `GetLatestConfiguration` da API:

- A resposta `GetLatestConfiguration` inclui uma seção `Configuration` que mostra os dados de configuração. A seção `Configuration` só aparece se o sistema encontrar dados de configuração novos ou atualizados. Se o sistema não encontrar dados de configuração novos ou atualizados, os dados de `Configuration` estarão vazios.
- Você recebe um novo `ConfigurationToken` a cada resposta de `GetLatestConfiguration`.

- Recomendamos ajustar a frequência de pesquisas das chamadas de API `GetLatestConfiguration` com base no seu orçamento, na frequência esperada de implantações de configuração e no número de destinos para uma configuração.

Estender fluxos de trabalho usando extensões

Uma extensão aumenta sua capacidade de injetar lógica ou comportamento em diferentes pontos durante o AWS AppConfig fluxo de trabalho de criação ou implantação de uma configuração. Por exemplo, você pode usar extensões para realizar os seguintes tipos de tarefas (para citar algumas):

- Enviar uma notificação para um tópico do Amazon Simple Notification Service (Amazon SNS) quando um perfil de configuração for implantado.
- Limpar o conteúdo de um perfil de configuração em busca de dados confidenciais antes do início da implantação.
- Criar ou atualizar uma ocorrência do Atlassian Jira sempre que uma alteração for feita em um sinalizador de recurso.
- Mesclar o conteúdo de um serviço ou fonte de dados com seus dados de configuração ao iniciar uma implantação.
- Fazer backup de uma configuração em um bucket do Amazon Simple Storage Service (Amazon S3) sempre que uma configuração for implantada.

Você pode associar esses tipos de tarefas a AWS AppConfig aplicativos, ambientes e perfis de configuração.

Conteúdo

Sobre AWS AppConfig extensões

Este tópico apresenta conceitos e terminologia de AWS AppConfig extensão. As informações são discutidas no contexto de cada etapa necessária para configurar e usar AWS AppConfig extensões.

Tópicos

- Etapa 3: crie uma associação de extensão
- Etapa 4: implante uma configuração e verifique se as ações da extensão são executadas

Etapa 1: determine o que você deseja fazer com as extensões

Você quer receber uma notificação para um webhook que envia mensagens para o Slack sempre que uma AWS AppConfig implantação for concluída? Deseja fazer backup de um perfil de configuração em um bucket do Amazon Simple Storage Service (Amazon S3) antes de uma configuração ser implantada? Deseja limpar os dados de configuração em busca de informações confidenciais antes da implantação da configuração? Você pode usar extensões para realizar esses tipos de tarefas e muito mais. Você pode criar extensões personalizadas ou usar as AWS extensões criadas incluídas com. AWS AppConfig

> **Note**
>
> Para a maioria dos casos de uso, para criar uma extensão personalizada, você deve criar uma AWS Lambda função para realizar qualquer computação e processamento definidos na extensão. Para ter mais informações, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

As extensões AWS criadas a seguir podem ajudá-lo a integrar rapidamente as implantações de configuração com outros serviços. Você pode usar essas extensões no AWS AppConfig console ou chamando ações da API de extensão diretamente do AWS CLI, AWS Tools for PowerShell, ou do SDK.

Extensão	Descrição
Amazon CloudWatch Evidently: testes A/B	Essa extensão permite que seu aplicativ o atribua variações às sessões do usuário localmente, em vez de chamar a EvaluateF eatureoperação. Para ter mais informações, consulte Trabalhando com a extensão Amazon CloudWatch Evidently.

Extensão	Descrição
AWS AppConfig eventos de implantação para EventBridge	Essa extensão envia eventos para o barramento de eventos EventBridge padrão quando uma configuração é implantada.
AWS AppConfig eventos de implantação no Amazon Simple Notification Service (Amazon SNS)	Esta extensão envia mensagens para um tópico do Amazon SNS que você especifica quando uma configuração é implantada.
AWS AppConfig eventos de implantação no Amazon Simple Queue Service (Amazon SQS)	Esta extensão enfileira mensagens em sua fila do Amazon SQS quando uma configuração é implantada.
Extensão de integração — Atlassian Jira	Essas extensões permitem AWS AppConfig criar e atualizar problemas sempre que você fizer alterações em um sinalizador de recurso.

Etapa 2: determine quando você deseja que a extensão seja executada

Uma extensão define uma ou mais ações que ela executa durante um AWS AppConfig fluxo de trabalho. Por exemplo, a `AWS AppConfig deployment events to Amazon SNS` extensão AWS criada inclui uma ação para enviar uma notificação para um tópico do Amazon SNS. Cada ação é invocada quando você interage com AWS AppConfig ou quando AWS AppConfig está executando um processo em seu nome. Eles são chamados de pontos de ação. AWS AppConfig as extensões suportam os seguintes pontos de ação:

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

As ações de extensão configuradas nos pontos de `PRE_*` ação são aplicadas após a validação da solicitação, mas antes de AWS AppConfig realizar a atividade que corresponde ao nome do ponto de ação. Essas invocações de ação são processadas ao mesmo tempo da solicitação. Se mais de uma solicitação for feita, as invocações de ação serão executadas sequencialmente. Observe também que os pontos de ação `PRE_*` recebem e podem alterar o conteúdo de uma configuração. Os pontos de ação `PRE_*` também podem responder a um erro e impedir que uma ação aconteça.

Uma extensão também pode ser executada paralelamente a um AWS AppConfig fluxo de trabalho usando um ponto de `ON_*` ação. `ON_*`os pontos de ação são invocados de forma assíncrona. `ON_*`os pontos de ação não recebem o conteúdo de uma configuração. Se uma extensão apresentar um erro durante um ponto de ação `ON_*`, o serviço ignorará o erro e continuará o fluxo de trabalho.

Etapa 3: crie uma associação de extensão

Para criar uma extensão ou configurar uma extensão de AWS autoria, você define os pontos de ação que invocam uma extensão quando um AWS AppConfig recurso específico é usado. Por exemplo, você pode optar por executar a extensão `AWS AppConfig deployment events to Amazon SNS` e receber notificações sobre um tópico do Amazon SNS sempre que uma implantação de configuração for iniciada para um aplicativo específico. Definir quais pontos de ação invocam uma extensão para um AWS AppConfig recurso específico é chamado de associação de extensão. Uma associação de extensão é uma relação especificada entre uma extensão e um AWS AppConfig recurso, como um aplicativo ou um perfil de configuração.

Um único AWS AppConfig aplicativo pode incluir vários ambientes e perfis de configuração. Se você associar uma extensão a um aplicativo ou ambiente, AWS AppConfig invoca a extensão para qualquer fluxo de trabalho relacionado ao aplicativo ou aos recursos do ambiente, se aplicável.

Por exemplo, digamos que você tenha um AWS AppConfig aplicativo chamado MobileApps que inclui um perfil de configuração chamado AccessList. E digamos que o MobileApps aplicativo inclua ambientes beta, de integração e de produção. Você cria uma associação de extensão para a extensão AWS de notificação criada pelo Amazon SNS e associa a extensão ao aplicativo. MobileApps A extensão de notificação do Amazon SNS é invocada sempre que a configuração é implantada para o aplicativo em qualquer um dos três ambientes.

> **Note**
>
> Você não precisa criar uma extensão para usar extensões de AWS autoria, mas precisa criar uma associação de extensão.

Etapa 4: implante uma configuração e verifique se as ações da extensão são executadas

Depois de criar uma associação, quando uma configuração hospedada é criada ou uma configuração é implantada, AWS AppConfig invoca a extensão e executa as ações especificadas. Quando uma extensão é invocada, se o sistema apresentar um erro durante um ponto de `PRE-*` ação, AWS AppConfig retornará informações sobre esse erro.

Trabalhar com extensões criadas pela AWS

AWS AppConfig inclui as seguintes extensões de AWS autoria. Essas extensões podem ajudar você a integrar o AWS AppConfig fluxo de trabalho com outros serviços. Você pode usar essas extensões no AWS Management Console ou chamando ações da API de extensão diretamente do AWS CLI, AWS Tools for PowerShell, ou do SDK.

Extensão	Descrição
Amazon CloudWatch Evidently: testes A/B	Essa extensão permite que seu aplicativ o atribua variações às sessões do usuário localmente, em vez de chamar a EvaluateF eatureoperação. Para ter mais informações, consulte Trabalhando com a extensão Amazon CloudWatch Evidently.
AWS AppConfig eventos de implantação para EventBridge	Essa extensão envia eventos para o barrament o de eventos EventBridge padrão quando uma configuração é implantada.
AWS AppConfig eventos de implantação no Amazon Simple Notification Service (Amazon SNS)	Esta extensão envia mensagens para um tópico do Amazon SNS que você especifica quando uma configuração é implantada.
AWS AppConfig eventos de implantação no Amazon Simple Queue Service (Amazon SQS)	Esta extensão enfileira mensagens em sua fila do Amazon SQS quando uma configuração é implantada.

Extensão	Descrição
Extensão de integração — Atlassian Jira	Essas extensões permitem AWS AppConfig criar e atualizar problemas sempre que você fizer alterações em um sinalizador de recurso.

Trabalhando com a extensão Amazon CloudWatch Evidently

Você pode usar o Amazon CloudWatch Evidently para validar com segurança novos recursos, disponibilizando-os a uma porcentagem específica de seus usuários enquanto você implanta o recurso. Você pode monitorar a performance do novo recurso para auxiliar na decisão de quando aumentar o tráfego para seus usuários. Isso ajuda você a reduzir riscos e identificar consequências não intencionais antes de iniciar totalmente o recurso. Você também pode realizar experimentos A/B para decidir sobre design de recursos com base em evidências e dados.

A AWS AppConfig extensão do CloudWatch Evidently permite que seu aplicativo atribua variações às sessões do usuário localmente, em vez de chamar a EvaluateFeatureoperação. Uma sessão local reduz os riscos de latência e disponibilidade associados a uma chamada de API. Para obter informações sobre como configurar e usar a extensão, consulte Executar lançamentos e experimentos A/B com o CloudWatch Evidently no Guia CloudWatch do usuário da Amazon.

Como trabalhar com a extensão **`AWS AppConfig deployment events to Amazon EventBridge`**

A `AWS AppConfig deployment events to Amazon EventBridge` extensão é uma extensão AWS criada por você que ajuda você a monitorar e agir no fluxo de trabalho de implantação da AWS AppConfig configuração. A extensão envia notificações de eventos para o barramento de eventos EventBridge padrão sempre que uma configuração é implantada. Depois de associar a extensão a um de seus AWS AppConfig aplicativos, ambientes ou perfis de configuração, AWS AppConfig envia notificações de eventos para o barramento de eventos após o início, término e reversão de cada implantação da configuração.

