Manual do usuário

AWS Proton

AWS Proton: Manual do usuário

Table of Contents

Aviso de fim do suporte: em 7 de outubro de 2026, AWS encerrará o suporte para AWS Proton. Depois de 7 de outubro de 2026, você não poderá mais acessar o AWS Proton console ou os AWS Proton recursos. Sua infraestrutura implantada permanecerá intacta. Para obter mais informações, consulte o Guia AWS Proton de descontinuação e migração de serviços.

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.

O que AWS Protoné

AWS Proton é:

- Infraestrutura automatizada como provisionamento de código e implantação de aplicações de tecnologia sem servidor baseadas em contêiner

  O AWS Proton serviço é uma estrutura de automação em duas frentes. Como administrador, você cria modelos de serviço versionados que definem a infraestrutura padronizada e as ferramentas de implantação para aplicações de tecnologia sem servidor baseadas em contêiner. Como desenvolvedor de aplicações, você pode selecionar entre os modelos de serviço disponíveis para automatizar suas implantações de aplicações ou serviços.

  AWS Proton identifica todas as instâncias de serviço existentes que estão usando uma versão de modelo desatualizada para você. Como administrador, você pode solicitar AWS Proton a atualização deles com um clique.
- Infraestrutura padronizada

  As equipes da plataforma podem usar AWS Proton uma infraestrutura versionada como modelos de código. Eles podem usar esses modelos para definir e gerenciar pilhas de aplicativos padrão que contêm a arquitetura, os recursos de infraestrutura e o pipeline de implantação CI/CD de software.
- Implantações integradas com CI/CD

  Quando os desenvolvedores usam a interface de AWS Proton autoatendimento para selecionar um modelo de serviço, eles estão selecionando uma definição padronizada de pilha de aplicativos para suas implantações de código. AWS Proton provisiona automaticamente os recursos, configura o CI/CD pipeline e implanta o código na infraestrutura definida.

AWS Proton para equipes de plataforma

Como administrador, você ou membros da sua equipe de plataforma criam modelos de ambiente e modelos de serviço contendo infraestrutura como código. O modelo de ambiente define a infraestrutura compartilhada usada por várias aplicações ou recursos. O modelo de serviço define o tipo de infraestrutura necessária para implantar e manter uma única aplicação ou um único microsserviço em um ambiente. Um AWS Proton serviço é uma instanciação de um modelo de serviço, que normalmente inclui várias instâncias de serviço e um pipeline. Uma instância de serviço

do AWS Proton é uma instanciação de um modelo de serviço em um ambiente específico. Você ou outras pessoas da sua equipe podem especificar quais modelos de ambiente são compatíveis com um determinado modelo de serviço. Para mais informações sobre modelos, consulte AWS Proton modelos.

Você pode usar a seguinte infraestrutura como provedores de código com AWS Proton:

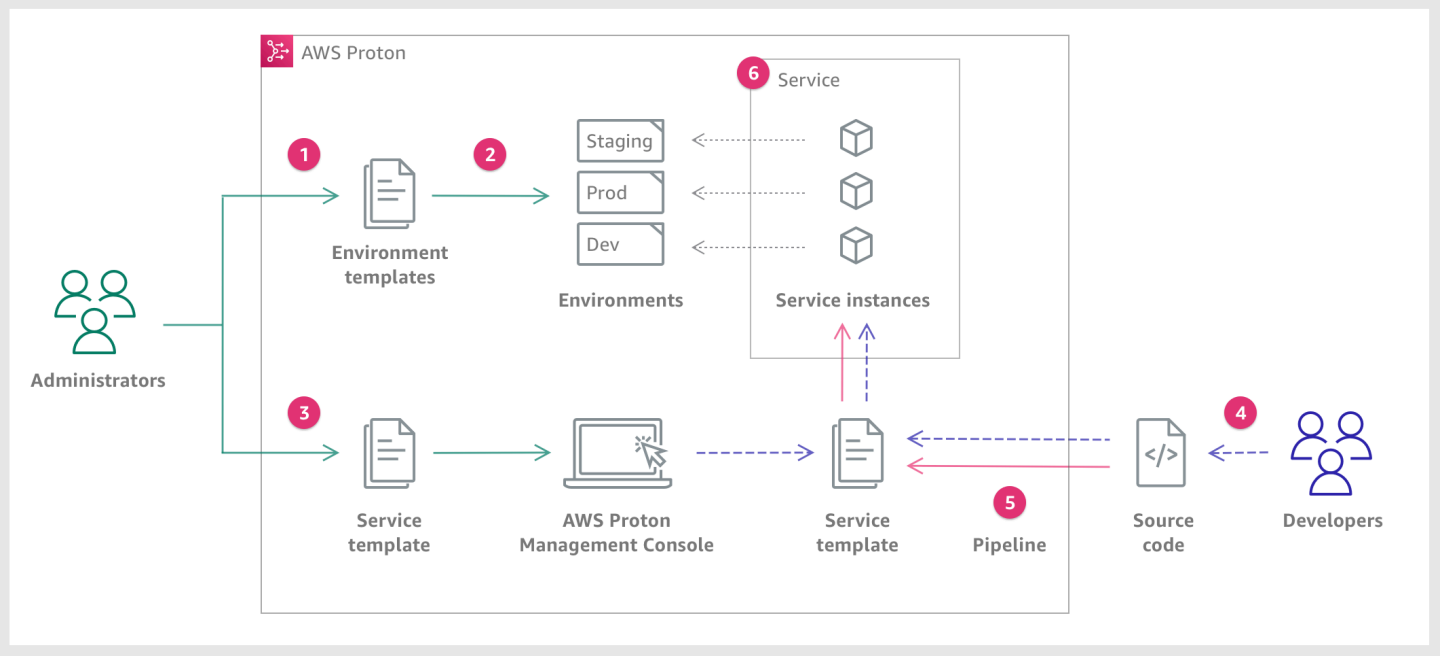
- CloudFormation
- Terraform

AWS Proton para desenvolvedores

Como desenvolvedor de aplicativos, você seleciona um modelo de serviço padronizado AWS Proton usado para criar um serviço que implanta e gerencia seu aplicativo em uma instância de serviço. Um serviço do AWS Proton é uma instanciação de um modelo de serviço, que normalmente inclui várias instâncias de serviço e um pipeline.

AWS Proton fluxo de trabalho

O diagrama a seguir é uma visualização dos principais AWS Proton conceitos discutidos no parágrafo anterior. Ele também oferece uma visão geral de alto nível do que constitui um fluxo de trabalho simples AWS Proton .



1. Como administrador, você cria e registra um modelo de ambiente com AWS Proton, que define os recursos compartilhados.
2. AWS Proton implanta um ou mais ambientes, com base em um modelo de ambiente.
3. Como administrador, você cria e registra um modelo de serviço com AWS Proton, que define a infraestrutura, o monitoramento e CI/CD os recursos relacionados, bem como os modelos de ambiente compatíveis.
4. Como desenvolvedor, você seleciona um modelo de serviço registrado e fornece um link para seu repositório de códigos-fonte.
5. AWS Proton provisiona o Serviço com um pipeline de CI/CD para suas instâncias de Serviço.
6. AWS Proton provisiona e gerencia o Serviço e as Instâncias de Serviço que estão executando o código-fonte conforme definido no Modelo de Serviço selecionado. Uma instância de serviço é uma instanciação do modelo de serviço selecionado em um ambiente para um único estágio de um pipeline (por exemplo, Prod).

AWS Proton Guia de descontinuação e migração de serviços

AWS decidiu descontinuar AWS Proton, com o suporte terminando em 7 de outubro de 2026. Novos clientes não poderão se inscrever após 7 de outubro de 2025, mas os clientes existentes poderão continuar usando o serviço até 7 de outubro de 2026.

Status do serviço até a suspensão de uso

Até 7 de outubro de 2026, AWS Proton os clientes existentes podem continuar usando o serviço normalmente. Durante esse período, AWS irá:

1. Forneça patches de segurança e correções de bugs críticos
2. Mantenha a disponibilidade e o desempenho do serviço
3. Continue oferecendo suporte por meio de AWS Support canais
4. Não adicionar novos recursos ao serviço

Informações importantes sobre migração

AWS Proton é principalmente uma CI/CD ferramenta para implantar infraestrutura. Quando AWS Proton for descontinuado, suas CloudFormation pilhas implantadas e os recursos que elas gerenciam permanecerão intactos e continuarão funcionando. A suspensão de uso afeta somente os pipelines de entrega e o AWS Proton serviço em si, não a infraestrutura implantada.

Soluções alternativas

Identificamos várias alternativas AWS Proton que podem ajudar você a manter sua infraestrutura como código e recursos de CI/CD.

CloudFormation Sincronização com Git

Ideal para: equipes CloudFormation que usam quem quer um GitOps fluxo de trabalho

O Git sync permite que as equipes da plataforma modelem CloudFormation modelos em um repositório git que as equipes de desenvolvimento podem bifurcar. Os desenvolvedores atualizam arquivos de parâmetros, enviam alterações para seus repositórios bifurcados e o Git sync atualiza a pilha.

Principais benefícios:

1. Experiência de desenvolvedor semelhante à AWS Proton
2. Aproveita o conhecimento existente CloudFormation
3. Separação clara entre a plataforma e as equipes de desenvolvedores

Limitações:

1. Nenhum conceito de ambientes
2. Sem recursos avançados de pipeline
3. Depende de GitHub recursos que podem não estar disponíveis em outros provedores de Git

Saiba mais: Git sync

Harmonix On AWS

Ideal para: empresas que precisam de um portal interno abrangente para desenvolvedores

O Harmonix é uma AWS Partner solução baseada no Backstage.io e fornece um AWS plug-in que permite às equipes criar modelos, ambientes e serviços semelhantes ao Proton.

Principais benefícios:

1. Funcionalidade semelhante à AWS Proton
2. Construído com base na popular estrutura Backstage
3. Experiência completa no portal do desenvolvedor

Limitações:

1. Não é mantido por uma AWS service (Serviço da AWS) equipe
2. Implementação de referência que pode exigir personalização

Saiba mais: https://harmonixonaws.io/

AWS CodePipeline and AWS CodeBuild

Ideal para: equipes que precisam do máximo de flexibilidade e controle

Use os CI/CD serviços AWS básicos para replicar a AWS Proton funcionalidade com maior flexibilidade e controle.

Principais benefícios:

1. Flexibilidade máxima
2. Integração profunda com AWS serviços
3. Manutenção ativa e novos recursos

Limitações:

1. Requer mais trabalho de implementação

2. Menos autoatendimento para out-of-box desenvolvedores

Saiba mais:

O que é AWS CodePipeline

O que é AWS CodeBuild

GitHub Ações

Ideal para: equipes menores GitHub que usam quem quer simplicidade

>Principais benefícios

1. Fácil integração com GitHub repositórios
2. Configuração simples para GitHub usuários
3. Grande mercado de ações reutilizáveis

Limitações:

1. Vinculado ao GitHub ecossistema
2. Pode exigir mais trabalho para os controles da equipe da plataforma

Saiba mais:

GitHub Documentação de ações

Exemplo de CI/CD: integração com GitHub ações — CI/CD pipeline para implantar um aplicativo web na Amazon EC2

Orientação de migração

O processo de migração depende da implementação e da alternativa escolhida. Etapas gerais:

1. Inventarie seus recursos do Proton:
2. Selecione uma solução alternativa:
3. Extraia os dados do seu modelo:
4. Implemente sua alternativa escolhida:

5. Migre as cargas de trabalho de produção:

Para obter assistência específica sobre migração, entre em contato com AWS Support nossa equipe de conta.

FAQs

P: Por que está AWS descontinuando AWS Proton? R: Identificamos melhores oportunidades para atender às necessidades dos clientes em relação à aplicação de políticas de infraestrutura como código por meio AWS de outras AWS Partner soluções.

P: Minha infraestrutura existente continuará funcionando após a data de suspensão de uso? R: Sim. AWS Proton é principalmente uma CI/CD ferramenta. Suas CloudFormation pilhas implantadas e os recursos que elas gerenciam permanecerão intactos e continuarão funcionando. A suspensão de uso afeta somente os pipelines de entrega, não sua infraestrutura implantada.

P: Como posso obter ajuda com a migração? R: AWS Support pode ajudar na sua migração. Entre em contato AWS Supportou entre em contato com seu Conta da AWS gerente para obter ajuda.

P: Qual alternativa devo escolher? R: A melhor alternativa depende do seu caso de uso específico:

1. Para um GitOps fluxo de trabalho simples: CloudFormation Git Sync
2. Para empresas que precisam de um portal para desenvolvedores: Harmonix On AWS
3. Para máxima flexibilidade: AWS CodePipeline e AWS CodeBuild
4. Para equipes que já estão ativas GitHub: GitHub Ações

P: O que acontece se eu não migrar até 7 de outubro de 2026? R: Você não poderá mais acessar AWS Proton. Sua infraestrutura existente continuará funcionando, mas você não poderá usá-la AWS Proton para gerenciá-la ou atualizá-la.

P: Por quanto tempo meus dados serão retidos? R: Até 7 de outubro de 2026. Após essa data, todos os dados serão excluídos.

Se você tiver mais perguntas, entre em contato AWS Support.

Configuração

Conclua as tarefas desta seção para que você possa criar e registrar modelos de serviço e ambiente. Você precisa deles para implantar ambientes e serviços AWS Proton.

> **Note**
>
> Estamos oferecendo sem AWS Proton custo adicional. Você pode criar, registrar e manter modelos de serviços e ambientes sem nenhum custo. Você também pode contar com AWS Proton o autogerenciamento de suas próprias operações, como armazenamento, segurança e implantação. As únicas despesas que você incorre ao usar AWS Proton são as seguintes.
>
> - Custos de implantação e uso de Nuvem AWS recursos que você instruiu AWS Proton a implantar e manter para você.
> - Custos de manter uma AWS CodeStar conexão com seu repositório de código.
> - Custos de manutenção de um bucket do Amazon S3, se você usar um bucket para fornecer entradas para o AWS Proton. Você pode evitar esses custos se mudar para the section called “Configurações de sincronização de modelo” usando repositórios Git para o seus the section called “Pacotes de modelos”.

Tópicos

Configurar com IAM

Quando você se inscreve AWS, você Conta da AWS é automaticamente inscrito em todos os serviços em AWS, inclusive AWS Proton. Você será cobrado apenas pelos serviços e recursos que usar.

> **Note**
>
> Você e sua equipe, incluindo administradores e desenvolvedores, devem estar todos na mesma conta.

Inscreva-se para 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 a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo 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 prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Criar um usuário do IAM

Para criar um usuário administrador, selecione uma das opções a seguir.

Selecionar uma forma de gerenciar o administrador	Para	Por	Você também pode
Centro de Identidade do IAM (Recomendado)	Usar credenciais de curto prazo para acessar a AWS. Isso está de acordo com as práticas recomendadas de segurança. Para obter	Seguindo as instruções em Conceitos básicos no Guia do usuário do Centro de Identidade do AWS IAM .	Configure o acesso programático configurando o AWS CLI para uso Centro de Identidade do AWS IAM no Guia do AWS Command Line Interface usuário.

Selecionar uma forma de gerenciar o administrador	Para	Por	Você também pode
	informações sobre as práticas recomendadas, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.		
No IAM (Não recomendado)	Usar credenciais de longo prazo para acessar a AWS.	Seguindo as instruções em Criar um acesso de emergência para um usuário do IAM no Guia do usuário do IAM.	Configurar o acesso programático, com base em Gerenciar chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Configurando funções AWS Proton de serviço

Talvez você queira criar algumas funções do IAM para diferentes partes da sua AWS Proton solução. Você pode criá-los com antecedência usando o console do IAM ou pode usar o AWS Proton console para criá-los para você.

Crie funções de AWS Proton ambiente AWS Proton para permitir fazer chamadas de API para outros Serviços da AWS, como CloudFormation AWS CodeBuild, e vários serviços de computação e armazenamento, em seu nome, para provisionar recursos para você. Um perfil de provisionamento gerenciado pela AWSé necessário quando um ambiente ou qualquer uma das instâncias de serviço em execução nele usa o provisionamento gerenciado pela AWS. Uma CodeBuildfunção é necessária quando um ambiente ou qualquer uma de suas instâncias de serviço usa o CodeBuildprovisionamento. Para saber mais sobre as funções do AWS Proton ambiente, consultethe section called “Perfis do IAM”. Ao criar um ambiente, você pode usar o AWS Proton

console para escolher uma função existente para qualquer uma dessas duas funções ou para criar uma função com privilégios administrativos para você.

Da mesma forma, crie funções de AWS Proton pipeline AWS Proton para permitir fazer chamadas de API para outros serviços em seu nome para provisionar um pipeline de CI/CD para você. Para saber mais sobre as funções do AWS Proton pipeline, consultethe section called “Perfis de serviço de pipeline”. Para obter mais informações sobre como definir CI/CD configurações, consultethe section called “Configurando as configurações do CI/CD funil da conta”.

> **Note**
>
> Como não sabemos quais recursos você definirá em seus AWS Proton modelos, as funções que você cria usando o console têm amplas permissões e podem ser usadas tanto como funções de serviço do AWS Proton pipeline quanto como funções AWS Proton de serviço. Para implantações de produção, recomendamos que você defina o escopo das permissões para os recursos específicos que serão implantados criando políticas personalizadas para as funções de serviço de AWS Proton pipeline e as funções de serviço de AWS Proton ambiente. Você pode criar e personalizar essas funções usando o AWS CLI ou o IAM. Para obter mais informações, consulte Funções de serviço para AWS Proton e Criar um serviço.

Configurando com AWS Proton

Se você quiser usar o AWS CLI para executar AWS Proton APIs, verifique se você o instalou. Se ainda não o tiver instalado, consulte Configurando o AWS CLI.

AWS Proton configuração específica:

- Para criar e gerenciar modelos:
  - Se você estiver usando configurações de sincronização de modelos, configure uma conexão do AWS CodeStar.
  - Caso contrário, configure um bucket do Amazon S3.
- Para provisionar infraestrutura:
  - Para o provisionamento autogerenciado, você deve configurar uma conexão do AWS CodeStar.
- (Opcional) Para provisionar pipelines:
  - Para provisionamento AWS gerenciado e provisionamento CodeBuildbaseado, configure as funções do pipeline.

- Para provisionamento autogerenciado, configure um repositório de pipeline.

Para obter mais informações sobre métodos de provisionamento, consulte the section called "Provisionamento gerenciado pela AWS".

Configurar um bucket do Amazon S3

Para configurar um bucket do S3, siga as instruções em Crie seu primeiro bucket do S3 para configurar um bucket do S3. Coloque suas entradas AWS Proton no bucket onde AWS Proton você pode recuperá-las. Essas entradas são conhecidas como pacotes de modelos. Você pode aprender mais sobre eles em outras seções deste guia.

Configurando uma AWS CodeStar conexão

Para se conectar AWS Proton a um repositório, você cria uma AWS CodeStar conexão que ativa um pipeline quando um novo commit é feito em um repositório de código-fonte de terceiros.

AWS Proton usa a conexão para:

- Ativar um pipeline de serviço quando um novo commit for feito no código-fonte do seu repositório.
- Faça uma pull request em uma infraestrutura como repositório de código.
- Crie uma nova versão secundária ou principal do modelo sempre que um commit for enviado para um repositório de modelos que altera um dos seus modelos, caso a versão ainda não exista.

Você pode se conectar aos repositórios Bitbucket GitHub, GitHub Enterprise e GitHub Enterprise Server com o. CodeConnections Consulte mais informações em CodeConnections no Guia de Usuário AWS CodePipeline .

Para configurar uma CodeStar conexão.

1. Abra o console de AWS Proton.
2. No painel de navegação, selecione Configurações e, em seguida, Conexões do repositório para ir até a página Conexões nas Configurações das Ferramentas do Desenvolvedor. A página exibe uma lista de conexões.
3. Escolha Criar conexão e siga as instruções.

Configurando as configurações do CI/CD funil da conta

AWS Proton pode provisionar CI/CD pipelines para implantar o código do aplicativo em suas instâncias de serviço. As AWS Proton configurações necessárias para o provisionamento do pipeline dependem do método de provisionamento escolhido para o pipeline.

AWS-provisionamento gerenciado e CodeBuild baseado — configure funções de pipeline

Com provisionamento e provisionamento AWS gerenciados, CodeBuild provisiona pipelines para você. AWS Proton Portanto, AWS Proton precisa de uma função de serviço que forneça permissões para o provisionamento de pipelines. Cada um desses dois métodos de provisionamento usa seu próprio perfil de serviço. Essas funções são compartilhadas em todos os pipelines AWS Proton de serviço e você as configura uma vez nas configurações da sua conta.

Para criar perfis de serviços do pipeline usando o console

1. Abra o console de AWS Proton.
2. No painel de navegação, selecione Configurações e em seguida Configurações de conta.
3. Na página CI/CD Configurações da conta, escolha Configurar.
4. Execute um destes procedimentos:
   - Ter AWS Proton criado uma função de serviço de pipeline para você

     [Para habilitar o provisionamento de pipelines gerenciado pela AWS] Na página Definir configurações da conta, na seção perfil de pipeline de provisionamento gerenciado pela AWS:

     a. Selecione Novo perfil de serviço.
     b. Forneça um nome para o perfil, por exemplo, **`myProtonPipelineServiceRole`**.
     c. Marque a caixa de seleção para concordar em criar uma AWS Proton função com privilégios administrativos em sua conta.

     [Para ativar o provisionamento CodeBuild baseado em pipelines] Na página Definir configurações da conta, na seção Função do CodeBuild pipeline, escolha Função de serviço existente e escolha a função de serviço que você criou na seção Função do CloudFormation pipeline. Ou, se você não atribuiu uma função de CloudFormation pipeline, repita as três etapas anteriores para criar uma nova função de serviço.

- Para escolher perfis de serviço de pipeline existentes

  [Para ativar o provisionamento de pipelines gerenciado pela AWS] Na página Definir configurações de conta, na seção Perfil do pipeline de provisionamento gerenciado pela AWS, escolha Perfil de serviço existente e escolha um perfil de serviço em sua conta da AWS .

  [Para habilitar o CodeBuild provisionamento de pipelines] Na página Configurar configurações da conta, na seção Função de aprovisionamento de CodeBuild pipeline, escolha Função de serviço existente e escolha uma função de serviço em sua conta. AWS

5. Escolha Salvar alterações.

   Seu novo perfil de serviço de pipeline é exibido na página de configurações da conta.

Provisionamento autogerenciado—configure um repositório de pipeline.

Com o provisionamento autogerenciado, AWS Proton envia uma pull request (PR) para um repositório de provisionamento que você configurou, e seu código de automação é responsável pelo provisionamento de pipelines. Portanto, AWS Proton não precisa de uma função de serviço para provisionar pipelines. Em vez disso, ele precisa de um repositório de provisionamento registrado. Seu código de automação no repositório deve assumir um perfil apropriado que forneça permissões para o provisionamento de pipelines.

Para registrar um repositório de provisionamento de pipeline usando o console

1. Crie um repositório de provisionamento de CI/CD pipeline se você ainda não tiver criado um. Para obter mais informações sobre pipelines no provisionamento autogerenciado, consulte the section called “Provisionamento autogerenciado”.
2. No painel de navegação, selecione Configurações e em seguida Configurações de conta.
3. Na página CI/CD Configurações da conta, escolha Configurar.
4. Na página Definir configurações da conta, na seção Repositório do pipeline de CI/CD:

   a. Selecione Novo repositório e, em seguida, escolha um dos provedores de repositório.

   b. Para CodeStar conexão, escolha uma de suas conexões.

> **Note**
>
> Se você ainda não tiver uma conexão com a conta do provedor de repositório relevante, escolha Adicionar uma nova CodeStar conexão, conclua o processo de criação da conexão e escolha o botão Atualizar ao lado do menu de CodeStarconexão. Agora você deve ser capaz de escolher sua nova conexão no menu.

c. Em Nome do repositório, escolha seu repositório de provisionamento de pipeline. O menu suspenso mostra a lista de repositórios na conta do provedor.

d. Em Nome da ramificação, escolha uma das ramificações do repositório.

5. Escolha Salvar alterações.

   Seu repositório de pipeline é exibido na página de configurações da conta.

Configurando o AWS CLI

Para usar o AWS CLI para fazer chamadas de AWS Proton API, verifique se você instalou a versão mais recente do AWS CLI. Para obter mais informações, consulte Conceitos básicos do AWS CLI no Manual do usuário do AWS Command Line Interface . Em seguida, para começar a usar o AWS CLI with AWS Proton, consultethe section called “Conceitos básicos de CLI”.

Começando com AWS Proton

Antes de começar, prepare-se para usar AWS Proton e verificar se você atendeu aos pré-requisitos de introdução.

Comece AWS Proton escolhendo um ou mais dos seguintes caminhos:

- Seguir um exemplo guiado de console ou fluxo de trabalho da CLI por meio de links de documentação.
- Executar um exemplo guiado de fluxo de trabalho do console.
- Execute um exemplo de AWS CLI fluxo de trabalho guiado.

Tópicos

Pré-requisitos

Antes de começar a usar AWS Proton, verifique se os pré-requisitos a seguir foram atendidos. Para obter mais informações, consulte Configuração.

- É necessário ter uma conta do IAM com permissões de administrador. Para obter mais informações, consulte Configurar com IAM.
- Você tem a função AWS Proton de serviço e a função de serviço de AWS Proton pipeline está anexada à sua conta. Para obter mais informações, consulte Configurando funções AWS Proton de serviço e Funções de serviço para AWS Proton.
- Você tem uma AWS CodeStar conexão. Para obter mais informações, consulte Configurando uma AWS CodeStar conexão.
- Você está familiarizado com a criação de CloudFormation modelos e a parametrização do Jinja. Para obter mais informações, consulte O que é CloudFormation? no Guia do CloudFormation usuário e no site da Jinja.

- Você tem conhecimento prático de serviços de AWS infraestrutura.
- Você está logado no seu. Conta da AWS

Fluxo de trabalho de introdução

Aprenda a criar pacotes de modelos, criar e registrar modelos e criar ambientes e serviços seguindo os exemplos de etapas e links.

Antes de começar, verifique se você criou um perfil de serviço do AWS Proton.

Se seu modelo de serviço incluir um pipeline AWS Proton de serviço, verifique se você criou uma AWS CodeStar conexão e uma função de serviço de AWS Proton pipeline.

Para obter mais informações, consulte The AWS Proton service API Reference.

Exemplo: Fluxo de trabalho de introdução

1. Consulte o diagrama em Como AWS Proton funciona para obter uma visão de alto nível das AWS Proton entradas e saídas.
2. Crie um pacote de ambiente e um pacote de modelo de serviço.
    a. Identifique os parâmetros de entrada.
    b. Crie um arquivo de esquema.
    c. Crie arquivos de infraestrutura como código (IaC).
    d. Para finalizar seu pacote de modelos, crie um arquivo de manifesto e organize seus arquivos IaC, arquivos de manifesto e arquivo de esquema em diretórios.
    e. Torne seu pacote de modelos acessível a. AWS Proton
3. Crie e registre uma versão do modelo de ambiente com AWS Proton.

    Quando você usa o console para criar e registrar um modelo, uma versão do modelo é criada automaticamente.

    Quando você usa o AWS CLI para criar e registrar um modelo:

    a. Criar um modelo de ambiente.
    b. Criar uma versão de modelo de ambiente.

Para obter mais informações, consulte CreateEnvironmentTemplatee CreateEnvironmentTemplateVersionna referência AWS Proton da API.

4. Publique seu modelo de ambiente para disponibilizá-lo para uso.

   Para obter mais informações, consulte UpdateEnvironmentTemplateVersiona referência AWS Proton da API.

5. Para criar um ambiente, selecione uma versão publicada do modelo de ambiente e forneça valores para as entradas necessárias.

   Para obter mais informações, consulte CreateEnvironmenta referência AWS Proton da API.

6. Crie e registre uma versão do modelo de serviço com AWS Proton.

   Quando você usa o console para criar e registrar um modelo, uma versão do modelo é criada automaticamente.

   Quando você usa o AWS CLI para criar e registrar um modelo:

   a. Crie um modelo de serviço.
   b. Criar uma versão de modelo de execução.

   Para obter mais informações, consulte CreateServiceTemplatee CreateServiceTemplateVersionna referência AWS Proton da API.

7. Publique seu modelo de serviço para disponibilizá-lo para uso.

   Para obter mais informações, consulte UpdateServiceTemplateVersiona referência AWS Proton da API.

8. Para criar um serviço, selecione uma versão publicada do modelo de serviço e forneça valores para as entradas necessárias.

   Para obter mais informações, consulte CreateServicea referência AWS Proton da API.

Começando com o Console de gerenciamento da AWS

Comece com AWS Proton

- Criar e visualizar um modelo de ambiente.

- Crie, visualize e publique um modelo de serviço que usa o modelo de ambiente que você acabou de criar.
- Criar um ambiente e um serviço (opcional).
- Excluir o modelo de serviço, o modelo de ambiente, o ambiente e o serviço, se criados.

Etapa 1: abrir o AWS Proton console

- Abra o console do AWS Proton.

Etapa 2: Preparar-se para usar os modelos de exemplo

1. Crie uma conexão Codestar com o Github e nomeie a conexão. my-proton-connection
2. Navegue até https://github.com/aws-samples/aws-proton-cloudformation-sample-templates
3. Crie uma bifurcação do repositório na sua conta do Github.

Etapa 3: Criar um modelo de ambiente

No painel de navegação, escolha Modelos de ambiente.

1. Na página Modelos de ambiente, escolha Criar modelo de ambiente.
2. Na página Criar modelo de ambiente, na seção Opções de modelo, escolha Criar um modelo para provisionar novos ambientes.
3. Na seção Fonte do pacote de modelos, escolha Sincronizar um pacote de modelos do Git.
4. Na seção Repositório de definição de modelo, selecione Escolher um repositório Git vinculado.
5. Selecione na my-proton-connectionlista de repositórios.
6. Selecione principal na Lista de ramificações.
7. Na seção de Detalhes do modelo de ambiente Proton.
    a. Insira o nome do modelo como **`fargate-env`**.
    b. Insira o nome de exibição do modelo de ambiente como **`My Fargate Environment`**.
    c. (Opcional) Insira uma descrição do modelo de ambiente.
8. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.

9. Escolha Criar modelo de ambiente.

   Agora você está em uma nova página que exibe o status e os detalhes do seu novo modelo de ambiente. Esses detalhes incluem uma lista de AWS tags gerenciadas pelo cliente. AWS Proton gera automaticamente tags AWS gerenciadas para você quando você cria AWS Proton recursos. Para obter mais informações, consulte AWS Proton recursos e marcação.

10. O status de um novo status de modelo de ambiente começa no estado Rascunho. Você e outras pessoas com permissões de `proton:CreateEnvironment` podem visualizá-lo e acessá-lo. Siga a próxima etapa para disponibilizar o modelo para outras pessoas.
11. Na seção Versões do modelo, escolha o botão de rádio à esquerda da versão secundária do modelo que você acabou de criar (1.0). Como alternativa, você pode escolher Publicar no banner do alerta de informações e pular a próxima etapa.
12. Na seção Versões do modelo, escolha Publicar.
13. O status do modelo muda para Publicado. Por ser a versão mais recente do modelo, é a versão recomendada.
14. No painel de navegação, selecione Modelos de ambiente.

    Uma nova página exibe uma lista dos modelos do seu ambiente junto com os detalhes do modelo.

Etapa 4: Criar um modelo de serviço

Crie um modelo de serviço.

1. No painel de navegação, escolha Modelos de serviço.
2. Na página Modelos de serviço, escolha Criar modelo de serviço.
3. Na página Criar modelo de serviço, na seção Fonte do pacote de modelos, escolha Sincronizar um pacote de modelos do Git.
4. Na seção Modelo, selecione Escolher um repositório Git vinculado.
5. Selecione na my-proton-connectionlista de repositórios.
6. Selecione principal na Lista de ramificações.
7. Na seção Detalhes do modelo de serviço Proton.

   a. Insira o nome do modelo de serviço como **`backend-fargate-svc`**.

   b. Insira o nome de exibição do modelo de serviço como **`My Fargate Service`**.

c. (Opcional) Insira uma Descrição para o modelo de serviço.

8. Na seção Modelos de ambiente compatíveis.

   - Marque a caixa à esquerda do modelo de ambiente My Fargate Environment para selecionar o modelo de ambiente compatível para o novo modelo de serviço.

9. Para Configurações de criptografia, mantenha as configurações padrão.
10. Na seção Definição do pipeline.

    - Mantenha o botão Este modelo inclui um CI/CD pipeline selecionado.

11. Selecione Criar modelo de serviço.

    Agora você está em uma nova página que exibe o status e os detalhes do seu novo modelo de serviço, incluindo uma lista de AWS etiquetas gerenciadas pelo cliente.

12. O status de um novo status de modelo de serviço começa no estado Rascunho. Somente administradores podem visualizá-lo e acessá-lo. Para disponibilizar o modelo de serviço para uso por desenvolvedores, siga a próxima etapa.
13. Na seção Versões do modelo, escolha o botão de rádio à esquerda da versão secundária do modelo que você acabou de criar (1.0). Como alternativa, você pode escolher Publicar no banner do alerta de informações e pular a próxima etapa.
14. Na seção Versões do modelo, escolha Publicar.
15. O status do modelo muda para Publicado.

    A primeira versão secundária do seu modelo de serviço foi publicada e está disponível para uso por desenvolvedores. Por ser a versão mais recente do modelo, é a versão recomendada.

16. No painel de navegação, escolha Modelos de serviço.

    Para visualizar o status de sua AMI enquanto ela estiver sendo criada, escolha AMIs.

Etapa 5: Criar um ambiente

No painel de navegação, escolha Ambientes.

1. Selecione Criar ambiente.
2. Na página Escolha um modelo de ambiente, selecione o modelo que você acabou de criar. É chamado de My Fargate Environment. Depois, selecione Configure.

3. Na página Configurar ambiente, na seção Provisionamento, escolha Provisionar por meio do AWS Proton.
4. Na seção Conta de implantação, selecione Esta Conta da AWS.
5. Na seção Configurações do ambiente, insira o nome do ambiente como **`my-fargate-environment`**.
6. Na seção Funções de ambiente, selecione Nova função de serviço ou, se você já tiver criado uma função de AWS Proton serviço, selecione Função de serviço existente.

   a. Selecione Novo perfil de serviço para criar um novo perfil.

      i. Insira o Nome do perfil de ambiente como **`MyProtonServiceRole`**.

      ii. Marque a caixa de seleção para concordar em criar uma função AWS Proton de serviço com privilégios administrativos para sua conta.

   b. Selecione Perfil de serviço existente para usar um perfil existente.

      - Selecione seu perfil no campo suspenso Nome do perfil de ambiente.
7. Escolha Próximo.
8. Na página Definir configurações personalizadas, use os padrões.
9. Escolha Avançar e revise suas entradas.
10. Escolha Criar.

    Visualize os detalhes e o status do ambiente, bem como as tags AWS gerenciadas e as tags gerenciadas pelo cliente para seu ambiente.
11. No painel de navegação, escolha Ambientes.

    Uma nova página exibe uma lista de seus ambientes junto com o status e outros detalhes do ambiente.

Etapa 6: opcional - Criar um serviço e implantar uma aplicação

1. Abra o console do AWS Proton.
2. No painel de navegação, escolha Serviços.
3. Na página Serviços, escolha Criar serviço.
4. Na página Escolher um modelo de serviço, selecione o modelo My Fargate Service escolhendo o botão de opção no canto superior direito do modelo de cartão.

5. Escolha Configurar no canto inferior direito da página.
6. Na página Configurar serviço, na seção Configurações do serviço, insira o nome do serviço do **`my-service`**.
7. (Opcional) Insira uma descrição para o serviço.
8. Na seção Configurações do repositório de serviços:
    a. Para CodeStar conexão, escolha sua conexão na lista.
    b. Para Nome do repositório, escolha o nome do seu repositório de códigos-fonte da lista.
    c. Em Nome da ramificação, escolha o nome da ramificação do repositório de código-fonte na lista.
9. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente. Escolha Próximo.
10. Na página Definir configurações personalizadas, na seção Instâncias de serviço, na seção Nova instância, siga as próximas etapas para fornecer valores personalizados para os parâmetros da sua instância de serviço.
    a. Insira o nome da instância do **`my-app-service`**.
    b. Escolha o ambiente **`my-fargate-environment`** para sua instância de serviço.
    c. Mantenha os padrões para os parâmetros de instância restantes.
    d. Mantenha os padrões para as Entradas do pipeline.
    e. Escolha Avançar e revise suas entradas.
    f. Escolha Criar e visualize o status e os detalhes do seu serviço.
11. Na página de detalhes do serviço, visualize o status da sua instância de serviço e do pipeline escolhendo as guias Visão geral e Pipeline. Nessas páginas, você também pode visualizar AWS as tags gerenciadas pelo cliente. AWS Proton cria automaticamente tags AWS gerenciadas para você. Escolha Gerenciar tags para criar e modificar tags gerenciadas pelo cliente. Para obter mais informações sobre marcação, consulte AWS Proton recursos e marcação.
12. Depois que o serviço estiver ativo, na guia Visão geral, na seção Instâncias de serviço, escolha o nome da sua instância de serviço, my-app-service.

    Agora, você está na página de detalhes de instância de serviço.
13. Para visualizar seu aplicativo, na seção Saídas, copie o ServiceEndpointlink para o seu navegador.

    Você vê um AWS Proton gráfico na página da web.

14. Após o serviço ser criado, no painel de navegação Serviços para exibir uma lista de seus serviços.

Etapa 7: Limpar

1. Abra o console do AWS Proton.
2. Excluir um serviço (se você criou um)
    a. No painel de navegação, escolha Serviços.
    b. Na página Serviços, escolha o nome do serviço my-service.

       Agora você está na página de detalhes do serviço my-service.
    c. No canto superior direito da página, escolha Ações, e então Excluir.
    d. Um modal pede que você confirme a ação de exclusão.
    e. Siga as instruções e escolha Sim, excluir.
3. Exclua um ambiente
    a. No painel de navegação, escolha Ambientes.
    b. Na página Ambientes, selecione o botão de rádio à esquerda do ambiente você acaba de criar.
    c. Escolha Ações e Excluir.
    d. Um modal pede que você confirme a ação de exclusão.
    e. Siga as instruções e escolha Sim, excluir.
4. Excluir um modelo de serviço
    a. No painel de navegação, escolha Modelos de serviço.
    b. Na página de modelos de serviço, selecione o botão de opção à esquerda do modelo de serviço my-svc-template.
    c. Escolha Ações e Excluir.
    d. Um modal pede que você confirme a ação de exclusão.
    e. Siga as instruções e escolha Sim, excluir. Isso exclui o modelo de serviço e todas as suas versões.
5. Excluir um modelo de ambiente
    a. No painel de navegação, escolha Modelos de ambiente.

b. Na página de modelos de ambiente, selecione o botão de rádio à esquerda de my-env-template.

c. Escolha Ações e Excluir.

d. Um modal pede que você confirme a ação de exclusão.

e. Siga as instruções e escolha Sim, excluir. Isso exclui o modelo de ambiente e todas as suas versões.

6. Excluir sua conexão Codestar

Começando com o AWS CLI

Para começar a AWS Proton usar o AWS CLI, siga este tutorial. O tutorial demonstra um serviço público de balanceamento de carga baseado em AWS Proton . AWS Fargate O tutorial também fornece um CI/CD pipeline que implanta um site estático com uma imagem exibida.

Antes de começar, verifique se tudo está configurado corretamente. Para obter detalhes, consulte the section called “Pré-requisitos”.

Etapa 1: Registrar um modelo de ambiente

Nesta etapa, como administrador, você registra um modelo de ambiente de exemplo, que contém um cluster do Amazon Elastic Container Service (Amazon ECS) e uma Amazon Virtual Private Cloud (Amazon VPC) com duas sub-redes. public/private

Para registrar um modelo de ambiente

1. Bifurque o repositório AWS Proton Sample CloudFormation Templates em sua GitHub conta ou organização. Esse repositório inclui os modelos de ambiente e serviço que usamos neste tutorial.

   Em seguida, registre seu repositório bifurcado com. AWS Proton Para obter mais informações, consulte the section called “Criar um link de repositório”.