Se você quiser ter mais controle sobre quais pontos de ação enviam EventBridge notificações, você pode criar uma extensão personalizada e inserir o Amazon Resource Name (ARN) do barramento de eventos EventBridge padrão para o campo URI. Para obter mais informações sobre como criar uma extensão, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

> **Important**
>
> Essa extensão suporta somente o barramento de eventos EventBridge padrão.

Como usar a extensão

Para usar a `AWS AppConfig deployment events to Amazon EventBridge` extensão, primeiro você anexa a extensão a um de seus AWS AppConfig recursos criando uma associação de extensão. Você cria a associação usando o AWS AppConfig console ou a ação CreateExtensionAssociationda API. Ao criar a associação, você especifica o ARN de um AWS AppConfig aplicativo, ambiente ou perfil de configuração. Se você associar a extensão a um aplicativo ou ambiente, uma notificação de evento será enviada para qualquer perfil de configuração contido no aplicativo ou ambiente especificado.

Depois de criar a associação, quando uma configuração para o AWS AppConfig recurso especificado é implantada, AWS AppConfig invoca a extensão e envia notificações de acordo com os pontos de ação especificados na extensão.

> **Note**
>
> Esta extensão é invocada pelos seguintes pontos de ação:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Não é possível personalizar os pontos de ação desta extensão. Para invocar pontos de ação diferentes, você pode criar sua própria extensão. Para ter mais informações, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

Use os procedimentos a seguir para criar uma associação de AWS AppConfig extensão usando o AWS Systems Manager console ou AWS CLI o.

Para criar uma associação de extensão (console)

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha AWS AppConfig.
3. Na guia Extensões, escolha Adicionar ao recurso.
4. Na seção Detalhes do recurso de extensão, em Tipo de recurso, escolha um tipo de AWS AppConfig recurso. Dependendo do recurso escolhido, AWS AppConfig solicita que você escolha outros recursos.
5. Escolha Criar associação ao recurso.

Aqui está um exemplo de evento enviado para EventBridge quando a extensão é invocada.

```
{
   "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"
   }
}
```

Como trabalhar com a extensão **AWS AppConfig deployment events to Amazon SNS**

A `AWS AppConfig deployment events to Amazon SNS` extensão é uma extensão AWS criada por você que ajuda você a monitorar e agir no fluxo de trabalho de implantação da AWS AppConfig configuração. A extensão publica mensagens em um tópico do Amazon SNS sempre que uma configuração é implantada. Depois de associar a extensão a um de seus AWS AppConfig aplicativos, ambientes ou perfis de configuração, AWS AppConfig publica uma mensagem no tópico após cada início, término e reversão da implantação da configuração.

Se quiser ter mais controle sobre quais pontos de ação enviam notificações do Amazon SNS, você poderá criar uma extensão personalizada e inserir o nome do recurso da Amazon (ARN) de um tópico do Amazon SNS no campo URI. Para obter mais informações sobre como criar uma extensão, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

Como usar a extensão

Esta seção descreve como usar a extensão `AWS AppConfig deployment events to Amazon SNS`.

Etapa 1: Configurar AWS AppConfig para publicar mensagens em um tópico

Adicione uma política de controle de acesso ao seu tópico do Amazon SNS concedendo ao AWS AppConfig (`appconfig.amazonaws.com`) permissões de publicação (`sns:Publish`). Para obter mais informações, consulte Casos de exemplo para controle de acesso do Amazon SNS.

Etapa 2: crie uma associação de extensão

Anexe a extensão a um de seus AWS AppConfig recursos criando uma associação de extensão. Você cria a associação usando o AWS AppConfig console ou a ação CreateExtensionAssociationda API. Ao criar a associação, você especifica o ARN de um AWS AppConfig aplicativo, ambiente ou perfil de configuração. Se você associar a extensão a um aplicativo ou ambiente, uma notificação

será enviada para qualquer perfil de configuração contido no aplicativo ou ambiente especificado. Ao criar a associação, você deve inserir um valor para o parâmetro `topicArn` que contém o ARN do tópico do Amazon SNS que você deseja usar.

Depois de criar a associação, quando uma configuração para o AWS AppConfig recurso especificado é implantada, AWS AppConfig invoca a extensão e envia notificações de acordo com os pontos de ação especificados na extensão.

> **Note**
>
> Esta extensão é invocada pelos seguintes pontos de ação:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Não é possível personalizar os pontos de ação desta extensão. Para invocar pontos de ação diferentes, você pode criar sua própria extensão. Para ter mais informações, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

Use os procedimentos a seguir para criar uma associação de AWS AppConfig extensão usando o AWS Systems Manager console ou AWS CLI o.

Para criar uma associação de extensão (console)

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha AWS AppConfig.
3. Na guia Extensões, escolha Adicionar ao recurso.
4. Na seção Detalhes do recurso de extensão, em Tipo de recurso, escolha um tipo de AWS AppConfig recurso. Dependendo do recurso escolhido, AWS AppConfig solicita que você escolha outros recursos.
5. Escolha Criar associação ao recurso.

Veja uma amostra da mensagem enviada para o tópico do Amazon SNS quando a extensão é invocada.

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

Como trabalhar com a extensão **AWS AppConfig deployment events to Amazon SQS**

A `AWS AppConfig deployment events to Amazon SQS` extensão é uma extensão AWS criada por você que ajuda você a monitorar e agir no fluxo de trabalho de implantação da AWS AppConfig configuração. A extensão enfileira as mensagens em sua fila do Amazon Simple Queue Service (Amazon SQS) sempre que uma configuração é implantada. Depois de associar a extensão a um de seus AWS AppConfig aplicativos, ambientes ou perfis de configuração, coloca uma mensagem na AWS AppConfig fila após cada início, término e reversão de implantação da configuração.

Se quiser ter mais controle sobre quais pontos de ação enviam notificações do Amazon SQS, você pode criar uma extensão personalizada e inserir um nome do recurso da Amazon (ARN) da fila do Amazon SQS no campo URI. Para obter mais informações sobre como criar uma extensão, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

Como usar a extensão

Esta seção descreve como usar a extensão `AWS AppConfig deployment events to Amazon SQS`.

Etapa 1: Configurar AWS AppConfig para enfileirar mensagens

Adicione uma política do Amazon SQS à sua fila do Amazon SQS concedendo ao AWS AppConfig (`appconfig.amazonaws.com`) permissões de envio de mensagens (`sqs:SendMessage`). Para obter mais informações, consulte Exemplos básicos de políticas do Amazon SQS.

Etapa 2: crie uma associação de extensão

Anexe a extensão a um de seus AWS AppConfig recursos criando uma associação de extensão. Você cria a associação usando o AWS AppConfig console ou a ação CreateExtensionAssociationda API. Ao criar a associação, você especifica o ARN de um AWS AppConfig aplicativo, ambiente ou perfil de configuração. Se você associar a extensão a um aplicativo ou ambiente, uma notificação será enviada para qualquer perfil de configuração contido no aplicativo ou ambiente especificado. Ao criar a associação, você deve inserir um parâmetro `Here` que contenha o ARN da fila do Amazon SQS que você deseja usar.

Depois de criar a associação, quando uma configuração para o AWS AppConfig recurso especificado é criada ou implantada, AWS AppConfig invoca a extensão e envia notificações de acordo com os pontos de ação especificados na extensão.

> **Note**
>
> Esta extensão é invocada pelos seguintes pontos de ação:
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> Não é possível personalizar os pontos de ação desta extensão. Para invocar pontos de ação diferentes, você pode criar sua própria extensão. Para ter mais informações, consulte Passo a passo: Criação de extensões personalizadas AWS AppConfig.

Use os procedimentos a seguir para criar uma associação de AWS AppConfig extensão usando o AWS Systems Manager console ou AWS CLI o.

Para criar uma associação de extensão (console)

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha AWS AppConfig.
3. Na guia Extensões, escolha Adicionar ao recurso.
4. Na seção Detalhes do recurso de extensão, em Tipo de recurso, escolha um tipo de AWS AppConfig recurso. Dependendo do recurso escolhido, AWS AppConfig solicita que você escolha outros recursos.
5. Escolha Criar associação ao recurso.

Veja um exemplo da mensagem enviada para a fila do Amazon SQS quando a extensão é invocada.

```
{
   "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"
}
```

Trabalhando com a extensão Atlassian Jira para AWS AppConfig

Ao se integrar com o Atlassian Jira, você AWS AppConfig pode criar e atualizar problemas no console Atlassian sempre que você fizer alterações em uma bandeira de recurso em seu para o especificado. Conta da AWS Região da AWS Cada problema do Jira inclui o nome do sinalizador, o ID do aplicativo, o ID do perfil de configuração e os valores do sinalizador. Depois de atualizar, salvar e implantar as alterações do seu sinalizador, o Jira atualiza os problemas existentes com os detalhes da alteração.

> **Note**
>
> O Jira atualiza problemas sempre que você cria ou atualiza um sinalizador de atributos. O Jira também atualiza problemas quando você exclui um atributo de um sinalizador de nível filho de um sinalizador de nível pai. O Jira não registra informações quando você exclui um sinalizador de nível pai.

Para configurar a integração, você deve fazer o seguinte:

- Configurando permissões para integração com o AWS AppConfig Jira
- Configuração do aplicativo de integração do AWS AppConfig com o Jira

Configurando permissões para integração com o AWS AppConfig Jira

Ao configurar a AWS AppConfig integração com o Jira, você especifica as credenciais de um usuário. Especificamente, você insere o ID da chave de acesso e a chave secreta do usuário no aplicativo AWS AppConfig para Jira. Esse usuário dá permissão ao Jira para se comunicar com AWS AppConfig. AWS AppConfig usa essas credenciais uma vez para estabelecer uma associação entre AWS AppConfig e o Jira. As credenciais não são armazenadas. Você pode remover a associação desinstalando o aplicativo AWS AppConfig for Jira.

A conta do usuário exige uma política de permissão que inclua as seguintes ações:

- `appconfig:CreateExtensionAssociation`
- `appconfig:GetConfigurationProfile`
- `appconfig:ListApplications`
- `appconfig:ListConfigurationProfiles`
- `appconfig:ListExtensionAssociations`
- `sts:GetCallerIdentity`

Execute as seguintes tarefas para criar uma política de permissão do IAM e um usuário para a integração do AWS AppConfig com o Jira:

Tarefas

- Tarefa 1: criar uma política de permissão do IAM para a integração AWS AppConfig com o Jira
- Tarefa 2: Criar um usuário para a integração AWS AppConfig com o Jira

Tarefa 1: criar uma política de permissão do IAM para a integração AWS AppConfig com o Jira

Use o procedimento a seguir para criar uma política de permissão do IAM que permita que o Atlassian Jira se comunique com. AWS AppConfig Recomendamos criar uma política e associar essa política a um novo perfil do IAM. A adição da permissão necessária a uma política e um perfil do IAM existentes viola o princípio do privilégio mínimo e não é recomendada.

Para criar uma política do IAM AWS AppConfig e a integração com o Jira

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Políticas e, em seguida, Criar política.

3. Na página Criar política, selecione a guia JSON e substitua o conteúdo padrão pela política a seguir. Na política a seguir, substitua *Região*, *account_ID*, e *application_ID* e *configuration_profile_ID* por informações do seu ambiente de sinalizadores de atributos do AWS AppConfig .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "appconfig:CreateExtensionAssociation",
                      "appconfig:ListExtensionAssociations",
                      "appconfig:GetConfigurationProfile"
             ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID",

 "arn:aws:appconfig:Region:account_ID:application/application_ID/
configurationprofile/configuration_profile_ID"
             ]
        },
       {
            "Effect": "Allow",
            "Action": [
                      "appconfig:ListApplications"

             ],
            "Resource": [
                      "arn:aws:appconfig:Region:account_ID:*"
             ]
        },
         {
            "Effect": "Allow",
            "Action": [
                      "appconfig:ListConfigurationProfiles"
             ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID"
             ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
            "Resource": "*"
        }
    ]
}
```

4. Escolha Próximo: etiquetas.
5. (Opcional) Adicione um ou mais pares de chave-valor de tag para organizar, monitorar ou controlar o acesso para esta função e selecione Next: Review (Próximo: revisar).
6. Na página Review policy (Revisar política), insira um nome na caixa Name (Nome), como **`AppConfigJiraPolicy`**, e insira uma descrição opcional.
7. Escolha Criar política.

Tarefa 2: Criar um usuário para a integração AWS AppConfig com o Jira

Use o procedimento a seguir para criar um usuário para AWS AppConfig a integração com o Atlassian Jira. Depois de criar o usuário, você poderá copiar o ID da chave de acesso e a chave secreta, que serão especificados ao concluir a integração.

Para criar um usuário para a integração AWS AppConfig com o Jira

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Users (Usuários) e Add users (Adicionar usuários).
3. No campo Nome do usuário, insira um nome, como **`AppConfigJiraUser`**.
4. Em Selecionar tipo de AWS credencial, escolha Chave de acesso - Acesso programático.
5. Escolha Próximo: permissões.
6. Na página Definir permissões, selecione Anexar políticas existentes diretamente. Pesquise a política que você criou e marque a caixa de seleção correspondente em Tarefa 1: criar uma política de permissão do IAM para a integração AWS AppConfig com o Jira e escolha Próximo: Tags.
7. Na página Adicionar tags (opcional), adicione um ou mais pares de chave-valor de tags para organizar, monitorar ou controlar acesso para esse usuário. Escolha Próximo: revisar.
8. Na página Revisar, verifique os detalhes do usuário.

9. Escolha Criar usuário. O sistema exibe o ID da chave de acesso e a chave secreta do usuário. Baixe o arquivo .csv ou copie essas credenciais em um local separado. Você especificará essas credenciais ao configurar a integração.

Configuração do aplicativo de integração do AWS AppConfig com o Jira

Use o procedimento a seguir para configurar as opções necessárias no aplicativo AWS AppConfig for Jira. Depois de concluir esse procedimento, o Jira cria um novo problema para cada sinalizador de recurso no seu Conta da AWS para o especificado Região da AWS. Se você fizer alterações em um sinalizador de recurso AWS AppConfig, o Jira registrará os detalhes nas edições existentes.

> **Note**
>
> Um sinalizador de AWS AppConfig recurso pode incluir vários atributos de sinalizador de nível infantil. O Jira cria um problema para cada sinalizador de atributos de nível pai. Se alterar um atributo de sinalizador de nível filho, você poderá ver os detalhes dessa alteração no problema do Jira para o sinalizador de nível pai.

Para configurar a integração

1. Faça login no Atlassian Marketplace.
2. No campo de pesquisa, digite **`AWS AppConfig`** e pressione Enter.
3. Instale o aplicativo na sua instância do Jira.
4. No console Atlassian, escolha Gerenciar aplicativos e, em seguida, escolha AWS AppConfig para Jira.
5. Selecione Configurar.
6. Em Detalhes da configuração, escolha Projeto Jira e, em seguida, escolha o projeto que você deseja associar ao seu sinalizador de atributos do AWS AppConfig .
7. Escolha Região da AWS e, em seguida, escolha a região em que seu sinalizador de atributos do AWS AppConfig está localizado.
8. No campo ID do aplicativo, insira o nome do aplicativo do AWS AppConfig que contém seu sinalizador de atributos.
9. No campo ID do perfil de configuração, insira o nome do perfil de configuração do AWS AppConfig do seu sinalizador de atributos.

10. Nos campos ID da chave de acesso e Chave secreta, insira as credenciais que você copiou em Tarefa 2: Criar um usuário para a integração AWS AppConfig com o Jira. Como opção, especifique também um token de sessão.
11. Selecione Enviar.
12. No console da Atlassian, escolha Projetos e, em seguida, escolha o projeto que você selecionou para AWS AppConfig integração. A página Problemas exibe um problema para cada sinalizador de recurso no especificado Conta da AWS Região da AWS e.

Como excluir o aplicativo e dados do AWS AppConfig para Jira

Se você não quiser mais usar a integração do Jira com sinalizadores de AWS AppConfig recursos, você pode excluir o aplicativo AWS AppConfig for Jira no console da Atlassian. A exclusão do aplicativo de integração faz o seguinte:

- Exclui a associação entre sua instância do Jira e AWS AppConfig
- Exclui os detalhes da sua instância do Jira de AWS AppConfig

Para excluir o aplicativo AWS AppConfig for Jira

1. No console Atlassian, escolha Gerenciar aplicativos.
2. Escolha AWS AppConfig para Jira.
3. Clique em Desinstalar.

Passo a passo: Criação de extensões personalizadas AWS AppConfig

Para criar uma AWS AppConfig extensão personalizada, conclua as tarefas a seguir. Cada tarefa é descrita em mais detalhes nos tópicos mais adiante.

> **Note**
>
> Você pode ver exemplos de AWS AppConfig extensões personalizadas em GitHub:
>
> - Exemplo de extensão que impede implantações com um calendário de `blocked day` moratória usando o Systems Manager Change Calendar

- Extensão de amostra que evita que segredos vazem para os dados de configuração usando git-secrets
- Extensão de amostra que impede que informações de identificação pessoal (PII) vazem para os dados de configuração usando o Amazon Comprehend

1. Crie uma AWS Lambda função

   Para a maioria dos casos de uso, para criar uma extensão personalizada, você deve criar uma AWS Lambda função para realizar qualquer computação e processamento definidos na extensão. Uma exceção a essa regra é se você criar versões personalizadas das extensões de notificação criadas pela AWS para adicionar ou remover pontos de ação. Para obter mais detalhes sobre esta exceção, consulte Criação de uma AWS AppConfig extensão personalizada.

2. Configure permissões para sua extensão personalizada

   Para configurar permissões para sua extensão personalizada, você pode realizar um dos seguintes procedimentos:

   - Crie uma função de serviço AWS Identity and Access Management (IAM) que inclua `InvokeFunction` permissões.
   - Crie uma política de recursos usando a ação da AddPermissionAPI Lambda.

   Este passo a passo descreve como criar o perfil de serviço do IAM.

3. Crie uma extensão

   Você pode criar uma extensão usando o AWS AppConfig console ou chamando a ação da CreateExtensionAPI a partir do AWS CLI, AWS Tools for PowerShell, ou do SDK. O passo a passo usa o console.

4. Crie uma associação de extensão

   Você pode criar uma associação de extensão usando o AWS AppConfig console ou chamando a ação da CreateExtensionAssociationAPI do AWS CLI, AWS Tools for PowerShell, ou do SDK. O passo a passo usa o console.

5. Execute uma ação que invoque a extensão

   Depois de criar a associação, AWS AppConfig invoca a extensão quando os pontos de ação definidos pela extensão ocorrerem para esse recurso. Por exemplo, se você associar uma

extensão que contém uma ação `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`, a extensão será invocada toda vez que você criar uma nova versão de configuração hospedada.

Os tópicos nesta seção descrevem cada tarefa envolvida na criação de uma extensão do AWS AppConfig personalizada. Cada tarefa é descrita no contexto de um caso de uso em que o cliente deseja criar uma extensão que faz backup automático de uma configuração em um bucket do Amazon Simple Storage Service (Amazon S3). A extensão é executada sempre que uma configuração hospedada é criada (`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`) ou implantada (`PRE_START_DEPLOYMENT`).

Tópicos

Criação de uma função Lambda para uma extensão personalizada AWS AppConfig

Para a maioria dos casos de uso, para criar uma extensão personalizada, você deve criar uma AWS Lambda função para realizar qualquer computação e processamento definidos na extensão. Esta seção inclui um exemplo de código da função Lambda para uma extensão personalizada AWS AppConfig . Esta seção inclui também detalhes de referência de solicitações e respostas de carga útil. Para obter mais informações sobre como criar uma função do Lambda, consulte a seção Conceitos básicos do Lambda no Guia do desenvolvedor do AWS Lambda .

Código de exemplo

O código de exemplo a seguir para uma função Lambda, quando invocado, faz backup automático de uma AWS AppConfig configuração em um bucket do Amazon S3. O backup da configuração é feito sempre que uma nova configuração é criada ou implantada. O exemplo usa parâmetros de extensão para que o nome do bucket não precise ser codificado na função do Lambda. Usando parâmetros de extensão, o usuário pode anexar a extensão a vários aplicativos e fazer backup das configurações em diferentes buckets. O exemplo de código inclui comentários para explicar melhor a função.

Exemplo de função Lambda para uma extensão 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)
```

Se você desejar usar esse exemplo durante este passo a passo, salve-o com o nome **`MyS3ConfigurationBackUpExtension`** e copie o nome do recurso da Amazon (ARN) da função. Você especifica o ARN ao criar a função de assumir AWS Identity and Access Management (IAM) na próxima seção. Você especifica o ARN e o nome ao criar a extensão.

Referência de carga útil

Esta seção inclui detalhes de referência de solicitação e resposta de carga útil para trabalhar com AWS AppConfig extensões personalizadas.

Estrutura de solicitações

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'
}
```

Eventos assíncronos

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'
}
```

Estrutura de respostas

Os exemplos a seguir mostram o que sua função Lambda retorna em resposta à solicitação de uma extensão personalizada. AWS AppConfig

Eventos síncronos: resposta bem-sucedida

Se você deseja transformar o conteúdo, use o seguinte:

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

Se você não deseja transformar o conteúdo, não retorne nada.