2. Criar um modelo de ambiente.

   O recurso do modelo de ambiente rastreia as versões do modelo de ambiente.

```
$ aws proton create-environment-template \
  --name "fargate-env" \
```

```
  --display-name "Public VPC Fargate" \
  --description "VPC with public access and ECS cluster"
```

3. Crie uma configuração de sincronização de modelos.

   AWS Proton configura uma relação de sincronização entre seu repositório e seu modelo de ambiente. Em seguida, ele cria a versão 1.0 do modelo em status de DRAFT.

```
$ aws proton create-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "environment-templates/fargate-env"
```

4. Aguarde até que a versão do modelo de ambiente seja registrada com sucesso.

   Quando esse comando retorna com um status de saída de `0`, o registro da versão é concluído. Isso é útil em scripts para garantir que você possa executar o comando com êxito na próxima etapa.

```
$ aws proton wait environment-template-version-registered \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0"
```

5. Publique a versão do modelo de ambiente para disponibilizá-la para criação de ambiente.

```
$ aws proton update-environment-template-version \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

Etapa 2: Registrar um modelo de serviço

Nesta etapa, como administrador, você registra um exemplo de modelo de serviço, que contém todos os recursos necessários para provisionar um serviço Amazon ECS Fargate por trás de um balanceador de carga e CI/CD um pipeline que usa. AWS CodePipeline

Para registrar um modelo de serviço

1. Crie um modelo de serviço.

   O recurso do modelo de serviço rastreia as versões do modelo de serviço.

   ```
   $ aws proton create-service-template \
     --name "load-balanced-fargate-svc" \
     --display-name "Load balanced Fargate service" \
     --description "Fargate service with an application load balancer"
   ```

2. Crie uma configuração de sincronização de modelos.

   AWS Proton configura uma relação de sincronização entre seu repositório e seu modelo de serviço. Em seguida, ele cria a versão 1.0 do modelo em status de DRAFT.

   ```
   $ aws proton create-template-sync-config \
     --template-name "load-balanced-fargate-svc" \
     --template-type "SERVICE" \
     --repository-name "your-forked-repo" \
     --repository-provider "GITHUB" \
     --branch "your-branch" \
     --subdirectory "service-templates/load-balanced-fargate-svc"
   ```

3. Aguarde até que a versão do modelo de serviço seja registrada com sucesso.

   Quando esse comando retorna com um status de saída de `0`, o registro da versão é concluído. Isso é útil em scripts para garantir que você possa executar o comando com êxito na próxima etapa.

   ```
   $ aws proton wait service-template-version-registered \
     --template-name "load-balanced-fargate-svc" \
     --major-version "1" \
     --minor-version "0"
   ```

4. Publique a versão do modelo de serviço para disponibilizá-la para criação de serviços.

   ```
   $ aws proton update-service-template-version \
     --template-name "load-balanced-fargate-svc" \
     --major-version "1" \
     --minor-version "0" \
     --status "PUBLISHED"
   ```

Etapa 3: implantar um ambiente

Nesta etapa, como administrador, você instancia um AWS Proton ambiente a partir do modelo de ambiente.

Para implantar um ambiente

1. Obtenha um exemplo de arquivo de especificação para o modelo de ambiente que você registrou.

   Você pode baixar os `environment-templates/fargate-env/spec/spec.yaml` do arquivo do repositório de exemplos de modelos. Como alternativa, você pode buscar o repositório inteiro localmente e executar o comando create-environment a partir do diretório `environment-templates/fargate-env`.

2. Criar um ambiente.

   AWS Proton lê os valores de entrada de sua especificação de ambiente, os combina com seu modelo de ambiente e provisiona recursos ambientais em sua AWS conta usando sua função AWS Proton de serviço.

   ```
   $ aws proton create-environment \
     --name "fargate-env-prod" \
     --template-name "fargate-env" \
     --template-major-version 1 \
     --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
    \
     --spec "file://spec/spec.yaml"
   ```

3. Aguarde até que o ambiente seja implantado com sucesso.

   ```
   $ aws proton wait environment-deployed --name "fargate-env-prod"
   ```

Etapa 4: Implantar um serviço [desenvolvedor de aplicações]

Nas etapas anteriores, um administrador registrou e publicou um modelo de serviço e implantou um ambiente. Como desenvolvedor de aplicativos, agora você pode criar um AWS Proton serviço e implantá-lo no AWS Proton ambiente

Para implantar um serviço.

1. Obtenha um exemplo de arquivo de especificação para o modelo de serviço que o administrador registrou.

   Você pode baixar os `service-templates/load-balanced-fargate-svc/spec/spec.yaml` do arquivo do repositório de exemplos de modelos. Como alternativa, você pode buscar o repositório inteiro localmente e executar o comando create-service a partir do diretório `service-templates/load-balanced-fargate-svc`.
2. Bifurque o repositório AWS Proton Sample Services em sua GitHub conta ou organização. Esse repositório inclui o código-fonte da aplicação que usamos neste tutorial.
3. Crie um serviço.

   AWS Proton lê os valores de entrada de sua especificação de serviço, os combina com seu modelo de serviço e provisiona recursos de serviço em sua AWS conta no ambiente especificado na especificação. Um AWS CodePipeline pipeline implanta o código do seu aplicativo a partir do repositório que você especifica no comando.

   ```
   $ aws proton create-service \
     --name "static-website" \
     --repository-connection-arn \
       "arn:aws:codestar-connections:us-east-1:123456789012:connection/your-codestar-connection-id" \
     --repository-id "your-GitHub-account/aws-proton-sample-services" \
     --branch-name "main" \
     --template-major-version 1 \
     --template-name "load-balanced-fargate-svc" \
     --spec "file://spec/spec.yaml"
   ```
4. Aguarde até que o serviço seja implantado com sucesso.

   ```
   $ aws proton wait service-created --name "static-website"
   ```
5. Recupere os resultados e visualize seu novo site.

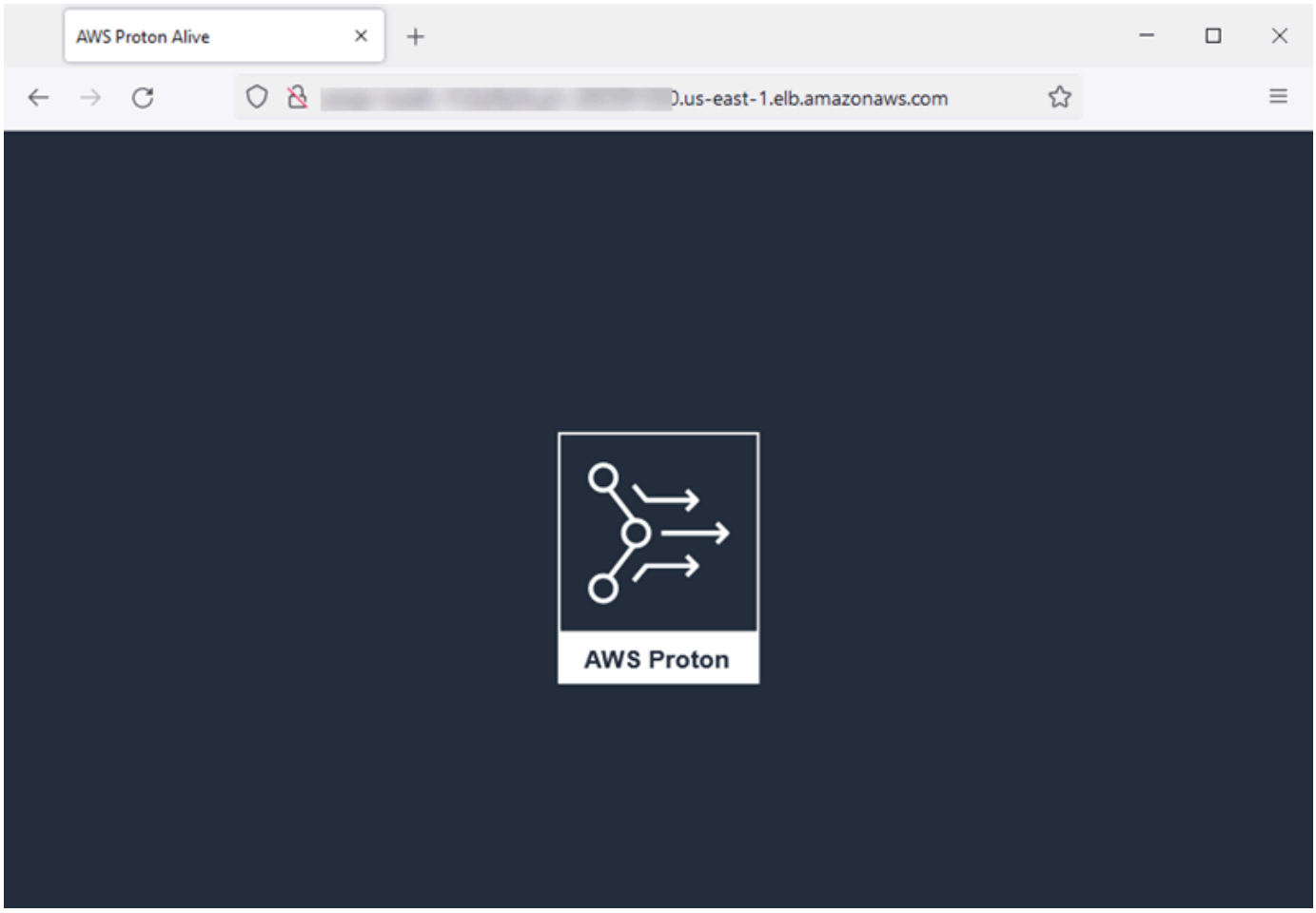
   Execute este comando: .

   ```
   $ aws proton list-service-instance-outputs \
     --service-name "static-website" \
     --service-instance-name load-balanced-fargate-svc-prod
   ```

A saída do comando deve ser semelhante ao mostrado a seguir:

```
{
    "outputs": [
        {
            "key": "ServiceURL",
            "valueString": "http://your-service-endpoint.us-
east-1.elb.amazonaws.com"
        }
    ]
}
```

O valor da saída da instância de `ServiceURL` é o endpoint do seu novo site de serviço. Use seu navegador para navegar até ele. Você deve ver o seguinte gráfico em uma página estática:



Etapa 5: Limpeza (opcional)

Nesta etapa, quando terminar de explorar os AWS recursos que você criou como parte deste tutorial e para economizar nos custos associados a esses recursos, você os exclui.

Para excluir recursos do tutorial

1. Para excluir o serviço, execute o seguinte comando:

```
$ aws proton delete-service --name "static-website"
```

2. Para excluir o ambiente, execute o seguinte comando:

```
$ aws proton delete-environment --name "fargate-env-prod"
```

3. Para excluir o modelo de serviço, execute os seguintes comandos:

```
$ aws proton delete-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE"
$ aws proton delete-service-template --name "load-balanced-fargate-svc"
```

4. Para excluir o modelo de ambiente, execute os seguintes comandos:

```
$ aws proton delete-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT"
$ aws proton delete-environment-template --name "fargate-env"
```

A biblioteca AWS Proton de modelos

A AWS Proton equipe mantém uma biblioteca de exemplos de modelos no GitHub. A biblioteca inclui exemplos de arquivos de infraestrutura como código (IaC) para muitos cenários comuns de ambiente e infraestrutura de aplicações.

A biblioteca de modelos é armazenada em dois GitHub repositórios:

- aws-proton-cloudformation-sample-templates — Exemplos de pacotes de modelos que usam AWS CloudFormationo Jinja como sua linguagem IaC. Você pode usar esses exemplos para ambientes de Provisionamento gerenciado pela AWS.

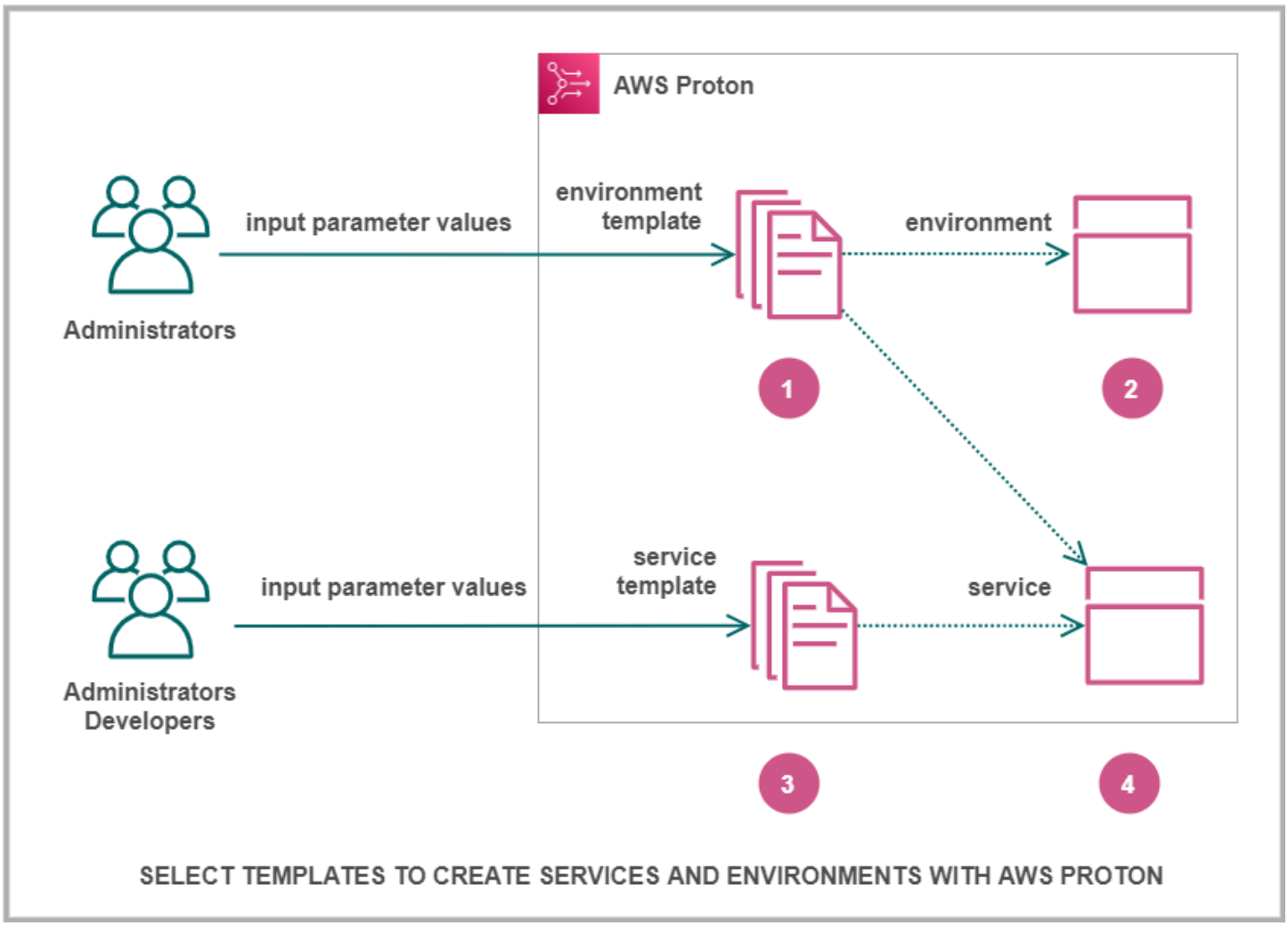
- aws-proton-terraform-sample-templates — Exemplos de pacotes de modelos que usam o Terraform como sua linguagem IaC. Você pode usar esses exemplos para ambientes de Provisionamento autogerenciado.

Cada um desses repositórios tem um arquivo `README` com informações completas sobre o conteúdo e a estrutura do repositório. Cada exemplo tem informações sobre o caso de uso que o modelo abrange, a arquitetura do exemplo e os parâmetros de entrada que o modelo usa.

Você pode usar os modelos dessa biblioteca diretamente bifurcando um dos repositórios da biblioteca em sua GitHub conta. Como alternativa, use esses exemplos como ponto de partida para desenvolver seus modelos de ambiente e serviço.

Como AWS Proton funciona

Com AWS Proton, você provisiona ambientes e, em seguida, serviços executados nesses ambientes. Ambientes e serviços são baseados em modelos de ambiente e serviço, respectivamente, que você escolhe em sua biblioteca de modelos AWS Proton versionados.



1 Quand você, como administrador, seleciona um modelo de ambiente com AWS Proton, você fornece valores para os parâmetros de entrada necessários.

2 AWS Proton usa o modelo do ambiente e os valores dos parâmetros para provisionar seu ambiente.

3

Quand

você, como desenvolvedor ou administrador, seleciona um modelo de serviço com AWS Proton, você fornece valores para os parâmetros de entrada necessários. Você também seleciona um ambiente para implantar sua aplicação ou serviço.

4

AWS Proton usa o modelo de serviço e os valores dos parâmetros do seu serviço e do ambiente selecionado para provisionar seu serviço.

Você fornece valores para os parâmetros de entrada para personalizar seu modelo para reutilização e vários casos de uso, aplicações ou serviços.

Para fazer isso funcionar, você cria pacotes de modelos de ambiente ou serviço e os carrega em modelos de ambiente ou serviço registrados, respectivamente.

Os pacotes de modelos contêm tudo o que é AWS Proton necessário para provisionar ambientes ou serviços.

Quando você cria um modelo de ambiente ou de serviço, você envia um pacote de modelos que contém os arquivos parametrizados de infraestrutura como código (IaC) que o AWS Proton usa para provisionar ambientes ou serviços.

Ao selecionar um modelo de ambiente ou serviço para criar ou atualizar um ambiente ou serviço, você fornece valores para os parâmetros do arquivo IaC do pacote de modelos.

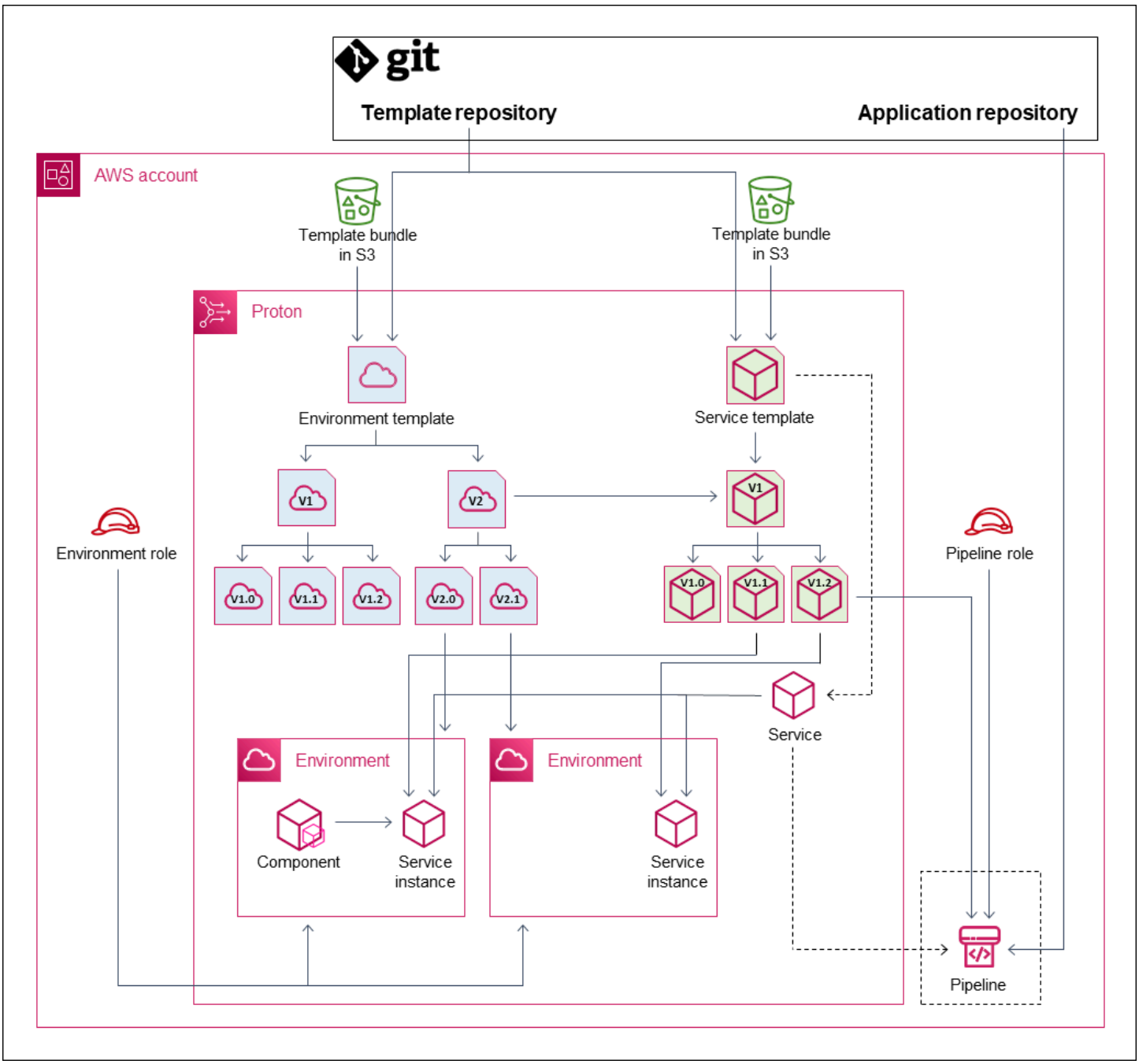
Tópicos

- AWS Proton objetos
- Como AWS Proton provisiona a infraestrutura
- AWS Proton terminologia

AWS Proton objetos

O diagrama a seguir mostra os AWS Proton objetos principais e sua relação com outros objetos AWS e objetos de terceiros. As setas representam a direção do fluxo de dados (a direção inversa da dependência).

Seguimos o diagrama com breves descrições e links de referência para esses AWS Proton objetos.



- Modelo de ambiente: Uma coleção de versões de modelo de ambiente que podem ser usadas para criar ambientes do AWS Proton .

  Para obter mais informações, consulte Criação e pacotes de modelos e Modelos.
- Versão do modelo de ambiente: Uma versão específica de um modelo de ambiente. Usa um pacote de modelos como entrada, seja de um bucket do S3 ou de um repositório Git. O pacote especifica a Infraestrutura como Código (IaC) e os parâmetros de entrada relacionados para um ambiente. AWS Proton

Para ter mais informações, consulte the section called “Versões”, the section called “Publicar” e the section called “Configurações de sincronização de modelo”.

- Ambiente — O conjunto de recursos de AWS infraestrutura compartilhada e políticas de acesso em que AWS Proton os serviços são implantados. AWS os recursos são provisionados usando uma versão de modelo de ambiente invocada com valores de parâmetros específicos. As políticas de acesso são fornecidas em um perfil de serviço.

  Para obter mais informações, consulte Ambientes do.

- Modelo de serviço: Uma coleção de versões de modelo de serviço que podem ser usadas para criar serviços do AWS Proton .

  Para obter mais informações, consulte Criação e pacotes de modelos e Modelos.

- Versão de modelo de serviço: Uma versão específica de um modelo de serviço. Usa um pacote de modelos como entrada, seja de um bucket do S3 ou de um repositório Git. O pacote especifica a Infraestrutura como Código (IaC) e os parâmetros de entrada relacionados para um serviço. AWS Proton

  Uma versão do modelo de serviço também especifica essas restrições nas instâncias de serviço com base na versão:

  - Modelos de ambiente compatíveis: As instâncias só podem ser executadas em ambientes baseados nesses modelos de ambiente compatíveis.
  - Fontes de componentes compatíveis: Os tipos de componentes que os desenvolvedores podem associar às instâncias.

  Para ter mais informações, consulte the section called “Versões”, the section called “Publicar” e the section called “Configurações de sincronização de modelo”.

- Serviço — Uma coleção de instâncias de serviço que executam um aplicativo usando recursos especificados em um modelo de serviço e, opcionalmente, um CI/CD pipeline que implanta o código do aplicativo nessas instâncias.

  No diagrama, a linha tracejada do modelo de serviço significa que o serviço passa o modelo para as instâncias de serviço e para o pipeline.

  Para obter mais informações, consulte Serviços.

- Instância de serviço — O conjunto de recursos de AWS infraestrutura que executam um aplicativo em um AWS Proton ambiente específico. AWS os recursos são provisionados usando uma versão do modelo de serviço invocada com valores de parâmetros específicos.

  Para obter mais informações, consulte Serviços e the section called “Atualizar instância”.

- Pipeline — Um CI/CD pipeline opcional que implanta um aplicativo nas instâncias de um serviço, com políticas de acesso para provisionar esse pipeline. As políticas de acesso são fornecidas em um perfil de serviço. Um serviço nem sempre tem um AWS Proton pipeline associado. Você pode optar por gerenciar suas implantações de código de aplicativo fora dele. AWS Proton

  No diagrama, a linha tracejada de Service e a caixa tracejada ao redor do Pipeline significam que, se você optar por gerenciar suas CI/CD implantações sozinho, o AWS Proton pipeline pode não ser criado e seu próprio pipeline pode não estar em sua conta. AWS

  Para obter mais informações, consulte Serviços e the section called “Atualizar pipeline”.

- Componente: Uma extensão definida pelo desenvolvedor para uma instância de serviço. Especifica recursos adicionais de AWS infraestrutura que um aplicativo específico pode precisar, além dos recursos fornecidos pelo ambiente e pela instância do serviço. As equipes de plataforma controlam a infraestrutura que um componente pode provisionar ao atribuir um perfil de componente ao ambiente.

  Para obter mais informações, consulte Componentes.

Como AWS Proton provisiona a infraestrutura

AWS Proton pode provisionar a infraestrutura de várias maneiras:

- AWS-provisionamento gerenciado — AWS Proton chama o mecanismo de provisionamento em seu nome. Esse método oferece suporte somente a pacotes de modelos do AWS CloudFormation . Para obter mais informações, consulte the section called “CloudFormation Arquivos IaC”.
- CodeBuild provisionamento — AWS Proton usa AWS CodeBuild para executar comandos de shell fornecidos por você. Seus comandos podem ler as entradas que AWS Proton fornecem e são responsáveis por provisionar ou desprovisionar a infraestrutura e gerar valores de saída. Um pacote de modelos para esse método inclui seus comandos em um arquivo de manifesto e quaisquer programas, scripts ou outros arquivos que esses comandos possam precisar.

Como exemplo do uso do CodeBuild provisionamento, você pode incluir um código que usa o AWS Cloud Development Kit (AWS CDK) para provisionar AWS recursos e um manifesto que instala o CDK e executa seu código CDK.

Para obter mais informações, consulte the section called “CodeBuild pacote”.

> **Note**
>
> Você pode usar o CodeBuild provisionamento com ambientes e serviços. No momento, você não pode provisionar componentes dessa forma.

- Provisionamento autogerenciado — AWS Proton emite uma pull request (PR) para um repositório que você fornece, onde seu próprio sistema de implantação de infraestrutura executa o processo de provisionamento. Esse método oferece suporte somente a pacotes de modelos do Terraform. Para obter mais informações, consulte the section called “Arquivos Terraform IaC”.

AWS Proton determina e define o método de provisionamento para cada ambiente e serviço separadamente. Quando você cria ou atualiza um ambiente ou serviço, o AWS Proton examina o pacote de modelos que você fornece e determina o método de provisionamento indicado pelo pacote de modelos. No nível do ambiente, você fornece os parâmetros que o ambiente e seus possíveis serviços podem precisar para seus métodos de provisionamento — funções AWS Identity and Access Management (IAM), uma conexão de conta de ambiente ou um repositório de infraestrutura.

Os desenvolvedores que costumam AWS Proton provisionar um serviço têm a mesma experiência, independentemente do método de provisionamento. Os desenvolvedores não precisam conhecer o método de provisionamento e não precisam alterar nada no processo de provisionamento de serviços. O modelo de serviço define o método de provisionamento, e cada ambiente no qual um desenvolvedor implanta o serviço fornece os parâmetros necessários para o provisionamento da instância de serviço.

O diagrama a seguir resume algumas características principais dos diferentes métodos de provisionamento. As seções que seguem a tabela fornecem detalhes sobre cada método.

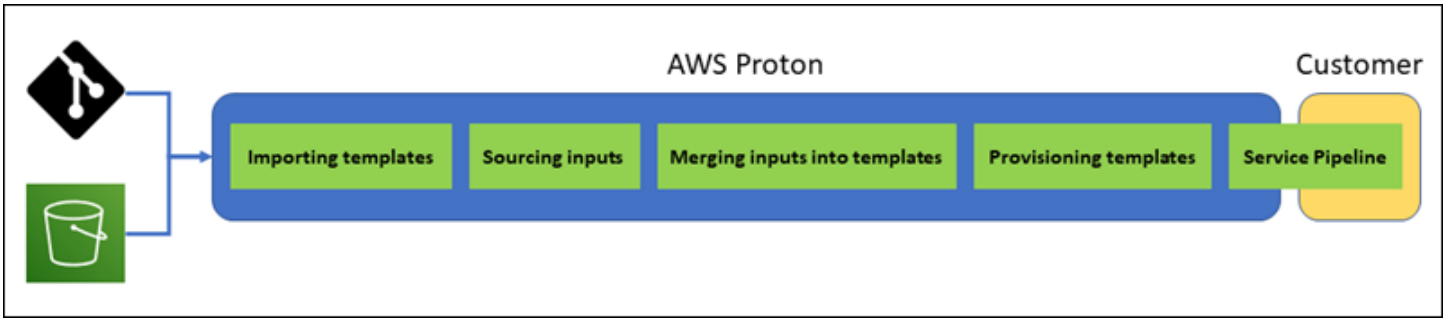
Método de provisionamento	Modelos	Provisionado por	Status rastreado por
Gerenciado pela AWS	manifesto, esquema, arquivo IaC () CloudFormation	AWS Proton (por meio de CloudFormation)	AWS Proton (por meio de CloudFormation)
CodeBuild	manifesto (com comandos), esquema, dependências de comando (por exemplo AWS CDK , código)	AWS Proton (por meio de CodeBuild)	AWS Proton (seus comandos retornam o status por meio de CodeBuild)
autogerenciado	manifesto, esquema, arquivos IaC (Terraform)	Seu código (por meio de ações do Git)	Seu código (passado AWS por meio de uma chamada de API)

Como funciona o provisionamento AWS gerenciado

Quando um ambiente ou serviço usa provisionamento AWS gerenciado, a infraestrutura é provisionada da seguinte forma:

1. Um AWS Proton cliente (administrador ou desenvolvedor) cria o AWS Proton recurso (um ambiente ou um serviço). O cliente seleciona um modelo para o recurso e fornece os parâmetros necessários. Para obter mais informações, consulte a seção the section called “Considerações sobre o provisionamento AWS gerenciado” a seguir.
2. AWS Proton renderiza um CloudFormation modelo completo para provisionar o recurso.
3. AWS Proton chamadas CloudFormation para iniciar o provisionamento usando o modelo renderizado.
4. AWS Proton monitora continuamente a CloudFormation implantação.
5. Quando o provisionamento é concluído, AWS Proton reporta os erros em caso de falha e captura as saídas do provisionamento, como o Amazon VPC ID, em caso de sucesso.

O diagrama a seguir mostra AWS Proton que ele executa a maioria dessas etapas diretamente.



Considerações sobre o provisionamento AWS gerenciado

- Função de provisionamento de infraestrutura — Quando um ambiente ou qualquer uma das instâncias de serviço em execução nele pode usar o provisionamento AWS gerenciado, um administrador precisa configurar uma função do IAM (diretamente ou como parte de uma AWS Proton conexão de conta do ambiente). AWS Proton usa essa função para provisionar a infraestrutura desses recursos de provisionamento AWS gerenciados. A função deve ter permissões de uso CloudFormation para criar todos os recursos que os modelos desses recursos incluem.

  Para obter mais informações, consulte the section called “Perfis do IAM” e the section called “Exemplos de política de perfil de serviço”.

- Provisionamento de serviços — Quando um desenvolvedor implanta uma instância de serviço que usa provisionamento AWS gerenciado no ambiente, AWS Proton usa a função fornecida a esse ambiente para provisionar a infraestrutura para a instância de serviço. Os desenvolvedores não veem esse perfil e não podem alterá-lo.

- Serviço com pipeline — Um modelo de serviço que usa provisionamento AWS gerenciado pode incluir uma definição de pipeline escrita no esquema YAML. CloudFormation AWS Proton também cria o pipeline chamando CloudFormation. A função AWS Proton usada para criar um pipeline é separada da função de cada ambiente individual. Essa função é fornecida AWS Proton separadamente, apenas uma vez no nível da AWS conta, e é usada para provisionar e gerenciar todos os pipelines AWS gerenciados. Esse perfil deve ter permissões para criar pipelines e outros recursos de que seus pipelines precisam.

  Os procedimentos a seguir mostram como fornecer o perfil de pipeline para o AWS Proton.

AWS Proton console

Para fornecer o perfil do pipeline

1. No console do AWS Proton, no painel de navegação, escolha Configurações > Configurações da conta e, em seguida, escolha Configurar.
2. Use a seção Função AWS gerenciada por pipeline para configurar uma função de pipeline nova ou existente para provisionamento AWS gerenciado.

AWS Proton API

Para fornecer o perfil do pipeline

1. Use a ação UpdateAccountSettingsda API.
2. Forneça o nome do recurso da Amazon (ARN) do seu perfil de serviço de pipeline no parâmetro do `pipelineServiceRoleArn`.

AWS CLI

Para fornecer o perfil do pipeline

Execute o seguinte comando:

```
$ aws proton update-account-settings \
    --pipeline-service-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

Como funciona o CodeBuild provisionamento

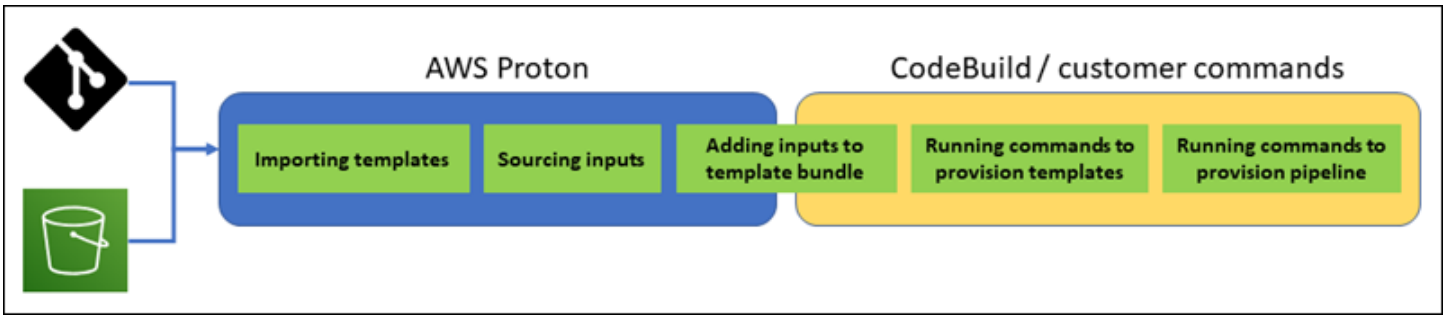
Quando um ambiente ou serviço usa CodeBuild provisionamento, a infraestrutura é provisionada da seguinte forma:

1. Um AWS Proton cliente (administrador ou desenvolvedor) cria o AWS Proton recurso (um ambiente ou um serviço). O cliente seleciona um modelo para o recurso e fornece os parâmetros necessários. Para obter mais informações, consulte a seção the section called “Considerações sobre o CodeBuild provisionamento” a seguir.

2. AWS Proton renderiza um arquivo de entrada com valores de parâmetros de entrada para provisionar o recurso.
3. AWS Proton chamadas CodeBuild para começar um trabalho. O CodeBuild trabalho executa os comandos do shell do cliente especificados no modelo. Esses comandos provisionam a infraestrutura desejada e, opcionalmente, leem os valores de entrada.
4. Quando o provisionamento é concluído, o comando final do cliente retorna o status do provisionamento CodeBuild e chama a ação da NotifyResourceDeploymentStatusChange AWS Proton API para fornecer resultados, como o Amazon VPC ID, se houver.

> **Important**
>
> Certifique-se de que seus comandos retornem corretamente o status de provisionamento CodeBuild e forneçam as saídas. Caso contrário, AWS Proton não conseguirão rastrear adequadamente o status do provisionamento e não poderão fornecer resultados corretos para as instâncias de serviço.

O diagrama a seguir ilustra as etapas que são AWS Proton executadas e as etapas que seus comandos executam em uma CodeBuild tarefa.



Considerações sobre o CodeBuild provisionamento

- Função de provisionamento de infraestrutura — Quando um ambiente ou qualquer uma das instâncias de serviço em execução nele pode usar o provisionamento CodeBuild baseado, um administrador precisa configurar uma função do IAM (diretamente ou como parte de uma conexão de conta do AWS Proton ambiente). AWS Proton usa essa função para provisionar a infraestrutura desses recursos de CodeBuild provisionamento. A função deve ter permissões de uso CodeBuild para criar todos os recursos que seus comandos nos modelos desses recursos fornecem.

Para obter mais informações, consulte the section called “Perfis do IAM” e the section called “Exemplos de política de perfil de serviço”.

- Provisionamento de serviços — Quando um desenvolvedor implanta uma instância de serviço que usa CodeBuild provisionamento no ambiente, AWS Proton usa a função fornecida a esse ambiente para provisionar a infraestrutura para a instância de serviço. Os desenvolvedores não veem esse perfil e não podem alterá-lo.
- Serviço com pipeline — Um modelo de serviço que usa CodeBuild provisionamento pode incluir comandos para provisionar um pipeline. AWS Proton também cria o pipeline chamando CodeBuild. A função AWS Proton usada para criar um pipeline é separada da função de cada ambiente individual. Essa função é fornecida AWS Proton separadamente, apenas uma vez no nível da AWS conta, e é usada para provisionar e gerenciar todos os pipelines CodeBuild baseados. Esse perfil deve ter permissões para criar pipelines e outros recursos de que seus pipelines precisam.

  Os procedimentos a seguir mostram como fornecer o perfil de pipeline para o AWS Proton.

  AWS Proton console

  Para fornecer o perfil do pipeline

  1. No console do AWS Proton, no painel de navegação, escolha Configurações > Configurações da conta e, em seguida, escolha Configurar.
  2. Use a seção Função de provisionamento de pipeline do Codebuild para configurar uma função de pipeline nova ou existente para provisionamento. CodeBuild

  AWS Proton API

  Para fornecer o perfil do pipeline

  1. Use a ação UpdateAccountSettingsda API.
  2. Forneça o nome do recurso da Amazon (ARN) do seu perfil de serviço de pipeline no parâmetro do `pipelineCodebuildRoleArn`.

  AWS CLI

  Para fornecer o perfil do pipeline

  Execute o seguinte comando:

```
$ aws proton update-account-settings \
    --pipeline-codebuild-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

Como funciona o provisionamento autogerenciado

Quando um ambiente é configurado para usar provisionamento autogerenciado, a infraestrutura é provisionada da seguinte forma:

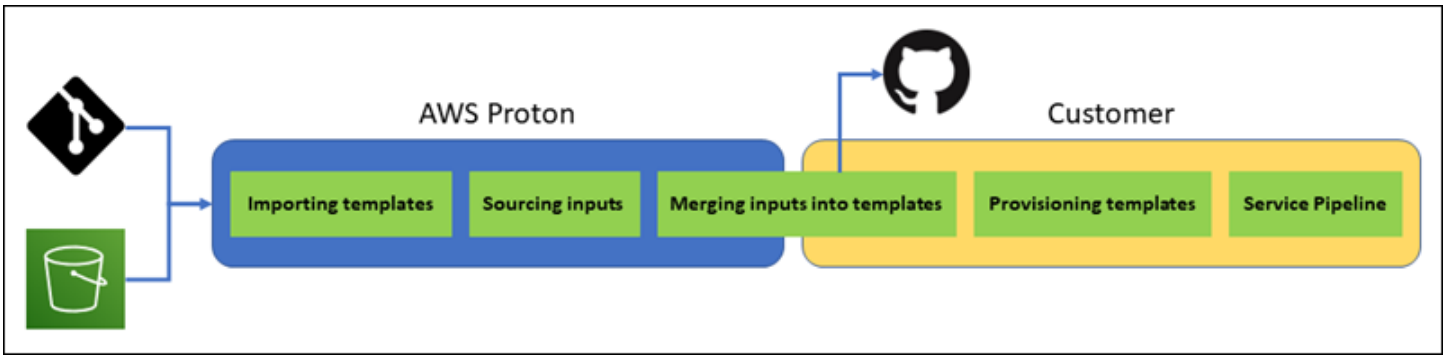
1. Um AWS Proton cliente (administrador ou desenvolvedor) cria o AWS Proton recurso (um ambiente ou um serviço). O cliente seleciona um modelo para o recurso e fornece os parâmetros necessários. Para um ambiente, o cliente também fornece um repositório de infraestrutura vinculado. Para obter mais informações, consulte a seção the section called “Considerações sobre o provisionamento autogerenciado” a seguir.
2. AWS Proton renderiza um modelo completo do Terraform. Ele consiste em um ou mais arquivos do Terraform, potencialmente em várias pastas, e um arquivo de `.tfvars` variáveis. AWS Proton grava os valores dos parâmetros fornecidos na chamada de criação do recurso nesse arquivo de variáveis.
3. AWS Proton envia um PR para o repositório de infraestrutura com o modelo renderizado do Terraform.
4. Quando o cliente (administrador ou desenvolvedor) mescla o PR, a automação do cliente aciona o mecanismo de provisionamento para iniciar o provisionamento da infraestrutura usando o modelo mesclado.

   > **Note**
   >
   > Se o cliente (administrador ou desenvolvedor) fechar o PR, AWS Proton reconhecerá o PR como fechado e marcará a implantação como cancelada.

5. Quando o provisionamento é concluído, a automação do cliente chama a ação da NotifyResourceDeploymentStatusChange AWS Proton API para indicar a conclusão, fornecer o status (sucesso ou falha) e fornecer resultados, como o Amazon VPC ID, se houver.

> **Important**
>
> Certifique-se de que seu código de automação seja retornado AWS Proton com o status e as saídas do provisionamento. Caso contrário, AWS Proton considere o provisionamento pendente por mais tempo do que deveria e continue mostrando o status Em andamento.

O diagrama a seguir ilustra as etapas que são AWS Proton executadas e as etapas que seu próprio sistema de provisionamento executa.



Considerações sobre o provisionamento autogerenciado

- Repositório de infraestrutura — Quando um administrador configura um ambiente para provisionamento autogerenciado, ele precisa fornecer um repositório de infraestrutura vinculado. AWS Proton se submete PRs a esse repositório para provisionar a infraestrutura do ambiente e todas as instâncias de serviço que são implantadas nele. A ação de automação de propriedade do cliente no repositório deve assumir uma função do IAM com permissões para criar todos os recursos que seus modelos de ambiente e serviço incluem e uma identidade que reflita a conta de destino. AWS Para ver um exemplo de GitHub ação que assume uma função, consulte Assumindo uma função na documentação “Configurar AWS credenciais” da ação para GitHub ações.
- Permissões — Seu código de provisionamento precisa ser autenticado com uma conta conforme necessário (por exemplo, autenticar-se em uma AWS conta) e fornecer autorização de provisionamento de recursos (por exemplo, fornecer uma função).
- Provisionamento de serviços — Quando um desenvolvedor implanta uma instância de serviço que usa provisionamento autogerenciado no ambiente, AWS Proton envia um PR ao repositório associado ao ambiente para provisionar a infraestrutura para a instância de serviço. Os desenvolvedores não veem o repositório e não podem alterá-lo.

> **Note**
>
> Os desenvolvedores que criam serviços usam o mesmo processo, independentemente do método de provisionamento, e a diferença é abstraída deles. No entanto, com o provisionamento autogerenciado, os desenvolvedores podem ter uma resposta mais lenta, porque precisam esperar até que alguém (que pode não ser eles mesmos) mescle o PR no repositório de infraestrutura antes que o provisionamento possa começar.

- Serviço com pipeline — Um modelo de serviço para um ambiente com provisionamento autogerenciado pode incluir uma definição de pipeline (por exemplo, um AWS CodePipeline pipeline), escrita em Terraform HCL. AWS Proton Para permitir o provisionamento desses pipelines, um administrador fornece um repositório de pipeline vinculado a. AWS Proton Ao provisionar um pipeline, a ação de automação de propriedade do cliente no repositório deve assumir uma função do IAM com permissões para provisionar o pipeline e uma identidade que reflita a conta de destino. AWS O repositório e o perfil do pipeline são separados daqueles usados para cada ambiente individual. O repositório vinculado é fornecido AWS Proton separadamente, somente uma vez no nível da AWS conta, e é usado para provisionar e gerenciar todos os pipelines. O perfil deve ter permissões para criar pipelines e outros recursos de que seus pipelines precisam.

  Os procedimentos a seguir mostram como fornecer o repositório e o perfil de pipeline para o AWS Proton.

  AWS Proton console

  Para fornecer o perfil do pipeline

  1. No console do AWS Proton, no painel de navegação, escolha Configurações > Configurações da conta e, em seguida, escolha Configurar.
  2. Use a seção de repositório de pipeline de CI/CD para configurar um link de repositório novo ou existente.

  AWS Proton API

  Para fornecer o perfil do pipeline

  1. Use a ação UpdateAccountSettingsda API.

2. Forneça o provedor, o nome e a ramificação do seu repositório de pipeline no parâmetro do `pipelineProvisioningRepository`.

AWS CLI

Para fornecer o perfil do pipeline

Execute o seguinte comando:

```
$ aws proton update-account-settings \
    --pipeline-provisioning-repository \
        "provider=GITHUB,name=my-pipeline-repo-name,branch=my-branch"
```

- Exclusão de recursos provisionados autogerenciados: os módulos do Terraform podem incluir elementos de configuração necessários para a operação do Terraform, além das definições de recursos. Portanto, não é AWS Proton possível excluir todos os arquivos do Terraform de um ambiente ou instância de serviço. Em vez disso, AWS Proton marca os arquivos para exclusão e atualiza um sinalizador nos metadados do PR. Sua automação pode ler essa bandeira e usá-la para acionar um comando terraform destroy.

AWS Proton terminologia

Modelo de ambiente

Define a infraestrutura compartilhada, como um VPC ou cluster, que é usada por várias aplicações ou recursos.

Pacote de modelos de ambiente

Uma coleção de arquivos que você carrega para criar e registrar um modelo de ambiente no AWS Proton. Um pacote de modelos de ambiente contém o seguinte:

1. Um arquivo de esquema que define parâmetros de entrada de infraestrutura como código.
2. Um arquivo de infraestrutura como código (IaC) que define a infraestrutura compartilhada, como um VPC ou cluster, que é usada por várias aplicações ou recursos.
3. Um arquivo de manifesto que lista o arquivo IaC.

Environment

Infraestrutura compartilhada provisionada, como um VPC ou cluster, que é usada por várias aplicações ou recursos.

Modelo de serviço

Define o tipo de infraestrutura necessária para implantar e manter uma aplicação ou um microsserviço em um ambiente.

Pacote de modelos de serviço

Uma coleção de arquivos que você carrega para criar e registrar um modelo de serviço no AWS Proton. Um pacote de modelos de serviço contém o seguinte:

1. Um arquivo de esquema que define parâmetros de entrada de infraestrutura como código (Iac).
2. Um arquivo IaC que define a infraestrutura necessária para implantar e manter uma aplicação ou um microsserviço em um ambiente.
3. Um arquivo de manifesto que lista o arquivo IaC.
4. Opcional
   a. Um arquivo IaC que define a infraestrutura do pipeline de serviços.
   b. Um arquivo de manifesto que lista o arquivo IaC.

Serviço

A infraestrutura provisionada necessária para implantar e manter uma aplicação ou um microsserviço em um ambiente.

Instância de serviço

A infraestrutura que suporta uma aplicação ou um microsserviço em um ambiente.

Pipeline de serviço

Infraestrutura provisionada que dá suporte a um pipeline.

Versão do modelo

Uma versão principal ou secundária de um modelo. Para obter mais informações, consulte Modelos versionados.

Parâmetros de entrada

Definido em um arquivo de esquema e usado em um arquivo de infraestrutura como código (IaC) para que o arquivo IaC possa ser usado de forma repetida e para uma variedade de casos de uso.

Arquivo de esquema

Define a infraestrutura como parâmetros de entrada de arquivo de código.

Arquivo de especificação

Especifica valores para infraestrutura como parâmetros de entrada do arquivo de código, conforme definido em um arquivo de esquema.

Arquivo manifesto

Lista uma infraestrutura como arquivo de código.

Criação de modelos e criação de pacotes para AWS Proton

AWS Proton provisiona recursos para você com base na infraestrutura como arquivos de código (IaC). Você descreve a infraestrutura em arquivos IaC reutilizáveis. Para tornar os arquivos reutilizáveis para diferentes ambientes e aplicações, você os cria como modelos, define parâmetros de entrada e usa esses parâmetros nas definições do IaC. Quando você cria posteriormente um recurso de provisionamento (ambiente, instância de serviço ou componente), AWS Proton usa um mecanismo de renderização, que combina valores de entrada com um modelo para criar um arquivo IaC pronto para provisionamento.

Os administradores criam a maioria dos modelos como pacotes de modelos e, em seguida, os carregam e os registram. AWS Proton O restante desta página aborda esses pacotes de AWS Proton modelos. Componentes definidos diretamente são uma exceção—os desenvolvedores os criam e fornecem arquivos de modelo de IaC diretamente. Para obter mais informações sobre componentes, consulte Componentes.

Tópicos

Pacotes de modelos

Como administrador, você cria e registra modelos com AWS Proton. Você usa esses modelos para criar ambientes e serviços. Quando você cria um serviço, AWS Proton provisiona e implanta instâncias de serviço em ambientes selecionados. Para obter mais informações, consulte AWS Proton para equipes de plataforma.

Para criar e registrar um modelo AWS Proton, você carrega um pacote de modelos que contém a infraestrutura como arquivos de código (IaC) que AWS Proton precisam provisionar um ambiente ou serviço.

Um pacote de modelos contém o seguinte:

- Um arquivo de infraestrutura como código (IaC) com um arquivo de manifesto de YAML que lista o arquivo IaC.
- Um arquivo YAML de esquema para as definições dos parâmetros de entrada do arquivo IaC.

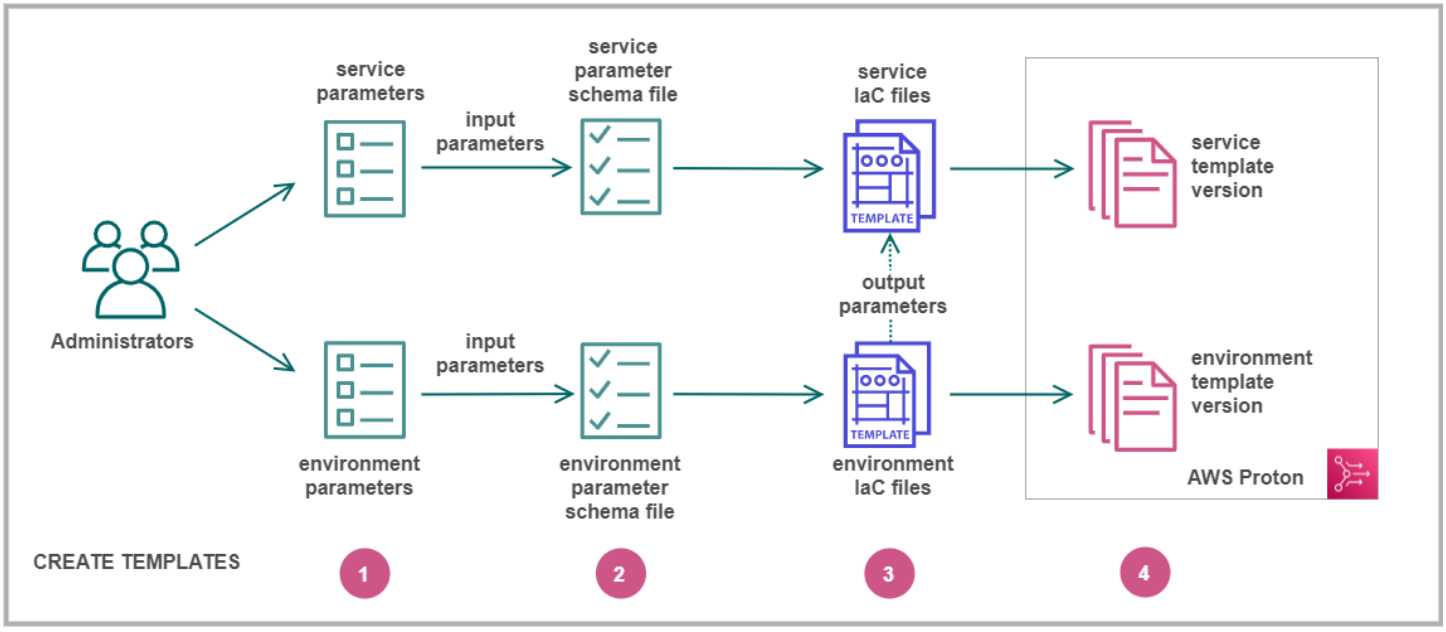
Um pacote de modelo de CloudFormation ambiente contém um arquivo IaC.

Um pacote CloudFormation de modelo de serviço contém um arquivo IaC para definições de instância de serviço e outro arquivo IaC opcional para uma definição de pipeline.

Os pacotes de modelos de serviço e ambientes do Terraform podem conter vários arquivos IaC cada.

AWS Proton requer um arquivo de esquema de parâmetros de entrada. Quando você usa AWS CloudFormation para criar seus arquivos IaC, você usa a sintaxe Jinja para referenciar seus parâmetros de entrada. AWS Proton fornece namespaces de parâmetros que você pode usar para referenciar parâmetros em seus arquivos IaC.

O diagrama a seguir mostra um exemplo das etapas que você pode seguir para criar um modelo para AWS Proton.



1

Identifique os parâmetros de entrada.

2

Crie um arquivo de esquema para definir seus parâmetros de entrada.

3

Crie arquivos IaC que façam referência aos seus parâmetros de entrada. Você pode referenciar as saídas de arquivos IaC do ambiente como entradas para seus arquivos IaC de serviço.

4

Registr

uma versão de modelo AWS Proton e faça o upload do seu pacote de modelos.

AWS Proton parâmetros

Você pode definir e usar parâmetros em seus arquivos de infraestrutura como código (IaC) para torná-los flexíveis e reutilizáveis. Você lê um valor de parâmetro em seus arquivos IaC fazendo referência ao nome do parâmetro no namespace do AWS Proton parâmetro. AWS Proton injeta valores de parâmetros nos arquivos IaC renderizados que ele gera durante o provisionamento de recursos. Para processar os parâmetros AWS CloudFormation do IaC, AWS Proton usa o Jinja. Para processar os parâmetros do Terraform IaC, AWS Proton gera um arquivo de valor do parâmetro do Terraform e conta com a capacidade de parametrização incorporada ao HCL.

ComCodeBuild aprovisionamento, AWS Proton gera um arquivo de entrada que seu código pode importar. O arquivo é um arquivo JSON ou HCL, dependendo de uma propriedade no manifesto do seu modelo. Para obter mais informações, consulte the section called “CodeBuild parâmetros de provisionamento”.

Você pode consultar os parâmetros em seus arquivos IaC de ambiente, serviço e componente ou código de provisionamento com os seguintes requisitos:

- O tamanho de cada nome de parâmetro não excede 100 caracteres.
- O tamanho do namespace do parâmetro e do nome do recurso combinados não excede o limite de caracteres para o nome do recurso.

AWS Proton o provisionamento falhará se essas cotas forem excedidas.

Tipos de parâmetros

Os seguintes tipos de parâmetros estão disponíveis para referência nos arquivos AWS Proton IaC:

Parâmetro de entrada

Ambientes e instâncias de serviço podem usar parâmetros de entrada que você define em um arquivo de esquema que você associa ao ambiente ou ao modelo de serviço. Você pode consultar os parâmetros de entrada de um recurso no arquivo IaC do recurso. Os arquivos IaC do componente podem se referir aos parâmetros de entrada da instância de serviço à qual o componente está anexado.

AWS Proton verifica os nomes dos parâmetros de entrada em relação ao seu arquivo de esquema e os compara com os parâmetros referenciados nos seus arquivos IaC para injetar os valores de entrada que você fornece em um arquivo de especificação durante o provisionamento de recursos.

Parâmetro de saída

Você pode definir saídas em qualquer um dos seus arquivos IaC. Uma saída pode ser, por exemplo, um nome, ID ou ARN de um dos recursos provisionados pelo modelo, ou pode ser uma forma de passar por uma das entradas do modelo. Você pode consultar essas saídas em arquivos IaC de outros recursos.

Nos arquivos CloudFormation IaC, defina os parâmetros de saída no `Outputs:` bloco. Em um arquivo Terraform IaC, defina cada parâmetro de saída usando uma instrução de `output`.

Parâmetro de recurso

AWS Proton cria automaticamente parâmetros AWS Proton de recursos. Esses parâmetros expõem as propriedades do objeto de AWS Proton recurso. Um exemplo de parâmetro de recurso é `environment.name`.

Usando AWS Proton parâmetros em seus arquivos IaC

Para ler um valor de parâmetro em um arquivo IaC, você se refere ao nome do parâmetro no namespace do AWS Proton parâmetro. Para arquivos AWS CloudFormation IaC, você usa a sintaxe Jinja e delimita o parâmetro com pares de chaves e aspas.

A tabela a seguir mostra a sintaxe de referência para cada linguagem de modelo compatível, com um exemplo.

Linguagem do modelo	Sintaxe	Exemplo: entrada de ambiente chamada “VPC”
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
Terraform	`var.parameter-name`	`var.environment.inputs.VPC` Definições de variáveis do Terraform geradas

> **Note**
>
> Se você usar parâmetros CloudFormation dinâmicos em seu arquivo IaC, deverá evitá-los para evitar erros de interpretação errônea do Jinja. Para obter mais informações, consulte Solução de problemas AWS Proton.

A tabela a seguir lista os nomes dos namespaces para todos os parâmetros do AWS Proton recurso. Cada tipo de arquivo de modelo pode usar um subconjunto diferente do namespace do parâmetro.

Arquivo de modelo	Tipo de parâmetro	Nome do parâmetro	Descrição
Ambiente	recurso	`environment.` **`name`**	Nome do ambiente
	input	`environment.inputs.` *`input-name`*	Entradas de ambiente definidas pelo esquema
Serviço	recurso	`environment.` **`name`** `environment.` **`account_id`**	Nome e Conta da AWS ID do ambiente
	saída	`environment.outputs.` *`output-name`*	Saídas de arquivo IaC do ambiente
	recurso	`service.`**`branch_name`** `service.`**`name`**	Nome do serviço e repositório de códigos

Arquivo de modelo	Tipo de parâmetro	Nome do parâmetro	Descrição
		`service.`**`repository_connection_arn`** `service.`**`repository_id`**	
	recurso	`service_instance.` **`name`**	Nome da instância de serviço
	input	`service_instance.inputs.` *`input-name`*	Entradas de instância de serviço definidas pelo esquema
	recurso	`service_instance.components.default.` **`name`**	Nome do componente padrão anexado
	saída	`service_instance.components.default.outputs.` *`output-name`*	Saídas de arquivo IaC do componente padrão anexado
Pipeline	recurso	`service_instance.environment.` **`name`** `service_instance.environment.` **`account_id`**	Nome e Conta da AWS ID do ambiente da instância de serviço
	saída	`service_instance.environment.outputs.` *`output-name`*	Saídas de arquivo IaC do ambiente de instância de serviço
	input	`pipeline.inputs.` *`input-name`*	Entradas de pipeline definidas pelo esquema

Arquivo de modelo	Tipo de parâmetro	Nome do parâmetro	Descrição
	recurso	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nome do serviço e repositório de códigos
	input	`service_instance.inputs.` *`input-name`*	Entradas de instância de serviço definidas pelo esquema
	conjunto	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	Uma coleção de instâncias de serviço que você pode percorrer
Componente	recurso	`environment.` **`name`** `environment.` **`account_id`**	Nome do ambiente e ID Conta da AWS da conta
	saída	`environment.outputs.` *`output-name`*	Saídas de arquivo IaC do ambiente
	recurso	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nome do serviço e repositório de códigos (componentes anexados)
	recurso	`service_instance.` **`name`**	Nome da instância de serviço (componentes anexados)

Arquivo de modelo	Tipo de parâmetro	Nome do parâmetro	Descrição
	input	`service_instance.inputs.` *`input-name`*	Entradas de instância de serviço definidas pelo esquema (componentes anexados)
	recurso	`component.` **`name`**	Nome do componente

Para obter mais informações e exemplos, consulte os subtópicos sobre parâmetros nos arquivos de modelo de IaC para diferentes tipos de recursos e linguagens de modelo.

Tópicos

- Detalhes e CloudFormation exemplos de parâmetros do arquivo Environment IaC
- Detalhes e exemplos de parâmetros do arquivo Service CloudFormation IaC
- Detalhes e exemplos de parâmetros do arquivo do componente CloudFormation IaC
- Filtros de parâmetros para arquivos CloudFormation IaC
- CodeBuild detalhes e exemplos de parâmetros de provisionamento
- Detalhes e exemplos de parâmetros de arquivo de infraestrutura como código (IaC) do Terraform

Detalhes e CloudFormation exemplos de parâmetros do arquivo Environment IaC

Você pode definir e referenciar parâmetros em seus arquivos de ambiente de infraestrutura como código (IaC). Para obter uma descrição detalhada dos AWS Proton parâmetros, dos tipos de parâmetros, do namespace do parâmetro e de como usar os parâmetros em seus arquivos IaC, consulte. the section called “Parâmetros”

Definir os parâmetros do ambiente

Você pode definir os parâmetros de entrada e saída para arquivos IaC do ambiente.

- Parâmetros de entrada: Defina os parâmetros de entrada do ambiente em seu arquivo de esquema.

A lista a seguir inclui exemplos de parâmetros de entrada do ambiente para casos de uso típicos.

- Valores CIDR de VPC
- Configurações do load balancer
- Configurações do banco de dados
- Tempo limite de verificação de integridade

Como administrador, você pode fornecer valores para os parâmetros de entrada ao criar um ambiente:

- Use o console para preencher um formulário baseado em esquema que AWS Proton fornece.
- Use a CLI para fornecer uma especificação que inclua os valores.

- Parâmetros de saída: Defina as saídas do ambiente nos arquivos IaC do seu ambiente. Você pode então consultar essas saídas em arquivos IaC de outros recursos.

Leia valores de parâmetros em arquivos IaC do ambiente

Você pode ler os parâmetros relacionados ao ambiente nos arquivos IaC do ambiente. Você lê um valor de parâmetro referindo o nome do parâmetro no namespace do parâmetro do AWS Proton .

- Parâmetros de entrada: Leia um valor de entrada do ambiente fazendo referência a `environment.inputs.input-name`.
- Parâmetros do recurso — Leia os parâmetros do AWS Proton recurso fazendo referência a nomes como`environment.name`.

> **Note**
>
> Nenhum parâmetro de saída de outros recursos está disponível para os arquivos IaC do ambiente.

Exemplos de arquivos IaC de ambiente e serviço com parâmetros

O exemplo a seguir demonstra a definição e a referência de parâmetros em um arquivo IaC do ambiente. Em seguida, o exemplo mostra como os parâmetros de saída do ambiente definidos no arquivo IaC do ambiente podem ser referenciados em um arquivo IaC do serviço.

Example Arquivo de ambiente CloudFormation IaC

Neste exemplo, observe o seguinte:

- O namespace `environment.inputs.` se refere aos parâmetros de entrada do ambiente.
- O parâmetro Amazon EC2 Systems Manager (SSM) `StoreInputValue` concatena as entradas do ambiente.
- A saída `MyEnvParameterValue` expõe a mesma concatenação de parâmetros de entrada que um parâmetro de saída. Três parâmetros de saída adicionais também expõem os parâmetros de entrada individualmente.
- Seis parâmetros de saída adicionais expõem os recursos que o ambiente fornece.

```
Resources:
  StoreInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ environment.inputs.my_sample_input }}
 {{ environment.inputs.my_other_sample_input}}
 {{ environment.inputs.another_optional_input }}"
              # input parameter references

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'environment.outputs' namespace, for example,
 service_instance.environment.outputs.ClusterName.
Outputs:
  MyEnvParameterValue:                                    # output definition
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                     # output definition
    Value: "{{ environment.inputs.my_sample_input }}"         #   input parameter
 reference
  MyOtherSampleInputValue:                                # output definition
    Value: "{{ environment.inputs.my_other_sample_input }}"   #   input parameter
 reference
  AnotherOptionalInputValue:                              # output definition
    Value: "{{ environment.inputs.another_optional_input }}"  #   input parameter
 reference
  ClusterName:                                            # output definition
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'                              #   provisioned resource
```

```
  ECSTaskExecutionRole:                                    # output definition
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'              #   provisioned resource
  VpcId:                                                   # output definition
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'                                      #   provisioned resource
  PublicSubnetOne:                                         # output definition
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'                          #   provisioned resource
  PublicSubnetTwo:                                         # output definition
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'                          #   provisioned resource
  ContainerSecurityGroup:                                  # output definition
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'                   #   provisioned resource
```

Example Arquivo Service CloudFormation IaC

O namespace `environment.outputs.` se refere às saídas do ambiente de um arquivo IaC do ambiente. Por exemplo, o nome `environment.outputs.ClusterName` lê o valor do parâmetro de saída do ambiente `ClusterName`.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
```

```
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}' # resource parameter
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}'  # output
reference to an environment infrastructure code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'  # resource parameter
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}' # resource parameter
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}' # resource parameter

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'  # resource parameter
```

```
      Cluster: '{{environment.outputs.ClusterName}}' # output reference to an
 environment infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'# input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output reference to an
 environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}' # output reference to an
 environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}' # output reference to an
 environment infrastructure as code file
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'  # resource parameter
          ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          TargetGroupArn: !Ref 'TargetGroup'
[...]
```

Detalhes e exemplos de parâmetros do arquivo Service CloudFormation IaC

Você pode definir e referenciar parâmetros em seus arquivos de serviço e pipeline de infraestrutura como código (IaC). Para obter uma descrição detalhada dos parâmetros do AWS Proton , dos tipos de parâmetros, do namespace do parâmetro e de como usar os parâmetros em seus arquivos IaC, consulte the section called “Parâmetros”.

Definir parâmetros de serviço

Você pode definir os parâmetros de entrada e saída para arquivos IaC do serviço.

- Parâmetros de entrada: Defina os parâmetros de instância de serviço do ambiente em seu arquivo de esquema.

  A lista a seguir inclui exemplos de parâmetros de entrada de serviço para casos de uso típicos.

  - Porta

- Tamanho da tarefa
- Imagem
- Contagem desejada
- Arquivo Docker
- Comando de teste unitário

Você fornece valores para os parâmetros de entrada ao criar um serviço:

- Use o console para preencher um formulário baseado em esquema que AWS Proton fornece.
- Use a CLI para fornecer uma especificação que inclua os valores.