Eventos assíncronos: resposta bem-sucedida

Não retorne nada.

Todos os eventos de erro

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

Configurando permissões para uma extensão personalizada AWS AppConfig

Use o procedimento a seguir para criar e configurar uma função de serviço AWS Identity and Access Management (IAM) (ou assumir uma função). AWS AppConfig usa essa função para invocar a função Lambda.

Para criar uma função de serviço do IAM e AWS AppConfig permitir assumi-la

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Roles e Create role.
3. Em Selecionar tipo de entidade confiável, escolha Política de confiança personalizada.
4. Cole a política JSON a seguir no campo Política de confiança personalizada.

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

```
    }
  ]
}
```

Escolha Próximo.

5. Na página Adicionar permissões, escolha Criar política. A página Create policy (Criar política) é aberta em uma nova guia.
6. Escolha a guia JSON e cole a política personalizada a seguir no editor. A ação `lambda:InvokeFunction` é usada para pontos de ação `PRE_*`. A ação `lambda:InvokeAsync` é usada para pontos de ação `ON_*`. Substitua *`o ARN do seu Lambda`* pelo nome do recurso da Amazon (ARN) do seu Lambda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "Your Lambda ARN"
    }
  ]
}
```

7. Escolha Próximo: etiquetas.
8. Na página Adicionar tags (opcional), adicione um ou mais pares chave-valor e escolha Próximo: revisão.
9. Na página Revisar política, insira um nome e uma descrição e, depois, escolha Criar política.
10. Na guia do navegador da sua política de confiança personalizada, escolha o ícone Atualizar e pesquise a política de permissões que você acabou de criar.
11. Marque a caixa de seleção da política de permissões e escolha Próximo.
12. Na página Nomear, revisar e criar, insira um nome na caixa Nome do perfil e, em seguida, insira uma descrição.
13. Selecione Create role (Criar função). O sistema faz com que você retorne para a página Roles. Escolha Exibir perfil no banner.

14. Copie o ARN. Você especifica esse ARN ao criar a extensão.

Criação de uma AWS AppConfig extensão personalizada

Uma extensão define uma ou mais ações que ela executa durante um AWS AppConfig fluxo de trabalho. Por exemplo, a `AWS AppConfig deployment events to Amazon SNS` extensão AWS criada inclui uma ação para enviar uma notificação para um tópico do Amazon SNS. Cada ação é invocada quando você interage com AWS AppConfig ou quando AWS AppConfig está executando um processo em seu nome. Eles são chamados de pontos de ação. AWS AppConfig as extensões suportam os seguintes pontos de ação:

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

As ações de extensão configuradas nos pontos de `PRE_*` ação são aplicadas após a validação da solicitação, mas antes de AWS AppConfig realizar a atividade que corresponde ao nome do ponto de ação. Essas invocações de ação são processadas ao mesmo tempo da solicitação. Se mais de uma solicitação for feita, as invocações de ação serão executadas sequencialmente. Observe também que os pontos de ação `PRE_*` recebem e podem alterar o conteúdo de uma configuração. Os pontos de ação `PRE_*` também podem responder a um erro e impedir que uma ação aconteça.

Uma extensão também pode ser executada paralelamente a um AWS AppConfig fluxo de trabalho usando um ponto de `ON_*` ação. `ON_*`os pontos de ação são invocados de forma assíncrona. `ON_*`os pontos de ação não recebem o conteúdo de uma configuração. Se uma extensão apresentar um erro durante um ponto de ação `ON_*`, o serviço ignorará o erro e continuará o fluxo de trabalho.

O exemplo de extensão a seguir define uma ação que chama o ponto de ação `PRE_CREATE_HOSTED_CONFIGURATION_VERSION`. No campo `Uri`, a ação especifica o nome do recurso da Amazon (ARN) da função do Lambda `MyS3ConfigurationBackUpExtension` criada

anteriormente neste passo a passo. A ação também especifica o ARN de função de assumir AWS Identity and Access Management (IAM) criado anteriormente nesta explicação passo a passo.

AWS AppConfig Extensão de amostra

```
{
    "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**
>
> Para ver a sintaxe da solicitação e as descrições dos campos ao criar uma extensão, consulte o CreateExtensiontópico na Referência da AWS AppConfig API.

Para criar uma extensão (console)

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha AWS AppConfig.
3. Na guia Extensões, escolha Criar extensão.
4. Em Nome da extensão, insira um nome exclusivo. Para a finalidade deste passo a passo, insira **`MyS3ConfigurationBackUpExtension`**. Como opção, insira uma descrição.
5. Na seção Ações, selecione Adicionar nova ação.

6. Em Nome da ação, insira um nome exclusivo. Para a finalidade deste passo a passo, insira **`PreCreateHostedConfigVersionActionForS3Backup`**. Esse nome descreve o ponto de ação usado pela ação e a finalidade da extensão.
7. Na lista Ponto de ação, escolha PRE_CREATE_HOSTED_CONFIGURATION_VERSION.
8. Para Uri, escolha Função do Lambda e, em seguida, escolha a função na lista Função do Lambda. Se você não vê sua função, verifique se você está no mesmo Região da AWS local em que criou a função.
9. Para Perfil do IAM, escolha o perfil que você criou anteriormente neste passo a passo.
10. Na seção Parâmetros de extensão (opcional), escolha Adicionar novo parâmetro.
11. Em Nome do parâmetro, insira um nome. Para a finalidade deste passo a passo, insira **`S3_BUCKET`**.
12. Repita as etapas de 5 a 11 para criar uma segunda ação para o ponto de ação `PRE_START_DEPLOYMENT`.
13. Escolha Criar extensão.

Personalização de extensões AWS de notificação criadas

Não é necessário criar um Lambda ou uma extensão para usar extensões de notificação criadas pela AWS. Basta criar uma associação de extensão e, em seguida, realizar uma operação que chame um dos pontos de ação suportados. Por padrão, as extensões AWS de notificação criadas oferecem suporte aos seguintes pontos de ação:

- `ON_DEPLOYMENT_START`
- `ON_DEPLOYMENT_COMPLETE`
- `ON_DEPLOYMENT_ROLLED_BACK`

Se criar versões personalizadas da extensão `AWS AppConfig deployment events to Amazon SNS` e das extensões `AWS AppConfig deployment events to Amazon SQS`, você poderá especificar os pontos de ação para os quais deseja receber notificações.

> **Note**
>
> A extensão `AWS AppConfig deployment events to EventBridge` não suporta os pontos de ação `PRE_*`. Você pode criar uma versão personalizada se quiser remover alguns dos pontos de ação padrão atribuídos à AWS versão criada.

Você não precisa criar nenhuma função do Lambda se criar versões personalizadas das extensões de notificação criadas pela AWS . Você só precisa especificar um nome do recurso da Amazon (ARN) no campo `Uri` para a nova versão da extensão.

- Para uma extensão de EventBridge notificação personalizada, insira o ARN dos eventos EventBridge padrão no `Uri` campo.
- Para uma extensão de notificação personalizada do Amazon SNS, insira o ARN de um tópico do Amazon SNS no campo `Uri`.
- Para uma extensão de notificação personalizada do Amazon SQS, insira o ARN de uma fila de mensagens do Amazon SQS no campo `Uri`.

Criando uma associação de extensão para uma AWS AppConfig extensão personalizada

Para criar uma extensão ou configurar uma extensão de AWS autoria, você define os pontos de ação que invocam uma extensão quando um AWS AppConfig recurso específico é usado. Por exemplo, você pode optar por executar a extensão `AWS AppConfig deployment events to Amazon SNS` e receber notificações sobre um tópico do Amazon SNS sempre que uma implantação de configuração for iniciada para um aplicativo específico. Definir quais pontos de ação invocam uma extensão para um AWS AppConfig recurso específico é chamado de associação de extensão. Uma associação de extensão é uma relação especificada entre uma extensão e um AWS AppConfig recurso, como um aplicativo ou um perfil de configuração.

Um único AWS AppConfig aplicativo pode incluir vários ambientes e perfis de configuração. Se você associar uma extensão a um aplicativo ou ambiente, AWS AppConfig invoca a extensão para qualquer fluxo de trabalho relacionado ao aplicativo ou aos recursos do ambiente, se aplicável.

Por exemplo, digamos que você tenha um AWS AppConfig aplicativo chamado MobileApps que inclui um perfil de configuração chamado AccessList. E digamos que o MobileApps aplicativo inclua ambientes beta, de integração e de produção. Você cria uma associação de extensão para

a extensão AWS de notificação criada pelo Amazon SNS e associa a extensão ao aplicativo. MobileApps A extensão de notificação do Amazon SNS é invocada sempre que a configuração é implantada para o aplicativo em qualquer um dos três ambientes.

Use os procedimentos a seguir para criar uma associação de AWS AppConfig extensão usando o AWS AppConfig console.

Para criar uma associação de extensão (console)

1. Abra o AWS Systems Manager console em https://console.aws.amazon.com/systems-manager/appconfig/.
2. No painel de navegação, escolha AWS AppConfig.
3. Na guia Extensões, escolha um botão de opção para uma extensão e escolha Adicionar ao recurso. Para fins deste passo a passo, escolha MyS3. ConfigurationBackUpExtension
4. Na seção Detalhes do recurso de extensão, em Tipo de recurso, escolha um tipo de AWS AppConfig recurso. Dependendo do recurso escolhido, AWS AppConfig solicita que você escolha outros recursos. Para a finalidade deste passo a passo, escolha Aplicativo.
5. Selecione um aplicativo na lista.
6. Na seção Parâmetros, verifique se S3_BUCKET está listado no campo Chave. No campo Valor, cole o ARN das extensões do Lambda. Por exemplo: `arn:aws:lambda:`*`aws-region`*`:111122223333:function:MyS3ConfigurationBackUpExtension`.
7. Escolha Criar associação ao recurso.

Executando uma ação que invoca uma extensão personalizada AWS AppConfig

Depois de criar a associação, você pode invocar a extensão `MyS3ConfigurationBackUpExtension` criando um novo perfil de configuração que especifique `hosted` para seu `SourceUri`. Como parte do fluxo de trabalho para criar a nova configuração, AWS AppConfig encontra o ponto de `PRE_CREATE_HOSTED_CONFIGURATION_VERSION` ação. O encontro desse ponto de ação invoca a extensão `MyS3ConfigurationBackUpExtension`, que faz backup automático da configuração recém-criada no bucket do S3 especificado na seção `Parameter` da associação da extensão.