- Parâmetros de saída: Defina as saídas da instância de serviço em seus arquivos IaC de serviço. Você pode então consultar essas saídas em arquivos IaC de outros recursos.

Leia valores de parâmetros em arquivos IaC de serviço

Você pode ler os parâmetros relacionados ao serviço e a outros recursos nos arquivos do serviço IaC. Você lê o valor de um parâmetro fazendo referência ao nome do parâmetro no namespace do AWS Proton parâmetro.

- Parâmetros de entrada: Leia um valor de instância do serviço fazendo referência a `service_instance.inputs.input-name`.
- Parâmetros do AWS Proton recurso — Leia os parâmetros do recurso fazendo referência a nomes como `service.name`e `service_instance.name`, e. `environment.name`
- Parâmetros de saída: Leia as saídas de outros recursos referenciando `environment.outputs.output-name` ou `service_instance.components.default.outputs.output-name`.

Exemplo de arquivo IaC de serviço com parâmetros

O exemplo a seguir é um trecho de um arquivo CloudFormation IaC de serviço. O namespace `environment.outputs.` se refere às saídas do ambiente de um arquivo IaC do ambiente. O namespace `service_instance.inputs.` se refere aos parâmetros de entrada da instância de serviço. A `service_instance.name` propriedade se refere a um parâmetro AWS Proton de recurso.

```
Resources:
```

```
  StoreServiceInstanceInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ service.name }} {{ service_instance.name }}
 {{ service_instance.inputs.my_sample_service_instance_required_input }}
 {{ service_instance.inputs.my_sample_service_instance_optional_input }}
 {{ environment.outputs.MySampleInputValue }}
 {{ environment.outputs.MyOtherSampleInputValue }}"
             #  resource parameter references               # input parameter
 references
                                          # output references to an environment
 infrastructure as code file
Outputs:
  MyServiceInstanceParameter:                                                  #
 output definition
    Value: !Ref StoreServiceInstanceInputValue
  MyServiceInstanceRequiredInputValue:                                         #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_required_input }}"  #
 input parameter reference
  MyServiceInstanceOptionalInputValue:                                         #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_optional_input }}"  #
 input parameter reference
  MyServiceInstancesEnvironmentSampleOutputValue:                              #
 output definition
    Value: "{{ environment.outputs.MySampleInputValue }}"                      #
 output reference to an environment IaC file
  MyServiceInstancesEnvironmentOtherSampleOutputValue:                         #
 output definition
    Value: "{{ environment.outputs.MyOtherSampleInputValue }}"                 #
 output reference to an environment IaC file
```

Detalhes e exemplos de parâmetros do arquivo do componente CloudFormation IaC

Você pode definir e referenciar parâmetros em seus arquivos de componente de infraestrutura como código (IaC). Para obter uma descrição detalhada dos AWS Proton parâmetros, dos tipos de parâmetros, do namespace do parâmetro e de como usar os parâmetros em seus arquivos IaC, consulte. the section called “Parâmetros” Para obter mais informações sobre componentes, consulte Componentes.

Defina os parâmetros de saída do componente

Você pode definir os parâmetros de saída nos arquivos IaC do componente. Você pode então consultar essas saídas em arquivos IaC de serviço.

> **Note**
>
> Você não pode definir entradas para arquivos IaC de componentes. Os componentes conectados podem obter entradas da instância de serviço à qual estão conectados. Os componentes separados não têm entradas.

Leia valores de parâmetros em arquivos IaC de componente

Você pode ler os parâmetros relacionados ao componente e a outros recursos nos arquivos do componente IaC. Você lê o valor de um parâmetro fazendo referência ao nome do parâmetro no namespace do AWS Proton parâmetro.

- Parâmetros de entrada: Leia um valor anexo de instância de serviço fazendo referência a `service_instance.inputs.`*`input-name`*.
- Parâmetros do AWS Proton recurso — Leia os parâmetros do recurso fazendo referência a nomes como `component.name`e`service_instance.name`, e. `environment.name`
- Parâmetros de saída: Leia as saídas do ambiente referindo `environment.outputs.`*`output-name`*.

Exemplos de arquivos IaC de componente e serviço com parâmetros

O exemplo a seguir mostra um componente que provisiona um bucket do Amazon Simple Storage Service (Amazon S3) e uma política de acesso relacionada e expõe os Amazon Resource Names ARNs () de ambos os recursos como saídas de componentes. Um modelo de serviço IaC adiciona as saídas do componente como variáveis de ambiente de contêiner de uma tarefa do Amazon Elastic Container Service (Amazon ECS) para disponibilizar as saídas para o código executado no contêiner e adiciona a política de acesso ao bucket ao perfil da tarefa. O nome do bucket é baseado nos nomes do ambiente, serviço, instância de serviço e componente, o que significa que o bucket é acoplado a uma instância específica do modelo de componente que estende uma instância de serviço específica. Os desenvolvedores podem criar vários componentes personalizados com base nesse modelo de componente para provisionar buckets do Amazon S3 para diferentes instâncias de serviço e necessidades funcionais.

O exemplo mostra como você usa a sintaxe de `{{ ... }}` Jinja para se referir a componentes e outros parâmetros de recursos em seu arquivo IaC de serviço. Você pode usar instruções de `{% if ... %}` para adicionar blocos de instruções somente quando um componente é anexado à instância de serviço. As palavras-chave `proton_cfn_*` são filtros que você pode usar para limpar e formatar os valores dos parâmetros de saída. Para obter mais informações sobre os filtros, consulte the section called “CloudFormation filtros de parâmetros”.

Como administrador, você cria o arquivo de modelo do serviço IaC.

Example arquivo CloudFormation IaC de serviço usando um componente

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
```

```
      # ...
```

Como desenvolvedor, você cria o arquivo de modelo do componente IaC.

Example arquivo CloudFormation IaC do componente

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

Quando AWS Proton renderiza um CloudFormation modelo para sua instância de serviço e substitui todos os parâmetros por valores reais, o modelo pode ter a aparência do arquivo a seguir.

Example arquivo IaC CloudFormation renderizado por instância de serviço

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
```

```
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          Environment:
            - Name: BucketName
              Value: arn:aws:s3:us-
east-1:123456789012:environment_name-service_name-service_instance_name-component_name
            - Name: BucketAccessPolicyArn
              Value: arn:aws:iam::123456789012:policy/cfn-generated-policy-name
  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Filtros de parâmetros para arquivos CloudFormation IaC

Ao fazer referências a AWS Proton parâmetros em seus arquivos AWS CloudFormation IaC, você pode usar modificadores Jinja conhecidos como filtros para validar, filtrar e formatar valores de parâmetros antes de serem inseridos no modelo renderizado. As validações de filtro são particularmente úteis quando se referem aos parâmetros de saída do componente, porque a criação e a anexação do componente são feitas pelos desenvolvedores, e um administrador que usa as saídas do componente em um modelo de instância de serviço pode querer verificar sua existência e validade. No entanto, você pode usar filtros em qualquer arquivo do Jinja IaC.

As seções a seguir descrevem e definem os filtros de parâmetros disponíveis e fornecem exemplos. AWS Proton define a maioria desses filtros. O filtro `default` é um filtro Jinja embutido.

Propriedades do ambiente de formato para tarefas do Amazon ECS

Declaração

```
dict # proton_cfn_ecs_task_definition_formatted_env_vars (raw: boolean = True) # YAML
 list of dicts
```

Descrição

Esse filtro formata uma lista de saídas a serem usadas em uma propriedade de ambiente na seção `ContainerDefinition` de definição de tarefa do Amazon Elastic Container Service (Amazon ECS).

Defina `raw` como `False` para também validar o valor do parâmetro. Nesse caso, o valor é necessário para corresponder à expressão regular `^[a-zA-Z0-9_-]*$`. Se o valor falhar nessa validação, a renderização do modelo falhará.

Exemplo

Com o seguinte modelo de componente personalizado:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

E o seguinte modelo de serviço:

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            {{ service_instance.components.default.outputs
              | proton_cfn_ecs_task_definition_formatted_env_vars }}
```

O modelo de serviço renderizado é o seguinte:

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            - Name: Output1
              Value: hello
            - Name: Output2
              Value: world
```

Formatar propriedades do ambiente para funções do Lambda

Declaração

```
dict # proton_cfn_lambda_function_formatted_env_vars (raw: boolean = True) # YAML dict
```

Descrição

Esse filtro formata uma lista de saídas a serem usadas em uma propriedade Environment na `Properties` seção de uma definição de AWS Lambda função.

Defina `raw` como `False` para também validar o valor do parâmetro. Nesse caso, o valor é necessário para corresponder à expressão regular `^[a-zA-Z0-9_-]*$`. Se o valor falhar nessa validação, a renderização do modelo falhará.

Exemplo

Com o seguinte modelo de componente personalizado:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

E o seguinte modelo de serviço:

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          {{ service_instance.components.default.outputs
            | proton_cfn_lambda_function_formatted_env_vars }}
```

O modelo de serviço renderizado é o seguinte:

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          Output1: hello
          Output2: world
```

Extraia ARNs a política do IAM para incluir nas funções do IAM

Declaração

```
dict # proton_cfn_iam_policy_arns # YAML list
```

Descrição

Esse filtro formata uma lista de saídas a serem usadas em uma ManagedPolicyArns propriedade na `Properties` seção de uma definição de função AWS Identity and Access Management (IAM). O filtro usa a expressão regular `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` para extrair uma política do IAM válida ARNs da lista de parâmetros de saída. Você pode usar esse filtro para acrescentar políticas nos valores dos parâmetros de saída a uma definição de perfil do IAM em um modelo de serviço.

Exemplo

Com o seguinte modelo de componente personalizado:

```
Resources:
  # ...
  ExamplePolicy1:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
  ExamplePolicy2:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...

  # ...

Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
  PolicyArn1:
    Description: "ARN of policy 1"
    Value: !Ref ExamplePolicy1
  PolicyArn2:
    Description: "ARN of policy 2"
    Value: !Ref ExamplePolicy2
```

E o seguinte modelo de serviço:

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
```

```
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

O modelo de serviço renderizado é o seguinte:

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-1
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-2

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Higienize os valores das propriedades

Declaração

```
string # proton_cfn_sanitize # string
```

Descrição

Esse é um filtro de uso geral. Use-o para validar a segurança do valor de um parâmetro. O filtro valida se o valor corresponde à expressão regular `^[a-zA-Z0-9_-]*$` ou se é um nome do recurso da Amazon (ARN) válido. Se o valor falhar nessa validação, a renderização do modelo falhará.

Exemplo

Com o seguinte modelo de componente personalizado:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example of valid output"
    Value: "This-is_valid_37"
  Output2:
    Description: "Example incorrect output"
    Value: "this::is::incorrect"
  SomeArn:
    Description: "Example ARN"
    Value: arn:aws:some-service::123456789012:some-resource/resource-name
```

- A seguinte referência em um modelo de serviço:

```
# ...
  {{ service_instance.components.default.outputs.Output1
    | proton_cfn_sanitize }}
```

  Renderiza da seguinte forma:

```
# ...
  This-is_valid_37
```

- A seguinte referência em um modelo de serviço:

```
# ...
  {{ service_instance.components.default.outputs.Output2
    | proton_cfn_sanitize }}
```

  Resultados com o seguinte erro de renderização:

```
Illegal character(s) detected in "this::is::incorrect". Must match regex ^[a-zA-
Z0-9_-]*$ or be a valid ARN
```

- A seguinte referência em um modelo de serviço:

```
# ...
  {{ service_instance.components.default.outputs.SomeArn
    | proton_cfn_sanitize }}
```

Renderiza da seguinte forma:

```
# ...
  arn:aws:some-service::123456789012:some-resource/resource-name
```

Fornecer valores padrão para referências inexistentes

Descrição

O filtro `default` fornece um valor padrão quando não existe uma referência de namespace. Use-o para escrever modelos robustos que possam ser renderizados sem falhas, mesmo quando o parâmetro ao qual você se refere estiver ausente.

Exemplo

A referência a seguir em um modelo de serviço faz com que a renderização do modelo falhe se a instância de serviço não tiver um componente anexado diretamente definido (padrão) ou se o componente anexado não tiver uma saída chamada `test`.

```
# ...
  {{ service_instance.components.default.outputs.test }}
```

Para evitar esse problema, adicione o filtro `default`.

```
# ...
  {{ service_instance.components.default.outputs.test | default("[optional-value]") }}
```

CodeBuild detalhes e exemplos de parâmetros de provisionamento

Você pode definir parâmetros em seus modelos para AWS Proton recursos CodeBuild baseados e referenciar esses parâmetros em seu código de provisionamento. Para obter uma descrição detalhada dos AWS Proton parâmetros, dos tipos de parâmetros, do namespace do parâmetro e de como usar os parâmetros em seus arquivos IaC, consulte. the section called “Parâmetros”

> **Note**
>
> Você pode usar o CodeBuild provisionamento com ambientes e serviços. No momento, você não pode provisionar componentes dessa forma.

Parâmetros de entrada

Ao criar um AWS Proton recurso, como um ambiente ou um serviço, você fornece valores para os parâmetros de entrada definidos no arquivo de esquema do seu modelo. Quando o recurso que você cria usaCodeBuild aprovisionamento, AWS Proton renderiza esses valores de entrada em um arquivo de entrada. Seu código de aprovisionamento pode importar e obter valores de parâmetros desse arquivo.

Para obter um exemplo de um CodeBuild modelo, consultethe section called “CodeBuild pacote”. Para obter mais informações sobre arquivos manifesto, consulte the section called “Manifesto e conclusão”.

O exemplo a seguir é um arquivo de entrada JSON gerado durante o provisionamento CodeBuild baseado de uma instância de serviço.

Exemplo: usando o AWS CDK com CodeBuild provisionamento

```
{
  "service_instance": {
    "name": "my-service-staging",
    "inputs": {
      "port": "8080",
      "task_size": "medium"
    }
  },
  "service": {
    "name": "my-service"
  },
  "environment": {
    "account_id": "123456789012",
    "name": "my-env-staging",
    "outputs": {
      "vpc-id": "hdh2323423"
    }
  }
}
```

Parâmetros de saída

Para comunicar as saídas do provisionamento de recursos AWS Proton, seu código de provisionamento pode gerar um arquivo JSON nomeado `proton-outputs.json` com valores

para os parâmetros de saída definidos no arquivo de esquema do seu modelo. Por exemplo, o `cdk deploy` comando tem o `--outputs-file` argumento que instrui o AWS CDK a gerar um arquivo JSON com saídas de provisionamento. Se seu recurso usa o AWS CDK, especifique o seguinte comando no manifesto CodeBuild do modelo:

```
aws proton notify-resource-deployment-status-change
```

AWS Proton procura esse arquivo JSON. Se o arquivo existir após a conclusão bem-sucedida do código de provisionamento, AWS Proton lerá os valores dos parâmetros de saída dele.

Detalhes e exemplos de parâmetros de arquivo de infraestrutura como código (IaC) do Terraform

Você pode incluir variáveis de entrada do Terraform em arquivos `variable.tf` em seu pacote de modelos. Você também pode criar um esquema para criar variáveis AWS Proton gerenciadas. AWS Proton cria uma variável `.tf files` a partir do seu arquivo de esquema. Para obter mais informações, consulte the section called “Arquivos Terraform IaC”.

Para referenciar AWS Proton as variáveis definidas pelo esquema em sua infraestrutura`.tf files`, você usa os AWS Proton namespaces mostrados na tabela Parâmetros e namespaces do Terraform IaC. Por exemplo, você poderá usar o `var.environment.inputs.vpc_cidr`. Entre aspas, coloque essas variáveis entre colchetes simples e adicione um cifrão na frente da primeira chave (por exemplo, “`${var.environment.inputs.vpc_cidr}`”).

O exemplo a seguir mostra como usar namespaces para incluir AWS Proton parâmetros em um ambiente. `.tf file`

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
```

```
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

AWS Proton infraestrutura como arquivos de código

As partes principais do pacote de modelos são arquivos de infraestrutura como código (IaC) que definem os recursos e propriedades de infraestrutura que você deseja provisionar. AWS CloudFormation e outras infraestruturas como mecanismos de código usam esses tipos de arquivos para provisionar recursos de infraestrutura.

> **Note**
>
> Um arquivo IaC também pode ser usado independentemente dos pacotes de modelos, como uma entrada direta para componentes definidos diretamente. Para obter mais informações sobre componentes, consulte Componentes.

AWS Proton atualmente suporta dois tipos de arquivos IaC:

- CloudFormationarquivos — Usados para provisionamento AWS gerenciado. AWS Proton usa Jinja em cima do formato de arquivo de CloudFormation modelo para parametrização.
- Arquivos HCL do Terraform: usados para provisionamento autogerenciado. O HCL suporta nativamente a parametrização.

Você não pode provisionar AWS Proton recursos usando uma combinação de métodos de provisionamento. Você deve usar um ou outro. Você não pode implantar um serviço AWS de provisionamento gerenciado em um ambiente de provisionamento autogerenciado ou vice-versa.

Para ter mais informações, consulte the section called “Métodos de provisionamento”, Ambientes do, Serviços e Componentes.

CloudFormation Arquivos IaC

Aprenda a usar a AWS CloudFormation infraestrutura como arquivos de código com AWS Proton. CloudFormation é um serviço de infraestrutura como código (IaC) que ajuda você a modelar e configurar seus AWS recursos. Você define seus recursos de infraestrutura em modelos, usando o Jinja em cima do formato de arquivo de CloudFormation modelo para parametrização. AWS Proton expande os parâmetros e renderiza o modelo completo CloudFormation . CloudFormation provisiona os recursos definidos como CloudFormation pilha. Para obter mais informações, consulte O que é o CloudFormation no Guia do usuário do CloudFormation .

AWS Proton suporta provisionamento AWS gerenciado para IaC. CloudFormation

Comece com sua própria infraestrutura existente como arquivos de código

Você pode adaptar sua própria infraestrutura existente como arquivos de código (IaC) para uso com AWS Proton.

Os CloudFormation exemplos a seguir, Exemplo 1 e Exemplo 2, representam seus próprios arquivos CloudFormation IaC existentes. CloudFormation pode usar esses arquivos para criar duas CloudFormation pilhas diferentes.

No Exemplo 1, o arquivo CloudFormation IaC é configurado para provisionar a infraestrutura a ser compartilhada entre aplicativos de contêiner. Neste exemplo, parâmetros de entrada são adicionados para que você possa usar o mesmo arquivo IaC para criar vários conjuntos de infraestrutura provisionada. Cada conjunto pode ter nomes diferentes junto com um conjunto diferente de valores CIDR de VPC e sub-rede. Como administrador ou desenvolvedor, você fornece valores para esses parâmetros ao usar um arquivo IaC para provisionar recursos de CloudFormation infraestrutura. Para sua conveniência, esses parâmetros de entrada são marcados com comentários e referenciados várias vezes no exemplo. As saídas são definidas no final do modelo. Eles podem ser referenciados em outros arquivos CloudFormation IaC.

No Exemplo 2, o arquivo CloudFormation IaC é configurado para implantar um aplicativo na infraestrutura provisionada a partir do Exemplo 1. Os parâmetros são comentados para a sua conveniência.

Exemplo 1: arquivo CloudFormation IaC

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery namespaces.
Parameters:
   VpcCIDR:       # input parameter
        Description: CIDR for VPC
        Type: String
        Default: "10.0.0.0/16"
   SubnetOneCIDR: # input parameter
        Description: CIDR for SubnetOne
        Type: String
        Default: "10.0.0.0/24"
   SubnetTwoCIDR: # input parameters
        Description: CIDR for SubnetTwo
        Type: String
        Default: "10.0.1.0/24"
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock:
        Ref: 'VpcCIDR'

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
         Ref: 'SubnetOneCIDR'
      MapPublicIpOnLaunch: true
```

```
  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 1
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
        Ref: 'SubnetTwoCIDR'
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable
```

```
  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster

  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values will be available to other templates to use.
Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSCluster"
  ECSTaskExecutionRole:                                  # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSTaskExecutionRole"
  VpcId:                                                 # output
    Description: The ID of the VPC that this stack is deployed in
```

```
    Value: !Ref 'VPC'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-VPC"
  PublicSubnetOne:                                          # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetOne"
  PublicSubnetTwo:                                          # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetTwo"
  ContainerSecurityGroup:                                   # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ContainerSecurityGroup"
```

Exemplo 2: arquivo CloudFormation IaC

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Parameters:
    ContainerPortInput:  # input parameter
        Description: The port to route traffic to
        Type: Number
        Default: 80
    TaskCountInput:      # input parameter
        Description: The default number of Fargate tasks you want running
        Type: Number
        Default: 1
    TaskSizeInput:       # input parameter
        Description: The size of the task you want to run
        Type: String
        Default: x-small
    ContainerImageInput: # input parameter
        Description: The name/url of the container image
```

```
        Type: String
        Default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
    TaskNameInput:       # input parameter
        Description: Name for your task
        Type: String
        Default: "my-fargate-instance"
    StackName:           # input parameter
        Description: Name of the environment stack to deploy to
        Type: String
        Default: "my-fargate-environment"
Mappings:
  TaskSizeMap:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName:
        Ref: 'TaskNameInput' # input parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref 'TaskNameInput'
      Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
      Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
      NetworkMode: awsvpc
```

```
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn:
        Fn::ImportValue:
          !Sub "${StackName}-ECSTaskExecutionRole"    # output parameter from another
CloudFormation template
             awslogs-region: !Ref 'AWS::Region'
             awslogs-stream-prefix: !Ref 'TaskNameInput'


 # The service_instance. The service is a resource which allows you to run multiple
 # copies of a type of task, and gather up their logs and metrics, as well
 # as monitor the number of running tasks and replace any that have crashed
 Service:
   Type: AWS::ECS::Service
   DependsOn: LoadBalancerRule
   Properties:
     ServiceName: !Ref 'TaskNameInput'
     Cluster:
       Fn::ImportValue:
         !Sub "${StackName}-ECSCluster"  # output parameter from another
CloudFormation template
     LaunchType: FARGATE
     DeploymentConfiguration:
       MaximumPercent: 200
       MinimumHealthyPercent: 75
     DesiredCount: !Ref 'TaskCountInput'
     NetworkConfiguration:
       AwsvpcConfiguration:
         AssignPublicIp: ENABLED
         SecurityGroups:
           - Fn::ImportValue:
               !Sub "${StackName}-ContainerSecurityGroup"    # output parameter from
another CloudFormation template
         Subnets:
           - Fn::ImportValue:r CloudFormation template
     TaskRoleArn: !Ref "AWS::NoValue"
     ContainerDefinitions:
       - Name: !Ref 'TaskNameInput'
         Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
         Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
         Image: !Ref 'ContainerImageInput'             # input parameter
         PortMappings:
           - ContainerPort: !Ref 'ContainerPortInput'  # input parameter
```

```
        LogConfiguration:
          LogDriver: 'awslogs'
          Options:
            awslogs-group: !Ref 'TaskNameInput'
              !Sub "${StackName}-PublicSubnetOne"    # output parameter from another
CloudFormation template
          - Fn::ImportValue:
              !Sub "${StackName}-PublicSubnetTwo"    # output parameter from another
CloudFormation template
     TaskDefinition: !Ref 'TaskDefinition'
     LoadBalancers:
       - ContainerName: !Ref 'TaskNameInput'
         ContainerPort: !Ref 'ContainerPortInput'  # input parameter
         TargetGroupArn: !Ref 'TargetGroup'

 # A target group. This is used for keeping track of all the tasks, and
 # what IP addresses / port numbers they have. You can query it yourself,
 # to use the addresses yourself, but most often this target group is just
 # connected to an application load balancer, or network load balancer, so
 # it can automatically distribute traffic across all the targets.
 TargetGroup:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     HealthCheckIntervalSeconds: 6
     HealthCheckPath: /
     HealthCheckProtocol: HTTP
     HealthCheckTimeoutSeconds: 5
     HealthyThresholdCount: 2
     TargetType: ip
     Name: !Ref 'TaskNameInput'
     Port: !Ref 'ContainerPortInput'
     Protocol: HTTP
     UnhealthyThresholdCount: 2
     VpcId:
       Fn::ImportValue:
         !Sub "${StackName}-VPC"    # output parameter from another CloudFormation
template

 # Create a rule on the load balancer for routing traffic to the target group
 LoadBalancerRule:
   Type: AWS::ElasticLoadBalancingV2::ListenerRule
   Properties:
     Actions:
```

```
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      Conditions:
        - Field: path-pattern
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
```

```
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"  # output parameter from another
CloudFormation template
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
```

```
        - MetricIntervalLowerBound: 25
          ScalingAdjustment: 3
      MetricAggregationType: 'Average'
      Cooldown: 60

# Create alarms to trigger these policies
LowCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - low-cpu
          - !Ref 'TaskNameInput'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "Low CPU utilization for service"
          - !Ref 'TaskNameInput'
    MetricName: CPUUtilization
    Namespace: AWS/ECS
    Dimensions:
      - Name: ServiceName
        Value: !Ref 'TaskNameInput'
      - Name: ClusterName
        Value:
          Fn::ImportValue:
            !Sub "${StackName}-ECSCluster"
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 20
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleDownPolicy

HighCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - high-cpu
          - !Ref 'TaskNameInput'
```

```
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId:
        Fn::ImportValue:
          !Sub "${StackName}-ContainerSecurityGroup"
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"
      SecurityGroupIngress:
```

```
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
       - Fn::ImportValue:
           !Sub "${StackName}-PublicSubnetOne"
       - Fn::ImportValue:
           !Sub "${StackName}-PublicSubnetTwo"
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
# These output values will be available to other templates to use.
Outputs:
  ServiceEndpoint:        # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

Você pode adaptar esses arquivos para uso com AWS Proton.

Traga sua infraestrutura como código para AWS Proton

Com pequenas modificações, você pode usar o Exemplo 1 como um arquivo de infraestrutura como código (IaC) para um pacote de modelo de ambiente AWS Proton usado para implantar um ambiente (conforme mostrado no Exemplo 3).

Em vez de usar os CloudFormation parâmetros, você usa a sintaxe Jinja para referenciar parâmetros que você definiu em um arquivo de esquema baseado em API aberta. Esses parâmetros de entrada são comentados para sua conveniência e referenciados várias vezes no arquivo IaC. Dessa forma, AWS Proton pode auditar e verificar os valores dos parâmetros. Ele também pode combinar e inserir valores de parâmetros de saída em um arquivo IaC com parâmetros em outro arquivo IaC.

Como administrador, você pode adicionar o AWS Proton `environment.inputs.` namespace aos parâmetros de entrada. Ao fazer referência às saídas do arquivo IaC do ambiente em um arquivo IaC de serviço, você pode adicionar o `environment.outputs.` namespace às saídas (por exemplo, `environment.outputs.ClusterName`). Por fim, você os cerca com colchetes e aspas.

Com essas modificações, seus arquivos CloudFormation IaC podem ser usados por AWS Proton.

Exemplo 3: infraestrutura AWS Proton do ambiente como arquivo de código

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery prefixes.
Mappings:
  # The VPC and subnet configuration is passed in via the environment spec.
  SubnetConfig:
    VPC:
      CIDR: '{{ environment.inputs.vpc_cidr}}'        # input parameter
    PublicOne:
      CIDR: '{{ environment.inputs.subnet_one_cidr}}' # input parameter
    PublicTwo:
      CIDR: '{{ environment.inputs.subnet_two_cidr}}' # input parameter
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock: !FindInMap ['SubnetConfig', 'VPC', 'CIDR']

  # Two public subnets, where containers will have public IP addresses
```

```
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicOne', 'CIDR']
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 1
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicTwo', 'CIDR']
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
```

```
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster

  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'service_instance.environment.outputs.' namespace, for example,
 service_instance.environment.outputs.ClusterName.

Outputs:
  ClusterName:                                          # output
```

```
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:                                    # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:                                                   # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
  PublicSubnetOne:                                         # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
  PublicSubnetTwo:                                         # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
  ContainerSecurityGroup:                                  # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
```

Os arquivos IaC no Exemplo 1 e no Exemplo 3 produzem CloudFormation pilhas ligeiramente diferentes. Os parâmetros são exibidos de forma diferente nos arquivos de modelo da pilha. O arquivo de modelo de CloudFormation pilha do Exemplo 1 exibe os rótulos dos parâmetros (chaves) na visualização do modelo de pilha. O arquivo de modelo de pilha de AWS Proton CloudFormation infraestrutura do Exemplo 3 exibe os valores dos parâmetros. AWS Proton os parâmetros de entrada não aparecem na visualização dos parâmetros da CloudFormation pilha do console.

No Exemplo 4, o arquivo IaC do AWS Proton serviço corresponde ao Exemplo 2.

Exemplo 4: arquivo IaC da instância de AWS Proton serviço

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
```

```
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}'
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
 parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}' # output from an
 environment infrastructure as code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}'
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}'

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
```

```
 # as monitor the number of running tasks and replace any that have crashed
 Service:
   Type: AWS::ECS::Service
   DependsOn: LoadBalancerRule
   Properties:
     ServiceName: '{{service_instance.name}}'
     Cluster: '{{environment.outputs.ClusterName}}' # output from an environment
infrastructure as code file
     LaunchType: FARGATE
     DeploymentConfiguration:
       MaximumPercent: 200
       MinimumHealthyPercent: 75
     DesiredCount: '{{service_instance.inputs.desired_count}}'       # input parameter
     NetworkConfiguration:
       AwsvpcConfiguration:
         AssignPublicIp: ENABLED
         SecurityGroups:
           - '{{environment.outputs.ContainerSecurityGroup}}' # output from an
environment infrastructure as code file
         Subnets:
           - '{{environment.outputs.PublicSubnetOne}}'        # output from an
environment infrastructure as code file
           - '{{environment.outputs.PublicSubnetTwo}}'
     TaskDefinition: !Ref 'TaskDefinition'
     LoadBalancers:
       - ContainerName: '{{service_instance.name}}'
         ContainerPort: '{{service_instance.inputs.port}}'
         TargetGroupArn: !Ref 'TargetGroup'

 # A target group. This is used for keeping track of all the tasks, and
 # what IP addresses / port numbers they have. You can query it yourself,
 # to use the addresses yourself, but most often this target group is just
 # connected to an application load balancer, or network load balancer, so
 # it can automatically distribute traffic across all the targets.
 TargetGroup:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     HealthCheckIntervalSeconds: 6
     HealthCheckPath: /
     HealthCheckProtocol: HTTP
     HealthCheckTimeoutSeconds: 5
     HealthyThresholdCount: 2
     TargetType: ip
     Name: '{{service_instance.name}}'
```

```
      Port: '{{service_instance.inputs.port}}'
      Protocol: HTTP
      UnhealthyThresholdCount: 2
      VpcId: '{{environment.outputs.VpcId}}' # output from an environment
 infrastructure as code file

  # Create a rule on the load balancer for routing traffic to the target group
  LoadBalancerRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      Conditions:
        - Field: path-pattern
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}' # output from an environment
 infrastructure as code file
            - '{{service_instance.name}}'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
```

```
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
```

```
      AdjustmentType: 'ChangeInCapacity'
      StepAdjustments:
        - MetricIntervalLowerBound: 0
          MetricIntervalUpperBound: 15
          ScalingAdjustment: 1
        - MetricIntervalLowerBound: 15
          MetricIntervalUpperBound: 25
          ScalingAdjustment: 2
        - MetricIntervalLowerBound: 25
          ScalingAdjustment: 3
      MetricAggregationType: 'Average'
      Cooldown: 60

# Create alarms to trigger these policies
LowCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - low-cpu
          - '{{service_instance.name}}'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "Low CPU utilization for service"
          - '{{service_instance.name}}'
    MetricName: CPUUtilization
    Namespace: AWS/ECS
    Dimensions:
      - Name: ServiceName
        Value: '{{service_instance.name}}'
      - Name: ClusterName
        Value:
          '{{environment.outputs.ClusterName}}'
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 20
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleDownPolicy

HighCpuUsageAlarm:
```

```
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - high-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: '{{environment.outputs.ContainerSecurityGroup}}'
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
```

```
      VpcId: '{{environment.outputs.VpcId}}'
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - '{{environment.outputs.PublicSubnetOne}}'
        - '{{environment.outputs.PublicSubnetTwo}}'
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
Outputs:
  ServiceEndpoint:         # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

No Exemplo 5, o arquivo IaC do AWS Proton pipeline provisiona a infraestrutura do pipeline para dar suporte às instâncias de serviço provisionadas pelo Exemplo 4.

Exemplo 5: arquivo IaC do pipeline de AWS Proton serviço

```
Resources:
```

```
  ECRRepo:
    Type: AWS::ECR::Repository
    DeletionPolicy: Retain
  BuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: true
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: repo_name
          Type: PLAINTEXT
          Value: !Ref ECRRepo
        - Name: service_name
          Type: PLAINTEXT
          Value: '{{ service.name }}'    # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - PublishRole
          - Arn
      Source:
        BuildSpec:
          Fn::Join:
            - ""
            - - >-
                {
                  "version": "0.2",
                  "phases": {
                    "install": {
                      "runtime-versions": {
                        "docker": 18
                      },
                      "commands": [
                        "pip3 install --upgrade --user awscli",
                        "echo
 'f6bd1536a743ab170b35c94ed4c7c4479763356bd543af5d391122f4af852460  yq_linux_amd64' >
 yq_linux_amd64.sha",
                        "wget https://github.com/mikefarah/yq/releases/download/3.4.0/
yq_linux_amd64",
                        "sha256sum -c yq_linux_amd64.sha",
```

```
                      "mv yq_linux_amd64 /usr/bin/yq",
                      "chmod +x /usr/bin/yq"
                    ]
                  },
                  "pre_build": {
                    "commands": [
                      "cd $CODEBUILD_SRC_DIR",
                      "$(aws ecr get-login --no-include-email --region
 $AWS_DEFAULT_REGION)",
                      "{{ pipeline.inputs.unit_test_command }}",    # input parameter
                    ]
                  },
                  "build": {
                    "commands": [
                      "IMAGE_REPO_NAME=$repo_name",
                      "IMAGE_TAG=$CODEBUILD_BUILD_NUMBER",
                      "IMAGE_ID=
              - Ref: AWS::AccountId
              - >-
                .dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG",
                      "docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG -f
 {{ pipeline.inputs.dockerfile }} .",     # input parameter
                      "docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_ID;",
                      "docker push $IMAGE_ID"
                    ]
                  },
                  "post_build": {
                    "commands": [
                      "aws proton --region $AWS_DEFAULT_REGION get-service --name
 $service_name | jq -r .service.spec > service.yaml",
                      "yq w service.yaml 'instances[*].spec.image' \"$IMAGE_ID\" >
 rendered_service.yaml"
                    ]
                  }
                },
                "artifacts": {
                  "files": [
                    "rendered_service.yaml"
                  ]
                }
              }
        Type: CODEPIPELINE
      EncryptionKey:
```

```
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% for service_instance in service_instances %}
  Deploy{{loop.index}}Project:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: false
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: service_name
          Type: PLAINTEXT
          Value:  '{{service.name}}'          # resource parameter
        - Name: service_instance_name
          Type: PLAINTEXT
          Value: '{{service_instance.name}}'  # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - DeploymentRole
          - Arn
      Source:
        BuildSpec: >-
          {
            "version": "0.2",
            "phases": {
              "build": {
                "commands": [
                  "pip3 install --upgrade --user awscli",
                  "aws proton --region $AWS_DEFAULT_REGION update-service-instance
 --deployment-type CURRENT_VERSION --name $service_instance_name --service-name
 $service_name --spec file://rendered_service.yaml",
                  "aws proton --region $AWS_DEFAULT_REGION wait service-instance-
deployed --name $service_instance_name --service-name $service_name"
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
```

```
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
```

```
                - :log-group:/aws/codebuild/
                - Ref: BuildProject
                - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
```

```
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole

  DeploymentRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
```

```
            - ""
            - - "arn:"
              - Ref: AWS::Partition
              - ":codebuild:"
              - Ref: AWS::Region
              - ":"
              - Ref: AWS::AccountId
              - :report-group/Deploy*Project
              - -*
        - Action:
            - proton:UpdateServiceInstance
            - proton:GetServiceInstance
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
```

```
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: DeploymentRoleDefaultPolicy
    Roles:
      - Ref: DeploymentRole
PipelineArtifactsBucketEncryptionKey:
  Type: AWS::KMS::Key
  Properties:
    KeyPolicy:
      Statement:
        - Action:
            - kms:Create*
            - kms:Describe*
            - kms:Enable*
            - kms:List*
            - kms:Put*
            - kms:Update*
            - kms:Revoke*
            - kms:Disable*
            - kms:Get*
            - kms:Delete*
            - kms:ScheduleKeyDeletion
            - kms:CancelKeyDeletion
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
```

```
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
        - Action:
```

```
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
      Version: "2012-10-17"
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineArtifactsBucket:
  Type: AWS::S3::Bucket
  Properties:
    VersioningConfiguration:
      Status: Enabled
    BucketEncryption:
      ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            KMSMasterKeyID:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
            SSEAlgorithm: aws:kms
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
  UpdateReplacePolicy: Retain
  DeletionPolicy: Retain
PipelineArtifactsBucketEncryptionKeyAlias:
  Type: AWS::KMS::Alias
  Properties:
    AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'
    TargetKeyId:
      Fn::GetAtt:
        - PipelineArtifactsBucketEncryptionKey
        - Arn
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
```

```
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
```

```
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'
                FullRepositoryId: '{{ service.repository_id }}'
                BranchName: '{{ service.branch_name }}'
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
```

```
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
```

```
      Location:
        Ref: PipelineArtifactsBucket
      Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"
```

```
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
```

```
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":logs:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :log-group:/aws/codebuild/
                  - Ref: BuildProject
                  - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
```

```
            - proton:GetService
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
            - s3:DeleteObject*
            - s3:PutObject*
            - s3:Abort*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: PublishRoleDefaultPolicy
```

```
      Roles:
        - Ref: PublishRole

  DeploymentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
```

```
            - codebuild:CreateReport
            - codebuild:UpdateReport
            - codebuild:BatchPutTestCases
          Effect: Allow
          Resource:
            Fn::Join:
              - ""
              - - "arn:"
                - Ref: AWS::Partition
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/Deploy*Project
                - -*
        - Action:
            - proton:UpdateServiceInstance
            - proton:GetServiceInstance
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
```

```
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: DeploymentRoleDefaultPolicy
    Roles:
      - Ref: DeploymentRole
PipelineArtifactsBucketEncryptionKey:
  Type: AWS::KMS::Key
  Properties:
    KeyPolicy:
      Statement:
        - Action:
            - kms:Create*
            - kms:Describe*
            - kms:Enable*
            - kms:List*
            - kms:Put*
            - kms:Update*
            - kms:Revoke*
            - kms:Disable*
            - kms:Get*
            - kms:Delete*
            - kms:ScheduleKeyDeletion
            - kms:CancelKeyDeletion
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
```

```
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - PipelineRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - PublishRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - PublishRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Principal:
```

```
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
        Version: "2012-10-17"
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineArtifactsBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'     # resource
parameter
      TargetKeyId:
```

```
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
```

```
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'   # resource
parameter
                FullRepositoryId: '{{ service.repository_id }}'            # resource
parameter
```

```
                BranchName: '{{ service.branch_name }}'                    # resource
parameter
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
```

```
          Name: 'Deploy{{service_instance.name}}'         # resource parameter
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
```

```
              - BuildProject
              - Arn
      Version: "2012-10-17"
    PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
    Roles:
      - Ref: PipelineBuildCodePipelineActionRole
PipelineDeployCodePipelineActionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
      Version: "2012-10-17"
PipelineDeployCodePipelineActionRoleDefaultPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Statement:
        - Action:
            - codebuild:BatchGetBuilds
            - codebuild:StartBuild
            - codebuild:StopBuild
          Effect: Allow
          Resource:
            Fn::Join:
              - ""
              - - "arn:"
                - Ref: AWS::Partition
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - ":project/Deploy*"
      Version: "2012-10-17"
```

```
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"
```

CodeBuild pacote de modelos de provisionamento

Com o CodeBuild provisionamento, em vez de usar modelos de IaC para renderizar arquivos de IaC e executá-los usando um mecanismo de provisionamento de IaC, basta executar seus comandos de shell. AWS Proton Para fazer isso, AWS Proton cria um AWS CodeBuild projeto para o ambiente, na conta do ambiente, e inicia um trabalho para executar seus comandos para cada criação ou atualização de AWS Proton recursos. Ao criar um pacote de modelos, você fornece um manifesto que especifica os comandos de provisionamento e desprovisionamento da infraestrutura e quaisquer programas, scripts e outros arquivos que esses comandos possam precisar. Seus comandos podem ler as entradas que o AWS Proton fornece e são responsáveis por provisionar ou desprovisionar a infraestrutura e gerar valores de saída.

O manifesto também especifica como AWS Proton renderizar o arquivo de entrada que seu código pode inserir e do qual obter valores de entrada. Ele pode ser renderizado em JSON ou HCL. Para obter mais informações sobre os parâmetros de entrada, consulte the section called “CodeBuild parâmetros de provisionamento”. Para obter mais informações sobre arquivos manifesto, consulte the section called “Manifesto e conclusão”.

> **Note**
>
> Você pode usar o CodeBuild provisionamento com ambientes e serviços. No momento, você não pode provisionar componentes dessa forma.

Exemplo: usando o AWS CDK com CodeBuild provisionamento

Como exemplo do uso do CodeBuild provisionamento, você pode incluir um código que usa os AWS recursos AWS Cloud Development Kit (AWS CDK) para provisionar (implantar) e desprovisionar (destruir) e um manifesto que instala o CDK e executa seu código CDK.

As seções a seguir listam exemplos de arquivos que você pode incluir em um pacote de modelos CodeBuild de provisionamento que provisiona um ambiente usando o. AWS CDK

Manifesto

O arquivo de manifesto a seguir especifica o CodeBuild provisionamento e inclui os comandos necessários para instalar e usar o arquivo de saída AWS CDK, processar o arquivo e reportar as saídas para. AWS Proton

Example infrastructure/manifest.yaml

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Esquema

O arquivo de esquema a seguir define parâmetros para o ambiente. Seu AWS CDK código pode se referir aos valores desses parâmetros durante a implantação.