AWS AppConfig integração de extensões com o Atlassian Jira

AWS AppConfig se integra ao Atlassian Jira. A integração permite AWS AppConfig criar e atualizar problemas no console Atlassian sempre que você fizer alterações em um sinalizador de recurso no seu Conta da AWS para o especificado. Região da AWS Cada problema do Jira inclui o nome do sinalizador, o ID do aplicativo, o ID do perfil de configuração e os valores do sinalizador. Depois de atualizar, salvar e implantar as alterações do seu sinalizador, o Jira atualiza os problemas existentes com os detalhes da alteração. Para ter mais informações, consulte Trabalhando com a extensão Atlassian Jira para AWS AppConfig.

Exemplos de código da AWS AppConfig

Esta seção inclui exemplos de código para executar ações comuns AWS AppConfig de forma programática. Recomendamos que você use esses exemplos com Java, Python e JavaScriptSDKs para realizar as ações em um ambiente de teste. Esta seção inclui uma amostra de código para limpar seu ambiente de teste depois que você terminar.

Tópicos

Criando ou atualizando uma configuração de formato livre armazenada no repositório de configurações hospedado

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código. Os exemplos nesta seção chamam as seguintes 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",
  })
);
```

Criando um perfil de configuração para um segredo armazenado no Secrets Manager

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código. Os exemplos nesta seção chamam as seguintes 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",
  })
);
```

Implantando um perfil de configuração

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código. Os exemplos nesta seção chamam as seguintes 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",
  })
);
```

Usando o AWS AppConfig Agente para ler um perfil de configuração de formato livre

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código.

Java

```
public void retrieveConfigFromAgent() throws Exception {
        /*
        In this sample, we will retrieve configuration data from the AWS AppConfig
 Agent.
        The agent is a sidecar process that handles retrieving configuration data
 from AppConfig
        for you in a way that implements best practices like configuration caching.

        For more information about the agent, see Simplified retrieval methods
        */

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

Python

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

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

import requests

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

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

JavaScript

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

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

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

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

Usando o AWS AppConfig Agente para ler um sinalizador de recurso específico

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código.

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 }
```

Usando a ação GetLatestConfig da API para ler um perfil de configuração de formato livre

Cada um dos exemplos a seguir inclui comentários sobre as ações executadas pelo código. Os exemplos nesta seção chamam as seguintes APIs:

- GetLatestConfiguration
- StartConfigurationSession

Java

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

        /*
```

```
        Start a new configuration session using the StartConfigurationSession API.
This operation does not return configuration data.
        Rather, it returns an initial configuration token that should be passed to
GetLatestConfiguration.
        IMPORTANT: This operation should only be performed once (per configuration),
prior to the first GetLatestConfiguration
        call you preform. Each GetLatestConfiguration will return a new
configuration token that you should then use in the
        next GetLatestConfiguration call.
        */
        StartConfigurationSessionResponse session =
            appConfigData.startConfigurationSession(req -> req
                .applicationIdentifier("MyDemoApp")
                .configurationProfileIdentifier("MyConfigProfile")
                .environmentIdentifier("Beta"));

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

        You can also avoid all the complexity around writing this code yourself by
leveraging our agent instead.
        For more information about the agent, see Simplified retrieval methods
        */

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

appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

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

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

 appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

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

Python

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

import boto3

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

appconfigdata = boto3.client('appconfigdata')

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

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

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

JavaScript

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

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

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

const appconfigdata = new AppConfigDataClient();

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

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

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

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

Limpando seu ambiente

Se você executou uma ou mais das amostras de código nesta seção, recomendamos usar uma das amostras a seguir para localizar e excluir os AWS AppConfig recursos criados por essas amostras de código. Os exemplos nesta seção chamam as seguintes 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})`)
  }
}
```

Segurança no AWS AppConfig

A segurança na nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de um datacenter e de uma arquitetura de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhadadescreve isso como a segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: AWS é responsável pela proteção da infraestrutura que executaAWS produtos da Nuvem AWS na AWS. A também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos AWSProgramas de conformidade. Para saber mais sobre os programas de conformidade que se aplicam ao AWS Systems Manager, consulte AWSServiços da em escopo por programa de conformidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

O AWS AppConfig é um recurso do AWS Systems Manager. Para entender como aplicar o modelo de responsabilidade compartilhada ao usar o AWS AppConfig, consulte Segurança em AWS Systems Manager. Essa seção descreve como configurar o Systems Manager para atingir os objetivos de segurança e conformidade do AWS AppConfig.

Implemente o acesso de privilégio mínimo

Como prática recomendada de segurança, conceda as permissões mínimas exigidas pelas identidades para realizar ações específicas em recursos específicos sob condições específicas. AWS AppConfig O agente oferece dois recursos que permitem que o agente acesse o sistema de arquivos de uma instância ou contêiner: backup e gravação em disco. Se você habilitar esses recursos, verifique se somente o AWS AppConfig Agente tem permissões para gravar nos arquivos de configuração designados no sistema de arquivos. Verifique também se somente os processos necessários para ler esses arquivos de configuração têm a capacidade de fazer isso. A implementação do acesso de privilégio mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Para obter mais informações sobre a implementação do acesso com privilégios mínimos, consulte SEC03-BP02 Conceder acesso com privilégios mínimos no Guia do usuário. AWS Well-Architected Tool Para obter mais informações sobre os recursos do AWS AppConfig Agente mencionados nesta seção, consulteRecursos adicionais de recuperação.

Criptografia de dados em repouso no AWS AppConfig

O AWS AppConfig fornece criptografia por padrão para proteger os dados em repouso do cliente usando Chaves pertencentes à AWS.

Chaves pertencentes à AWS: o AWS AppConfig usa essas chaves por padrão para criptografar automaticamente os dados implantados pelo serviço e hospedados no armazenamento de dados do AWS AppConfig. Você não pode visualizar, gerenciar ou usar Chaves pertencentes à AWS, nem auditar seu uso. No entanto, você não precisa tomar nenhuma medida nem alterar nenhum programa para proteger as chaves que criptografam seus dados. Para obter mais informações, consulte Chaves pertencentes à AWS no AWS Key Management ServiceGuia do desenvolvedor.

Embora não seja possível desativar essa camada de criptografia nem selecionar um tipo alternativo de criptografia, você pode especificar uma chave gerenciada pelo cliente a ser usada ao salvar dados de configuração hospedados no armazenamento de dados do AWS AppConfig e ao implantar seus dados de configuração.

Chaves gerenciadas pelo cliente: o AWS AppConfig suporta o uso de uma chave simétrica gerenciada pelo cliente que você cria, possui e gerencia para adicionar uma segunda camada de criptografia sobre o Chave pertencente à AWS existente. Por ter controle total dessa camada de criptografia, você pode realizar tarefas como:

- Estabelecer e manter as políticas e concessões de chaves
- Estabelecer e manter políticas do IAM
- Ativar e desativar políticas de chaves
- Alternar os materiais de criptografia de chaves
- Adicionar etiquetas
- Criar aliases de chaves
- Programar a exclusão de chaves

Para obter mais informações, consulte Chave gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service.

O AWS AppConfig suporta chaves gerenciadas pelo cliente

O AWS AppConfig oferece suporte para criptografia de chaves gerenciadas pelo cliente para dados de configuração. Para versões de configuração salvas no armazenamento de dados hospedado pelo AWS AppConfig, os clientes podem definir um `KmsKeyIdentifier` no perfil de configuração correspondente. Cada vez que uma nova versão de dados de configuração é criada usando a operação `CreateHostedConfigurationVersion` da API, o AWS AppConfig gera uma chave de dados AWS KMS a partir de `KmsKeyIdentifier` para criptografar os dados antes de armazená-los. Quando os dados são acessados posteriormente, seja durante as operações `GetHostedConfigurationVersion` ou `StartDeployment` da API, o AWS AppConfig descriptografa os dados de configuração usando informações sobre a chave de dados gerada.

O AWS AppConfig também oferece suporte para criptografia de chaves gerenciadas pelo cliente para dados de configuração implantados. Para criptografar os dados de configuração, os clientes podem fornecer um `KmsKeyIdentifier` para sua implantação. O AWS AppConfig gera a chave de dados AWS KMS com esse `KmsKeyIdentifier` para criptografar dados na operação `StartDeployment` da API.

Acesso à criptografia do AWS AppConfig

Ao criar uma chave gerenciada pelo cliente, siga esta política de chaves para garantir que a chave possa ser usada.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "Allow use of the key",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::account_ID:role/role_name"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
            ],
        "Resource": "*"
        }
 ]
```

Para criptografar dados de configuração hospedados com uma chave gerenciada pelo cliente, a chamada de identidade `CreateHostedConfigurationVersion` precisa da seguinte declaração de política, que pode ser atribuída a um usuário, grupo ou função:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Se estiver usando um segredo do Secrets Manager ou qualquer outro dado de configuração criptografado com uma chave gerenciada pelo cliente, o `retrievalRoleArn` precisará de `kms:Decrypt` para descriptografar e recuperar os dados.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:account_ID:configuration source/object"
    }
  ]
}
```

Ao chamar a operação da AWS AppConfig StartDeploymentAPI, a chamada de identidade `StartDeployment` precisa da seguinte política do IAM, que pode ser atribuída a um usuário, grupo ou função:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
```

```
      ],
      "Resource": "arn:aws:kms:Region:account_ID:key_ID"
    }
  ]
}
```

Ao chamar a operação da AWS AppConfig GetLatestConfigurationAPI, a chamada de identidade `GetLatestConfiguration` precisa da seguinte política, que pode ser atribuída a um usuário, grupo ou função:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

Contexto de criptografia

Um contexto de criptografia é um conjunto opcional de pares chave-valor que contêm informações contextuais adicionais sobre os dados.

O AWS KMS usa o contexto de criptografia como dados autenticados adicionais para oferecer suporte à criptografia autenticada. Quando você inclui um contexto de criptografia em uma solicitação para criptografar dados, o AWS KMS vincula o contexto de criptografia aos dados criptografados. Para descriptografar os dados, você inclui o mesmo contexto de criptografia na solicitação.

Contexto de criptografia do AWS AppConfig: o AWS AppConfig usa um contexto de criptografia em todas as operações criptográficas de AWS KMS para dados de configuração hospedados criptografados e implantações. O contexto contém uma chave correspondente ao tipo de dados e um valor que identifica o dado específico.

Monitoramento das chaves de criptografia da AWS

Ao usar chaves gerenciadas pelo AWS KMS cliente comAWS AppConfig, você pode usar AWS CloudTrail o Amazon CloudWatch Logs para rastrear solicitações AWS AppConfig enviadas paraAWS KMS.

O exemplo a seguir é um CloudTrail evento `Decrypt` para monitorar AWS KMS operações chamadas por AWS AppConfig para acessar dados criptografados pela chave gerenciada pelo cliente:

```
{
    "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"
}
```

Acesse o AWS AppConfig usando um VPC endpoint de interface (AWS PrivateLink)

É possível usar o AWS PrivateLink para criar uma conectividade privada entre a sua VPC e o AWS AppConfig. Você pode acessar o AWS AppConfig como se estivesse em sua VPC, sem usar um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão AWS Direct Connect. As instâncias na sua VPC não precisam de endereços IP públicos para acessar AWS AppConfig.

Você estabelece essa conectividade privada criando um endpoint de interface, desenvolvido pelo AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Estas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao AWS AppConfig.

Para obter mais informações, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink.

Considerações para o AWS AppConfig

Antes de configurar um endpoint de interface para AWS AppConfig, revise as Considerações no Guia AWS PrivateLink.

AWS AppConfigsuporta a realização de chamadas para os `appconfigdata`serviços `appconfig`e por meio do endpoint da interface.

Criar um endpoint de interface para AWS AppConfig

Você pode criar um endpoint de interface para AWS AppConfig usando o console da Amazon VPC ou o AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criando um Endpoint de InterfaceAWS PrivateLink no Guia.

Crie um endpoint para o AWS AppConfig usando o seguinte nome de serviço:

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

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

Se você habilitar o DNS privado para o endpoint da interface, poderá fazer solicitações de API a AWS AppConfig usando seu nome DNS regional padrão. Por exemplo, `appconfig.us-east-1.amazonaws.com` e `appconfigdata.us-east-1.amazonaws.com`.

Criar uma política de endpoint para o endpoint da interface

Política de endpoint é um recurso do IAM que você pode anexar ao endpoint de interface. A política de endpoint padrão permite acesso total ao AWS AppConfig por meio de endpoint de interface. Para controlar o acesso permitido ao AWS AppConfig pela VPC, anexe uma política de endpoint personalizada ao endpoint de interface.

Uma política de endpoint especifica as seguintes informações:

- Os diretores que podem realizar ações (Contas da AWSusuários do IAM e funções do IAM).
- As ações que podem ser executadas.
- Os recursos nos quais as ações podem ser executadas.

Para obter mais informações, consulte Controlar o Acesso a Serviços Usando Políticas de Endpoint no AWS PrivateLink Guia.

Exemplo: política de endpoint da VPC para ações do AWS AppConfig

O exemplo a seguir refere-se a uma política de endpoint personalizada. Quando anexada ao endpoint da sua interface, essa política concede acesso às ações do AWS AppConfig listadas para todas as entidades principais em todos os recursos.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "appconfig:CreateApplication",
            "appconfig:CreateEnvironment",
            "appconfig:CreateConfigurationProfile",
            "appconfig:StartDeployment",
            "appconfig:GetLatestConfiguration"
            "appconfig:StartConfigurationSession"
```

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

Alternância de chaves do Secrets Manager

Esta seção descreve informações de segurança importantes sobre a integração do AWS AppConfig com o Secrets Manager. Para obter mais informações sobre o Secrets Manager, consulte O que é o AWS Secrets Manager? no Guia do usuário do AWS Secrets Manager.

Configuração da alternância automática dos segredos do Secrets Manager implantados pelo AWS AppConfig

Alternância é o processo de atualizar periodicamente um segredo armazenado no Secrets Manager. Quando o Secrets Manager alterna um segredo, ele atualiza as credenciais tanto no segredo quanto no banco de dados ou serviço. Você pode configurar a alternância automática de segredos no Secrets Manager usando uma função AWS Lambda para atualizar o segredo e o banco de dados. Para obter mais informações, consulte Alternar os segredos do AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

Para ativar a alternância de chaves de segredos do Secrets Manager implantados por AWS AppConfig, atualize sua função do Lambda para alternância e implante o segredo alternado.

> **Note**
>
> Implante seu perfil de configuração do AWS AppConfig depois que seu segredo for alternado e totalmente atualizado para a nova versão. Você pode determinar se o segredo foi alternado porque o status de `VersionStage` muda de `AWSPENDING` para `AWSCURRENT`. A conclusão da alternância de segredos ocorre dentro da função `finish_secret` dos modelos de alternância do Secrets Manager.

Veja um exemplo de função que inicia uma implantação do AWS AppConfig após a alternância de um segredo.

```
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))
```

Como monitorar o AWS AppConfig

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e a performance do AWS AppConfig e das outras soluções da AWS. A AWS fornece as ferramentas de monitoramento a seguir para observar o AWS AppConfig, informar quando algo está errado e realizar ações automaticamente quando apropriado:

- O AWS CloudTrail captura chamadas de API e eventos relacionados feitos por sua conta da AWS ou em nome dela e entrega os arquivos de log a um bucket do Amazon S3 especificado por você. Você pode identificar quais usuários e contas chamaram a AWS, o endereço IP de origem no qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o AWS CloudTrailGuia do Usuário.
- O Amazon CloudWatch Logs permite que você monitore, armazene e acesse seus arquivos de log das instâncias do Amazon EC2 e de outras fontes. CloudTrail CloudWatch Os registros podem monitorar as informações nos arquivos de log e notificá-lo quando determinados limites forem atingidos. Você também pode arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.

Tópicos

Registrar em log chamadas de API do AWS AppConfig usando o AWS CloudTrail

AWS AppConfigé integrado comAWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço emAWS AppConfig. CloudTrail captura todas as chamadas de API AWS AppConfig como eventos. As chamadas capturadas incluem as aquelas do AWS AppConfig console e chamadas de código para AWS AppConfig operações API. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para. AWS AppConfig Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feitaAWS

AppConfig, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

AWS AppConfiginformações em CloudTrail

CloudTrail é ativado no seu Conta da AWS quando você cria a conta. Quando a atividade ocorre emAWS AppConfig, essa atividade é registrada em um CloudTrail evento junto com outros eventos AWS de serviço no histórico de eventos. Você pode exibir, pesquisar e baixar eventos recentes em sua Conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para obter um registro contínuo de eventos na sua Conta da AWS, incluindo eventos para o AWS AppConfig, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra eventos de todas as regiões na partição da AWS e entrega os arquivos de log no bucket do Amazon S3 que você especificou Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- CloudTrail serviços e integrações suportados
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e Recebendo arquivos de CloudTrail log de várias contas

Todas AWS AppConfig as ações são registradas CloudTrail e documentadas na Referência da AWS AppConfig API. Por exemplo, chamadas para o `CreateApplication` `GetApplication` e `ListApplications` as ações geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou usuário do IAM AWS Identity and Access Management
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.

- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

AWS AppConfigeventos de dados em CloudTrail

Os eventos de dados fornecem informações sobre as operações de recursos realizadas em ou em um recurso (por exemplo, recuperar a configuração mais recente implantada por meio de uma chamada GetLatestConfiguration). Elas também são conhecidas como operações de plano de dados. Eventos de dados geralmente são atividades de alto volume. Por padrão, CloudTrail não registra eventos de dados. O histórico de CloudTrail eventos não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrailPreços.

Você pode registrar eventos de dados para os tipos de AWS AppConfig recursos usando o CloudTrail console ou AWS CLI as operações CloudTrail da API. A tabela nesta seção mostra os tipos de recursos disponíveis paraAWS AppConfig.

- Para registrar eventos de dados usando o CloudTrail console, crie um armazenamento de dados de trilhas ou eventos para registrar eventos de dados ou atualize um armazenamento de dados de trilhas ou eventos existente para registrar eventos de dados.

  1. Escolha Eventos de dados para registrar eventos de dados.
  2. Na lista Tipo de evento de dados, escolha AWS AppConfig.
  3. Escolha o modelo do seletor de registros que você deseja usar. Você pode registrar todos os eventos de dados do tipo de recurso, registrar todos os `readOnly` eventos, registrar todos os `writeOnly` eventos ou criar um modelo de seletor de registros personalizado para filtrar os `resources.ARN` campos `readOnly` `eventName`, e.
  4. Em Nome do seletor, insira AppConfigDataEvents. Para obter informações sobre como habilitar o Amazon CloudWatch Logs para sua trilha de eventos de dados, consulteMétricas de registro para chamadas AWS AppConfig de planos de dados.
- Para registrar eventos de dados usando oAWS CLI, configure o `--advanced-event-selectors` parâmetro para definir o `eventCategory` campo igual `Data` e o `resources.type` campo igual ao valor do tipo de recurso (consulte a tabela). Você pode adicionar condições para filtrar os valores dos `resources.ARN` campos `readOnly` `eventName`, e.

- Para configurar uma trilha para registrar eventos de dados, execute o put-event-selectorscomando. Para obter mais informações, consulte Registro de eventos de dados para trilhas com AWS CLI o.
- Para configurar um armazenamento de dados de eventos para registrar eventos de dados, execute o create-event-data-storecomando para criar um novo armazenamento de dados de eventos para registrar eventos de dados ou execute o update-event-data-storecomando para atualizar um armazenamento de dados de eventos existente. Para obter mais informações, consulte Registro de eventos de dados para armazenamentos de dados de eventos com AWS CLI o.

A tabela a seguir lista os tipos de recursos do AWS AppConfig. A coluna Tipo de evento de dados (console) mostra o valor a ser escolhido na lista Tipo de evento de dados no CloudTrail console. A coluna de valor resources.type mostra o **`resources.type`** valor, que você especificaria ao configurar seletores de eventos avançados usando as APIs ou. AWS CLI CloudTrail A CloudTrail coluna Data APIs logged to mostra as chamadas de API registradas CloudTrail para o tipo de recurso.

Tipo de evento de dados (console)	valor resources.type	APIs de dados registradas em * CloudTrail
AWS AppConfig	`AWS::AppConfig::Configuration`	• GetLatestConfiguration • StartConfigurationSession

*Você pode configurar seletores de eventos avançados para filtrar os `resources.ARN` campos `eventNamereadOnly`, e e para registrar somente os eventos que são importantes para você. Consulte mais informações sobre esses campos em AdvancedFieldSelector.

AWS AppConfigeventos de gerenciamento em CloudTrail

Os eventos de gerenciamento fornecem informações sobre operações de gerenciamento executadas em recursos na sua conta da AWS. Elas também são conhecidas como operações de plano de controle. Por padrão, CloudTrail registra eventos de gerenciamento.

AWS AppConfigregistra todas as operações do plano de AWS AppConfig controle como eventos de gerenciamento. Para ver uma lista das operações do plano de AWS AppConfig controle AWS AppConfig registradas CloudTrail, consulte a Referência da AWS AppConfig API.

Noções básicas sobre entradas de arquivos de log do AWS AppConfig

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a StartConfigurationSessionação.


```
{
      "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"
      }
    }
```

Métricas de registro para chamadas AWS AppConfig de planos de dados

Se você configurou AWS CloudTrail para registrar eventos de AWS AppConfig dados, você pode permitir que o Amazon CloudWatch Logs registre métricas para chamadas para o plano de AWS AppConfig dados. Em seguida, você pode pesquisar e filtrar dados de registro no CloudWatch Logs criando um ou mais filtros de métrica. Os filtros de métricas definem os termos e padrões a serem procurados nos dados de registro à medida que são enviados para o CloudWatch Logs. CloudWatch O Logs usa filtros métricos para transformar dados de registro em CloudWatch métricas numéricas. Você pode representar graficamente as métricas ou configurá-las com um alarme.

Antes de começar

Ative o registro de eventos de AWS AppConfig dados emAWS CloudTrail. O procedimento a seguir descreve como ativar o registro de métricas para uma AWS AppConfigtrilha existente CloudTrail. Para obter informações sobre como habilitar o CloudTrail registro para chamadas AWS AppConfig de planos de dados, consulteAWS AppConfigeventos de dados em CloudTrail.

Use o procedimento a seguir para permitir que o CloudWatch Logs registre métricas para chamadas para o plano AWS AppConfig de dados.

Para permitir que o CloudWatch Logs registre métricas de chamadas para o plano AWS AppConfig de dados

1. Abra o CloudTrail console em https://console.aws.amazon.com/cloudtrail/.
2. No painel, escolha sua AWS AppConfig trilha.
3. Na seção CloudWatch Logs, selecione Editar.
4. Selecione Ativado.
5. Em Nome do grupo de registros, deixe o nome padrão ou insira um nome. Anote o nome. Posteriormente, você escolherá o grupo de CloudWatch registros no console de registros.
6. Em Nome do perfil, insira um nome.
7. Escolha Salvar alterações.

Use o procedimento a seguir para criar uma métrica e um filtro de métrica para o AWS AppConfig CloudWatch Logs. O procedimento descreve como criar um filtro métrico para chamadas de `operation` e (opcionalmente) chamadas de `operation` e. `Amazon Resource Name (ARN)`

Para criar uma métrica e um filtro de métrica para o AWS AppConfig CloudWatch Logs

1. Abra o CloudWatch console emhttps://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Logs e, em seguida, escolha Grupos de log.
3. Escolha a caixa de seleção ao lado do grupo de AWS AppConfig registros.
4. Escolha Actions (Ações) e Create metric filter (Criar filtro de métrica).
5. Em Nome do filtro, insira um nome.
6. Em Padrão de filtro, insira o seguinte:

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

7. (Opcional) Na seção Padrão de teste, escolha seu grupo de registros na lista Selecionar dados de registro para testar. Se CloudTrail não tiver registrado nenhuma chamada, você pode pular esta etapa.
8. Escolha Próximo.
9. Em Metric namespace, insira. **`AWS AppConfig`**

10. Em Metric name (Nome da métrica), insira **`Calls`**.
11. Em Metric Value (Valor de métrica), insira **`1`**.
12. Ignore o valor padrão e a unidade.
13. Em Nome da dimensão, insira**`operation`**.
14. Em Valor da dimensão, insira**`$.eventName`**.

    (Opcional) Você pode inserir uma segunda dimensão que inclua o Amazon Resource Name (ARN) fazendo a chamada. Para adicionar uma segunda dimensão, em Nome da dimensão, insira**`resource`**. Em Valor da dimensão, insira**`$.resources[0].ARN`**.

    Escolha Próximo.
15. Revise os detalhes do filtro e crie um filtro métrico.

(Opcional) Você pode repetir esse procedimento para criar um novo filtro métrico para um código de erro específico, como AccessDenied. Se você fizer isso, insira os seguintes detalhes:

1. Em Nome do filtro, insira um nome.
2. Em Padrão de filtro, insira o seguinte:

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

   Por exemplo

   ```
   { $.errorCode = "AccessDenied" }
   ```
3. Em Metric namespace, insira. **`AWS AppConfig`**
4. Em Metric name (Nome da métrica), insira **`Errors`**.
5. Em Metric Value (Valor de métrica), insira **`1`**.
6. Em Valor padrão, insira um zero (0).
7. Ignore a unidade, as dimensões e os alarmes.

Depois de CloudTrail registrar as chamadas de API, você pode ver as métricas em CloudWatch. Para obter mais informações, consulte Visualizar suas métricas e registros no console no Guia do CloudWatch usuário da Amazon. Para obter informações sobre como localizar uma métrica que você criou, consulte Pesquisar métricas disponíveis.

> **Note**
>
> Se você configurar a métrica de erro sem dimensão, conforme descrito aqui, poderá visualizar essas métricas na página Métricas sem dimensão.

Criando um alarme para uma CloudWatch métrica

Depois de criar métricas, você pode criar alarmes métricos em CloudWatch. Por exemplo, você pode criar um alarme para a métrica de AWS AppConfigchamadas criada no procedimento anterior. Especificamente, você pode criar um alarme para chamadas para a ação da AWS AppConfig `StartConfigurationSession` API que ultrapassam um limite. Para obter informações sobre como criar um alarme para uma métrica, consulte Criar um CloudWatch alarme com base em um limite estático no Guia do CloudWatch usuário da Amazon. Para obter informações sobre limites padrão para chamadas para o plano de AWS AppConfig dados, consulte Limites padrão do plano de dados no Referência geral da Amazon Web Services.

AWS AppConfig Histórico do documento do Guia do Usuário

A tabela a seguir descreve as mudanças importantes na documentação desde a última versão do AWS AppConfig.

Versão atual da API: 09/10/2019

Alteração	Descrição	Data
AWS AppConfig amostras de extensões personalizadas	O tópico Passo a passo: Criação de AWS AppConfig extensões personalizadas agora inclui links para os seguintes exemplos de extensões em: GitHub • Exemplo de extensão que impede implantações com um calendário de `blocked day` moratória usando o Systems Manager Change Calendar • Extensão de amostra que evita que segredos vazem para os dados de configuração usando git-secrets • Extensão de amostra que impede que informações de identificação pessoal (PII) vazem para os dados de configuração usando o Amazon Comprehend	28 de fevereiro de 2024
Novo tópico: Registro de chamadas de AWS AppConfig API usando AWS CloudTrail	AWS AppConfig é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizada	18 de janeiro de 2024

	s por um usuário, função ou AWS serviço em AWS AppConfig. CloudTrail captura todas as chamadas de API AWS AppConfig como eventos. Esse novo tópico fornece conteúdo AWS AppConfig específic o em vez de um link para o conteúdo correspondente no Guia do AWS Systems Manager usuário. Para obter mais informações, consulte Logging AWS AppConfig API chamando usando AWS CloudTrail.	
AWS AppConfig agora suporta AWS PrivateLink	Você pode usar AWS PrivateLi nk para criar uma conexão privada entre sua VPC e. AWS AppConfig Você pode acessar AWS AppConfig como se estivesse em sua VPC, sem o uso de um gateway de internet, dispositivo NAT, conexão VPN ou conexão. AWS Direct Connect As instâncias na sua VPC não precisam de endereços IP públicos para acessar AWS AppConfig. Para obter mais informações, consulte Acesso AWS AppConfig usando um endpoint de interface (AWS PrivateLink).	6 de dezembro de 2023

Recursos adicionais de recuperação de AWS AppConfig agentes e um novo modo de desenvolvimento local	AWS AppConfig O agente oferece os seguintes recursos adicionais para ajudá-lo a recuperar configurações para seus aplicativos. Recursos adicionais de recuperação • Recuperação de várias contas: use o AWS AppConfig Agente de uma conta primária ou de recuperação Conta da AWS para recuperar dados de configuração de várias contas de fornecedores. • Gravar cópia da configura ção no disco: use o AWS AppConfig Agente para gravar dados de configura ção no disco. Esse recurso permite que os clientes com aplicativos que leem dados de configuração do disco se integrem AWS AppConfig. **Note** A configuração de gravação em disco não foi projetada como um recurso de backup de configuração. AWS AppConfig O agente não lê os arquivos de	1.º de dezembro de 2023

configuração copiados para o disco. Se você quiser fazer backup das configurações em disco, consulte as variáveis de `PRELOAD_BACKUP` ambiente `BACKUP_DIRECTORY` e de uso do agente com o Amazon EC2 ou o uso do AWS AppConfig agente com AWS AppConfig o Amazon ECS e o Amazon EKS.

Modo de desenvolvimento local

AWS AppConfig O agente suporta um modo de desenvolvimento local. Se você ativar o modo de desenvolvimento local, o agente lerá os dados de configuração de um diretório especificado no disco. Ele não recupera dados de configuração de AWS AppConfig. Você pode simular implantações de configuração atualizando arquivos no diretório especificado. Recomendamos o modo de desenvolvimento local para os seguintes casos de uso:

	• Teste diferentes versões de configuração antes de implantá-las usando o. AWS AppConfig • Teste diferentes opções de configuração para um novo recurso antes de confirmar as alterações no seu repositório de código. • Teste diferentes cenários de configuração para verificar se eles funcionam conforme o esperado.	
Novo tópico de exemplos de código	Foi adicionado um novo tópico de exemplos de código a este guia. O tópico inclui exemplos em Java, Python e JavaScript para executar programaticamente seis ações comuns. AWS AppConfig	17 de novembro de 2023
Índice revisado para refletir melhor o fluxo de trabalho do AWS AppConfig	O conteúdo deste guia do usuário está agora agrupado sob os títulos Criação, Implantação, Recuperação e Extensão de fluxos de trabalho. Essa organização reflete melhor o fluxo de trabalho de uso do AWS AppConfig e tem como objetivo ajudar a tornar o conteúdo mais fácil de ser descoberto.	7 de novembro de 2023

Adição da referência de carga útil	O tópico Criação de uma função do Lambda para uma extensão personalizada do AWS AppConfig agora inclui uma referência de carga útil de solicitação e resposta.	7 de novembro de 2023
Nova estratégia de implantação AWS predefinida	AWS AppConfig agora oferece e recomenda a estratégia de implantação `AppConfig.Linear20PercentEvery6Minutes` predefinida. Para obter mais informações, consulte estratégias de implantação predefinidas.	11 de agosto de 2023
AWS AppConfig integração com o Amazon EC2	Você pode se integrar AWS AppConfig com aplicativos executados em suas instâncias Linux do Amazon Elastic Compute Cloud (Amazon EC2) usando o Agent. AWS AppConfig O agente oferece suporte às arquiteturas x86_64 e ARM64 para o Amazon EC2. Para obter mais informações, consulte Integração do AWS AppConfig com o Amazon EC2.	20 de julho de 2023

AWS CloudFormation suporte para novos AWS AppConfig recursos e um exemplo de sinalizador de recursos	AWS CloudFormation agora oferece suporte ao AWS::AppConfig::Extensione aos AWS::AppConfig::ExtensionAssociationrecursos para ajudar você a começar a usar AWS AppConfig extensões. Os recursos AWS::AppConfig::ConfigurationProfilee AWS::AppConfig::HostedConfigurationVersion agora incluem um exemplo para criar um perfil de configuração de sinalizador de recurso no armazenamento de configuração AWS AppConfig hospedado.	12 de abril de 2023

AWS AppConfig integração com AWS Secrets Manager	AWS AppConfig se integra com AWS Secrets Manager. O Secrets Manager ajuda você a criptografar, armazenar e recuperar credenciais com segurança para bancos de dados e outros serviços. Em vez de codificar credencia is em seus aplicativos, você pode fazer chamadas ao Secrets Manager para recuperar suas credencia is sempre que necessário. O Secrets Manager ajuda você a proteger o acesso aos seus recursos e dados de TI, permitindo que você alterne e gerencie o acesso aos seus segredos. Ao criar um perfil de configura ção de formato livre, você pode escolher o Secrets Manager como fonte de seus dados de configuração. Você deve se integrar ao Secrets Manager e criar um segredo antes de criar o perfil de configuração. Para obter mais informações sobre o Secrets Manager, consulte O que é AWS Secrets Manager? no Guia do AWS Secrets Manager usuário. Para obter informações sobre como criar um perfil de configuração,	2 de fevereiro de 2023

consulte Criação de um perfil de configuração de formato livre.

AWS AppConfig integração com o Amazon ECS e o Amazon EKS	Você pode se integrar AWS AppConfig ao Amazon Elastic Container Service (Amazon ECS) e ao Amazon Elastic Kubernetes Service (Amazon EKS) usando o agente. AWS AppConfig O agente funciona como um contêiner auxiliar executado junto com seus aplicativos de contêiner Amazon ECS e Amazon EKS. O agente aprimora o processamento e o gerenciamento de aplicativos em contêineres das seguintes maneiras: • O agente liga AWS AppConfig em seu nome usando uma função AWS Identity and Access Management (IAM) e gerenciando um cache local de dados de configuração. Ao extrair dados de configuração do cache local, seu aplicativo exige menos atualizações de código para gerenciar dados de configuração, recupera dados de configuração em milissegundos e não é afetado por problemas de rede que podem afetar as chamadas para esses dados.	2 de dezembro de 2022

- O agente oferece uma experiência nativa para recuperar e resolver sinalizadores de AWS AppConfig recursos.
- Pronto para uso, o agente fornece as práticas recomendadas para estratégias de armazenamento em cache, intervalos de pesquisa e disponibilidade de dados de configuração local, enquanto rastreia os tokens de configuração necessários para chamadas de serviço subsequentes.
- Durante a execução em segundo plano, o agente consulta periodicamente o plano de AWS AppConfig dados para atualizações de dados de configuração. Seu aplicativo em contêiner pode recuperar os dados conectando-se ao localhost na porta 2772 (um valor de porta padrão personalizável) e chamando HTTP GET para recuperar os dados.
- O AWS AppConfig agente atualiza os dados de configuração em seus contêineres sem precisar reiniciá-los ou reciclá-los.

Para obter mais informações, consulte Integração do AWS AppConfig com o Amazon ECS e o Amazon EKS.

Nova extensão: AWS AppConfig extensão para CloudWatch Evidently	Você pode usar o Amazon CloudWatch Evidently para validar com segurança novos recursos, disponibilizando-os a uma porcentagem específica de seus usuários enquanto você implanta o recurso. Você pode monitorar a performance do novo recurso para auxiliar na decisão de quando aumentar o tráfego para seus usuários. Isso ajuda você a reduzir riscos e identificar consequências não intencionais antes de iniciar totalmente o recurso. Você também pode realizar experimentos A/B para decidir sobre design de recursos com base em evidências e dados. A AWS AppConfig extensão do CloudWatch Evidently permite que seu aplicativo atribua variações às sessões do usuário localmente, em vez de chamar a EvaluateFeatureoperação. Uma sessão local reduz os riscos de latência e disponibilidade associados a uma chamada de API. Para obter informações sobre como configurar e usar a extensão, consulte Executar lançamentos e experimentos A/B com o	13 de setembro de 2022

	CloudWatch Evidently no Guia CloudWatch do usuário da Amazon.	
Desaprovação da ação `GetConfiguration` da API	Em 18 de novembro de 2021, AWS AppConfig lançou um novo serviço de plano de dados. Esse serviço substitui o processo anterior de recuperação de dados de configuração usando a ação `GetConfiguration` da API. O serviço de plano de dados usa duas novas ações de API StartConfiguration SessionGetLatestConfigura tione. O serviço de plano de dados também usa novos endpoints. Para obter mais informações, consulte Sobre o serviço AWS AppConfig de plano de dados.	13 de setembro de 2022
Nova versão da extensão AWS AppConfig Agent Lambda	A versão 2.0.122 da extensão Agent AWS AppConfig Lambda já está disponíve l. A nova extensão usa diferentes nomes de recursos da Amazon (ARNs). Para obter mais informações, consulte Notas de versão da extensão do Lambda do AWS AppConfig Agent.	23 de agosto de 2022

Lançamento de AWS AppConfig extensões	Uma extensão aumenta sua capacidade de injetar lógica ou comportamento em diferentes pontos durante o AWS AppConfig fluxo de trabalho de criação ou implantação de uma configuração. Você pode usar extensões AWS criadas por - autoria ou criar suas próprias. Para obter mais informações, consulte Trabalhando com AWS AppConfig extensões.	12 de julho de 2022
Nova versão da extensão AWS AppConfig Agent Lambda	A versão 2.0.58 da extensão Agent AWS AppConfig Lambda já está disponível. A nova extensão usa diferente s nomes de recursos da Amazon (ARNs). Para obter mais informações, consulte Versões disponíveis da extensão AWS AppConfig Lambda.	3 de maio de 2022

AWS AppConfig integração com Atlassian Jira	A integração com o Atlassian Jira permite AWS AppConfig criar e atualizar problemas no console Atlassian sempre que você fizer alterações em um sinalizador de recurso em seu para o especificado. Conta da AWS Região da AWS Cada problema do Jira inclui o nome do sinalizador, o ID do aplicativo, o ID do perfil de configuração e os valores do sinalizador. Depois de atualizar, salvar e implantar as alterações do seu sinalizador, o Jira atualiza os problemas existentes com os detalhes da alteração. Para obter mais informações, consulte Integração do AWS AppConfig com o Atlassian Jira.	7 de abril de 2022

Disponibilidade geral de sinalizadores de atributos e suporte à extensão do Lambda para processadores ARM64 (Graviton2)	Com sinalizadores de AWS AppConfig recursos, você pode desenvolver um novo recurso e implantá-lo na produção enquanto oculta o recurso dos usuários. Você começa adicionando o sinalizador AWS AppConfig como dados de configura ção. Quando o atributo estiver pronto para ser lançado, você poderá atualizar os dados de configuração do sinalizador sem implantar nenhum código. Esse atributo melhora a segurança do seu ambiente de desenvolvimento e operações porque você não precisa implantar nenhum novo código para liberar o atributo. Para obter mais informações, consulte Criar um perfil de configuração de sinalizadores de atributos. A disponibilidade geral dos sinalizadores de atributos no AWS AppConfig inclui os seguintes aprimoramentos: • O console inclui uma opção para designar um sinalizad or como um sinalizador de curto prazo. Você pode filtrar e classificar a lista de sinalizadores em sinalizad ores de curto prazo.	15 de março de 2022

	• Para clientes que usam sinalizadores de atributos no AWS Lambda, a nova extensão do Lambda permite que você chame sinalizadores de atributos individuais usando um endpoint HTTP. Para obter mais informações, consulte Recuperação de um ou mais sinalizadores a partir de uma configuração de sinalizadores de atributos. Essa atualização também fornece suporte para extensões do AWS Lambda desenvolvidas para processadores ARM64 (Graviton2). Para obter mais informações, consulte Versões disponíveis da extensão AWS AppConfig Lambda.	
A ação GetConfiguration da API está obsoleta	A ação `GetConfiguration` da API está obsoleta. Em vez disso, as chamadas para receber dados de configuração devem usar as APIs `StartConfigurationSession` e `GetLatestConfiguration`. Para obter mais informações sobre essas APIs e como usá-las, consulte Recuperação da configuração.	28 de janeiro de 2022

Novo ARN de região para extensão AWS AppConfig Lambda	AWS AppConfig A extensão Lambda está disponível na nova região Ásia-Pacífico (Osaka). O nome do recurso da Amazon (ARN) é necessário para criar um Lambda na região. Para obter mais informações sobre o ARN da região Ásia-Pacífico (Osaka), consulte Adicionar a extensão AWS AppConfig Lambda.	4 de março de 2021
AWS AppConfig Extensão Lambda	Se você usa AWS AppConfig para gerenciar configurações para uma função Lambda, recomendamos que você adicione a extensão AWS AppConfig Lambda. Essa extensão inclui as melhores práticas que AWS AppConfig simplificam o uso e reduzem os custos. Custos reduzidos resultam de menos chamadas de API para o AWS AppConfig serviço e, separadamente, custos reduzidos de tempos mais curtos de processamento da função Lambda. Para obter mais informações, consulte Integração do AWS AppConfig com extensões do Lambda.	8 de outubro de 2020

Nova seção	Foi adicionada uma nova seção que fornece instruções para configuração do AWS AppConfig. Para obter mais informações, consulte Configurar AWS AppConfig.	30 de setembro de 2020
Adição de procedimentos de linha de comando	Os procedimentos neste guia do usuário agora incluem etapas de linha de comando para o AWS Command Line Interface (AWS CLI) e o Tools for Windows PowerShell. Para obter mais informações, consulte Trabalhando com AWS AppConfig.	30 de setembro de 2020
Lançamento do guia AWS AppConfig do usuário	Use AWS AppConfig a capacidade de criar AWS Systems Manager, gerenciar e implantar rapidamente configurações de aplicativos. AWS AppConfig oferece suporte a implantações controladas em aplicativos de qualquer tamanho e inclui verificações e monitoramento de validação integrados. Você pode usar AWS AppConfig com aplicativos hospedados em instâncias do EC2 AWS Lambda, contêineres, aplicativos móveis ou dispositivos de IoT.	31 de julho de 2020

Glossário do AWS

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência do Glossário da AWS.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.