Example schema/schema.yaml

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "MyEnvironmentInputType"
  types:
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:
          type: string
          description: "Another sample input"
      required:
        - my_other_sample_input
```

AWS CDK arquivos

Os arquivos a seguir são um exemplo de um projeto CDK do Node.js.

Example infrastructure/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironmente": "bin/ProtonEnvironment.js"
  },
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk"
  },
  "devDependencies": {
    "@types/jest": "^28.1.7",
    "@types/node": "18.7.6",
    "jest": "^28.1.3",
```

```
    "ts-jest": "^28.0.8",
    "aws-cdk": "2.37.1",
    "ts-node": "^10.9.1",
    "typescript": "~4.7.4"
  },
  "dependencies": {
    "aws-cdk-lib": "2.37.1",
    "constructs": "^10.1.77",
    "source-map-support": "^0.5.21"
  }
}
```

Example infrastructure/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2018",
    "module": "commonjs",
    "lib": [
      "es2018"
    ],
    "declaration": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noImplicitThis": true,
    "alwaysStrict": true,
    "noUnusedLocals": false,
    "noUnusedParameters": false,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": false,
    "inlineSourceMap": true,
    "inlineSources": true,
    "experimentalDecorators": true,
    "strictPropertyInitialization": false,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "typeRoots": [
      "./node_modules/@types"
    ]
  },
  "exclude": [
    "node_modules",
```

```
    "cdk.out"
  ]
}
```

Example infrastructure/cdk.json

```
{
  "app": "npx ts-node --prefer-ts-exts bin/ProtonEnvironment.ts",
  "outputsFile": "proton-outputs.json",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
      "yarn.lock",
      "node_modules",
      "test"
    ]
  },
  "context": {
    "@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId": true,
    "@aws-cdk/core:stackRelativeExports": true,
    "@aws-cdk/aws-rds:lowercaseDbIdentifier": true,
    "@aws-cdk/aws-lambda:recognizeVersionProps": true,
    "@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2_2021": true,
    "@aws-cdk-containers/ecs-service-extensions:enableDefaultLogDriver": true,
    "@aws-cdk/aws-ec2:uniqueImdsv2TemplateName": true,
    "@aws-cdk/core:target-partitions": [
      "aws",
      "aws-cn"
    ]
  }
}
```

Example infrastructure/bin/ProtonEnvironment.ts

```
#!/usr/bin/env node
```

```
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { ProtonEnvironmentStack } from '../lib/ProtonEnvironmentStack';

const app = new cdk.App();
new ProtonEnvironmentStack(app, 'ProtonEnvironmentStack', {});
```

Example infrastructure/lib/ProtonEnvironmentStack.ts

```
import { Stack, StackProps } from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import input from '../proton-inputs.json';

export class ProtonEnvironmentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, { ...props, stackName: process.env.STACK_NAME });

    const ssmParam = new ssm.StringParameter(this, "ssmParam", {
      stringValue: input.environment.inputs.my_sample_input,
      parameterName: `${process.env.STACK_NAME}-Param`,
      tier: ssm.ParameterTier.STANDARD
    })

    new cdk.CfnOutput(this, 'ssmParamOutput', {
      value: ssmParam.parameterName,
      description: 'The name of the ssm parameter',
      exportName: `${process.env.STACK_NAME}-Param`
    });
  }
}
```

Arquivo de entrada renderizado

Quando você cria um ambiente usando um modelo de provisionamento CodeBuild baseado, AWS Proton renderiza um arquivo de entrada com os valores dos parâmetros de entrada fornecidos por você. Seu código pode se referir a esses valores. O arquivo a seguir é um exemplo de um arquivo de entrada renderizado.

Example infrastructure/proton-inputs.json

```
{
```

```
  "environment": {
    "name": "myenv",
    "inputs": {
      "my_sample_input": "10.0.0.0/16",
      "my_other_sample_input": "11.0.0.0/16"
    }
  }
}
```

Arquivos Terraform IaC

Aprenda a usar a infraestrutura do Terraform como arquivos de código (IaC) com. AWS ProtonO Terraform é um mecanismo IaC de código aberto amplamente usado que foi desenvolvido por. HashiCorp Os módulos do Terraform são desenvolvidos na linguagem HashiCorp HCL e oferecem suporte a vários provedores de infraestruturas de back-end, incluindo a Amazon Web Services.

AWS Proton oferece suporte ao provisionamento autogerenciado para o Terraform IaC.

Para obter um exemplo completo de um repositório de provisionamento que responde a pull requests e implementa o provisionamento de infraestrutura, consulte o modelo de automação do Terraform Actions para ver mais. OpenSource GitHub AWS Proton GitHub

Como o provisionamento autogerenciado funciona com os arquivos de pacote de modelos do Terraform IaC:

1. Ao criar um ambiente a partir de pacotes de modelos do Terraform, AWS Proton compila seus `.tf` arquivos com parâmetros de console ou `spec file` de entrada.
2. Ele faz uma solicitação pull para mesclar os arquivos IaC compilados no repositório que você registrou no AWS Proton.
3. Se a solicitação for aprovada, AWS Proton aguarda o status de provisionamento fornecido por você.
4. Se a solicitação for rejeitada, a criação do ambiente será cancelada.
5. Se o tempo limite do pull request expirar, a criação do ambiente não será concluída.

AWS Proton com as considerações do Terraform IaC:

- AWS Proton não gerencia seu provisionamento do Terraform.
- Você deve registrar um repositório de provisionamento com. AWS Proton AWS Proton faz pull requests nesse repositório.

- Você deve criar uma CodeStar conexão para se conectar AWS Proton ao seu repositório de provisionamento.
- Para provisionar a partir de arquivos IaC AWS Proton compilados, você deve responder às AWS Proton pull requests. AWS Proton faz pull requests após as ações de criação e atualização do ambiente e do serviço. Para obter mais informações, consulte AWS Proton ambientes e AWS Proton serviços.
- Para provisionar um pipeline a partir de arquivos IaC AWS Proton compilados, você deve criar um repositório de CI/CD pipeline.
- Sua automação de provisionamento baseada em pull request deve incluir etapas para notificar sobre qualquer alteração no AWS Proton status do recurso provisionado. AWS Proton Você pode usar a AWS Proton NotifyResourceDeploymentStatusChange API.
- Você não pode implantar serviços, pipelines e componentes criados a partir de arquivos CloudFormation IaC em ambientes criados a partir de arquivos do Terraform IaC.
- Você não pode implantar serviços, pipelines e componentes criados a partir de arquivos do Terraform IaC em ambientes criados a partir de CloudFormation arquivos IaC.

Ao preparar seus arquivos do Terraform IaC para AWS Proton, você anexa namespaces às suas variáveis de entrada, conforme mostrado nos exemplos a seguir. Para obter mais informações, consulte Parâmetros.

Exemplo 1: arquivo Terraform IaC do AWS Proton ambiente

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}
```

```
// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

Infraestrutura compilada como código

Quando você cria um ambiente ou serviço, AWS Proton compila sua infraestrutura como arquivos de código com console ou `spec file` entradas. Ele cria arquivos `proton.`*`resource-type`*`.variables.tf` e `proton.auto.tfvars.json` para suas entradas, os quais podem ser usados pelo Terraform, conforme mostrado nos exemplos a seguir. Esses arquivos estão localizados em um repositório especificado em uma pasta que corresponde ao nome do ambiente ou da instância do serviço.

O exemplo mostra como AWS Proton inclui tags na definição e nos valores das variáveis e como você pode propagar essas AWS Proton tags para recursos provisionados. Para obter mais informações, consulte the section called “Propagação de tags para recursos provisionados”.

Exemplo 2: arquivos IaC compilados para um ambiente chamado “dev”.

dev/environment.tf:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
```

```
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

dev/proton.environment.variables.tf:

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

dev/proton.auto.tfvars.json:

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
```

```
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

Caminhos do repositório

AWS Proton usa entradas de console ou especificação de ações de criação de ambiente ou serviço para encontrar o repositório e o caminho onde estão os arquivos IaC compilados. Os valores de entrada são passados para parâmetros de entrada com namespace.

AWS Proton suporta dois layouts de caminho de repositório. Nos exemplos a seguir, os caminhos são nomeados pelos parâmetros de recursos com namespace de dois ambientes. Cada ambiente tem instâncias de serviço de dois serviços, e as instâncias de serviço de um dos serviços têm componentes diretamente definidos.

Tipo de recurso	Parâmetro de nome	=	Nome do recurso
Ambiente	`environment.name`		`"env-prod"`
Ambiente	`environment.name`		`"env-staged"`
Serviço	`service.name`	=	`"service-one"`
Instância de serviço	`service_instance.name`		`"instance-one-prod"`
Instância de serviço	`service_instance.name`		`"instance-one-staged"`

Tipo de recurso	Parâmetro de nome	=	Nome do recurso
Serviço	`service.name`		`"service-two"`
Instância de serviço	`service_instance.name`		`"instance-two-prod"`
Componente	`service_instance.components.default.name`		`"component-prod"`
Instância de serviço	`service_instance.name`		`"instance-two-staged"`
Componente	`service_instance.components.default.name`		`"component-staged"`

Layout 1

Se AWS Proton encontrar o repositório especificado com uma `environments` pasta, ele cria uma pasta que inclui os arquivos IaC compilados e é nomeada com o. `environment.name`

Se AWS Proton encontrar o repositório especificado com uma `environments` pasta que contém um nome de pasta que corresponde ao nome de ambiente compatível com a instância de serviço, ele cria uma pasta que inclui os arquivos IaC da instância compilada e é nomeada com o. `service_instance.name`

```
/repo
    /environments
        /env-prod                               # environment folder
            main.tf
            proton.environment.variables.tf
            proton.auto.tfvars.json

            /service-one-instance-one-prod    # instance folder
```

```
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                   # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                           # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                 # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Layout 2

Se AWS Proton encontrar o repositório especificado sem uma `environments` pasta, ele cria uma `environment.name` pasta onde localiza os arquivos IaC do ambiente compilado.

Se AWS Proton encontrar o repositório especificado com um nome de pasta que corresponda ao nome do ambiente compatível com a instância de serviço, ele cria uma `service_instance.name` pasta na qual localiza os arquivos IaC da instância compilada.

```
/repo
    /env-prod                           # environment folder
        main.tf
        proton.environment.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                   # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                         # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                 # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Arquivo de esquema

Como administrador, quando você usa a seção Modelos de dados de API aberta (esquemas) para definir um arquivo YAML de esquema de parâmetros para seu pacote de modelos, AWS Proton pode validar as entradas de valores de parâmetros em relação aos requisitos definidos em seu esquema.

Para obter mais informações sobre formatos e palavras-chave disponíveis, consulte a seção Objeto de esquema da OpenAPI.

Requisitos de esquema para pacotes de modelos de ambiente

Seu esquema deve seguir a seção Modelos de dados (esquemas) da OpenAPI no formato YAML. Ele também deve fazer parte do seu pacote de modelos de ambiente.

Para o esquema do seu ambiente, você deve incluir os cabeçalhos formatados para estabelecer que está usando a seção Modelos de dados (esquemas) da API aberta. Nos exemplos de esquema de ambiente a seguir, esses cabeçalhos aparecem nas três primeiras linhas.

Um `environment_input_type` deve ser incluído e definido com um nome fornecido por você. Nos exemplos a seguir, isso é definido na linha 5. Ao definir esse parâmetro, você o associa a um recurso AWS Proton do ambiente.

Para seguir o modelo de esquema da Open API, você deve incluir `types`. No exemplo a seguir, isso está na linha 6.

Após `types`, você deve definir um tipo de `environment_input_type`. Você define os parâmetros de entrada para seu ambiente como propriedades do `environment_input_type`. Você deve incluir pelo menos uma propriedade com um nome que corresponda a pelo menos um parâmetro listado no arquivo do ambiente de infraestrutura como código (IaC) que está associado ao esquema.

Quando você cria um ambiente e fornece valores de parâmetros personalizados, AWS Proton usa o arquivo de esquema para corresponder, validar e injetá-los nos parâmetros entre colchetes no arquivo IaC associado. CloudFormation Para cada propriedade (parâmetro), forneça um `name` e `type`. Opcionalmente, forneça também um `description`, `default`, e. `pattern`

Os parâmetros definidos para o exemplo de esquema de modelo de ambiente padrão a seguir incluem `vpc_cidr`, `subnet_one_cidr` e `subnet_two_cidr` com a palavra-chave `default` e os valores padrão. Ao criar um ambiente com esse esquema de pacote de modelo de ambiente, você pode aceitar os valores padrão ou fornecer os seus próprios. Se um parâmetro não tiver um valor

padrão e estiver listado como uma propriedade (parâmetro) `required`, você deverá fornecer valores para ele ao criar um ambiente.

O segundo exemplo de esquema de modelo de ambiente padrão lista o parâmetro `required` `my_other_sample_input`.

Você pode criar um esquema para dois tipos de modelos de ambiente. Para obter mais informações, consulte Registre e publique modelos.

- Modelos de ambiente padrão

  No exemplo a seguir, um tipo de entrada de ambiente é definido com uma descrição e propriedades de entrada. Esse exemplo de esquema pode ser usado com o arquivo AWS Proton CloudFormation IaC mostrado no Exemplo 3.

  Exemplo de esquema para um modelo de ambiente padrão:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "PublicEnvironmentInput"
  types:                          # required
    # defined by administrator
    PublicEnvironmentInput:
      type: object
      description: "Input properties for my environment"
      properties:
        vpc_cidr:                  # parameter
          type: string
          description: "This CIDR range for your VPC"
          default: 10.0.0.0/16
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_one_cidr:           # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.0.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_two_cidr:           # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.1.0/24
```

```
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
```

Exemplo de esquema para um modelo de ambiente padrão que inclui um parâmetro de `required`:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "MyEnvironmentInputType"
  types:                          # required
    # defined by administrator
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:          # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:    # parameter
          type: string
          description: "Another sample input"
        another_optional_input:   # parameter
          type: string
          description: "Another optional input"
          default: "!"
      required:
        - my_other_sample_input
```

- Modelos de ambiente gerenciado pelo cliente

  No exemplo a seguir, o esquema inclui apenas uma lista de saídas que replicam as saídas do IaC que você usou para provisionar sua infraestrutura gerenciada pelo cliente. Você precisa definir os tipos de valores de saída somente como strings (não listas, matrizes ou outros tipos). Por exemplo, o próximo trecho de código mostra a seção de saídas de um modelo externo do CloudFormation . Isso é do modelo mostrado no Exemplo 1. Ele pode ser usado para criar uma infraestrutura externa gerenciada pelo cliente para um serviço AWS Proton Fargate criado a partir do Exemplo 4.

> **Important**
>
> Como administrador, você deve garantir que sua infraestrutura provisionada e gerenciada e todos os parâmetros de saída sejam compatíveis com os modelos de ambiente gerenciado pelo cliente associados. AWS Proton não pode contabilizar as alterações em seu nome porque essas alterações não estão visíveis para AWS Proton. As inconsistências resultam em falhas.

Exemplo de CloudFormation saídas de arquivo IaC para um modelo de ambiente gerenciado pelo cliente:

```
// Cloudformation Template Outputs
[...]
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

O esquema do pacote de modelos de ambiente gerenciado pelo AWS Proton cliente correspondente é mostrado no exemplo a seguir. Cada valor de saída é definido como uma string.

Exemplo de esquema para um modelo de ambiente gerenciado pelo cliente:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "EnvironmentOutput"
  types:                          # required
    # defined by administrator
    EnvironmentOutput:
      type: object
```

```
      description: "Outputs of the environment"
      properties:
        ClusterName:               # parameter
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:      # parameter
          type: string
          description: "The ARN of the ECS role"
        VpcId:                     # parameter
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Requisitos de esquema para pacotes de modelos de serviço

Seu esquema deve seguir a seção Modelos de dados (esquemas) da OpenAPI no formato YAML como mostrado nos exemplos seguintes. Você deve fornecer um arquivo de esquema em seu pacote de modelos de serviço.

Nos exemplos de esquema de serviço a seguir, você deve incluir os cabeçalhos formatados. No exemplo a seguir, isso está nas três primeiras linhas. Isso serve para estabelecer que você está usando a seção Modelos de dados (esquemas) da API aberta.

Um `service_input_type` deve ser incluído e definido com um nome fornecido por você. No exemplo a seguir, isso está na linha 5. Isso associa os parâmetros a um recurso AWS Proton de serviço.

Um pipeline AWS Proton de serviço é incluído por padrão quando você usa o console ou a CLI para criar um serviço. Ao incluir um pipeline de serviços para seu serviço, você deve incluir a `pipeline_input_type` com um nome fornecido por você. No exemplo a seguir, isso está na linha 7. Não inclua esse parâmetro se você não estiver incluindo um pipeline AWS Proton de serviços. Para obter mais informações, consulte Registre e publique modelos.

Para seguir o modelo de esquema da API aberta, você deve incluir `types`. No exemplo a seguir, isso está na linha 9.

Após `types`, você deve definir um tipo de `service_input_type`. Você define os parâmetros de entrada para seu serviço como propriedades do `service_input_type`. Você deve incluir pelo menos uma propriedade com um nome que corresponda a pelo menos um parâmetro listado no arquivo do serviço de infraestrutura como código (IaC) que está associado ao esquema.

Para definir um pipeline de serviços, abaixo da sua definição de `service_input_type`, você deve definir um `pipeline_input_type`. Como acima, você deve incluir pelo menos uma propriedade com um nome que corresponda a pelo menos um parâmetro listado em um arquivo de pipeline que está associado ao esquema. Não inclua essa definição se você não estiver incluindo um pipeline AWS Proton de serviços.

Quando você, como administrador ou desenvolvedor, cria um serviço e fornece valores de parâmetros personalizados, AWS Proton usa o arquivo de esquema para corresponder, validar e injetá-los nos parâmetros entre chaves curvas do arquivo CloudFormation IaC associado. Para cada propriedade (parâmetro), forneça um `name` e um `type`. Opcionalmente, forneça também um `description`, `default` e. `pattern`

Os parâmetros definidos para o exemplo de esquema incluem `port`, `desired_count`, `task_size` e `image` com os valores de palavra-chave e padrão `default`. Ao criar um serviço com esse esquema de pacote de modelo de serviço, você pode aceitar os valores padrão ou fornecer os seus próprios. O parâmetro `unique_name` também está incluído no exemplo e não tem um valor padrão. Ela é listada como uma propriedade (parâmetro) do `required`. Você, como administrador ou desenvolvedor, deve fornecer valores para os parâmetros do `required` ao criar serviços.

Se você quiser criar um modelo de serviço com um pipeline de serviço, inclua o `pipeline_input_type` em seu esquema.

Exemplo de arquivo de esquema de serviço para um serviço que inclui um pipeline AWS Proton de serviços.

Esse exemplo de esquema pode ser usado com os arquivos AWS Proton IaC mostrados no Exemplo 4 e no Exemplo 5. Um pipeline de serviços está incluído.

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required           defined by administrator
  service_input_type: "LoadBalancedServiceInput"
  # only include if including AWS Proton service pipeline, defined by administrator
  pipeline_input_type: "PipelineInputs"

  types:                          # required
    # defined by administrator
    LoadBalancedServiceInput:
      type: object
      description: "Input properties for a loadbalanced Fargate service"
```

```
      properties:
        port:                       # parameter
          type: number
          description: "The port to route traffic to"
          default: 80
          minimum: 0
          maximum: 65535
        desired_count:              # parameter
          type: number
          description: "The default number of Fargate tasks you want running"
          default: 1
          minimum: 1
        task_size:                  # parameter
          type: string
          description: "The size of the task you want to run"
          enum: ["x-small", "small", "medium", "large", "x-large"]
          default: "x-small"
        image:                      # parameter
          type: string
          description: "The name/url of the container image"
          default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
          minLength: 1
          maxLength: 200
        unique_name:                # parameter
          type: string
          description: "The unique name of your service identifier. This will be used
to name your log group, task definition and ECS service"
          minLength: 1
          maxLength: 100
      required:
        - unique_name
    # defined by administrator
    PipelineInputs:
      type: object
      description: "Pipeline input properties"
      properties:
        dockerfile:                 # parameter
          type: string
          description: "The location of the Dockerfile to build"
          default: "Dockerfile"
          minLength: 1
          maxLength: 100
        unit_test_command:          # parameter
          type: string
```

```
          description: "The command to run to unit test the application code"
          default: "echo 'add your unit test command here'"
          minLength: 1
          maxLength: 200
```

Se você quiser criar um modelo de serviço sem um pipeline de serviço, não inclua o `pipeline_input_type` em seu esquema, conforme demonstrado na página seguinte.

Exemplo de arquivo de esquema de serviço para um serviço que não inclui um pipeline AWS Proton de serviços

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required            defined by administrator
  service_input_type: "MyServiceInstanceInputType"

  types:                          # required
    # defined by administrator
    MyServiceInstanceInputType:
      type: object
      description: "Service instance input properties"
      required:
        - my_sample_service_instance_required_input
      properties:
        my_sample_service_instance_optional_input:   # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_sample_service_instance_required_input:   # parameter
          type: string
          description: "Another sample input"
```

Encapsular arquivos de modelo para AWS Proton

Após preparar seus arquivos de infraestrutura como código (IaC) do ambiente e do serviço e seus respectivos arquivos de esquema, você deve organizá-los em diretórios.. Você também deve criar um arquivo YAML de manifesto. O arquivo de manifesto lista os arquivos IaC em um diretório, o mecanismo de renderização e a linguagem de modelo usada para desenvolver o IaC nesse modelo.

> **Note**
>
> Um arquivo de manifesto também pode ser usado independentemente dos pacotes de modelos, como uma entrada direta para componentes definidos diretamente. Nesse caso, ele sempre especifica um único arquivo de modelo IaC, para ambos CloudFormation e para o Terraform. Para obter mais informações sobre componentes, consulte Componentes.

O arquivo de manifesto precisa seguir o formato e o conteúdo mostrados no exemplo a seguir.

CloudFormation formato de arquivo de manifesto:

Com CloudFormation, você lista um único arquivo.

```
infrastructure:
  templates:
    - file: "cloudformation.yaml"
      rendering_engine: jinja
      template_language: cloudformation
```

Formato de arquivo de manifesto do Terraform:

Com o terraform, você pode listar explicitamente um único arquivo ou usar o caractere curinga * para listar cada um dos arquivos em um diretório.

> **Note**
>
> O caractere curinga inclui apenas arquivos cujos nomes terminam em `.tf`. Outros arquivos são ignorados.

```
infrastructure:
  templates:
    - file: "*"
      rendering_engine: hcl
      template_language: terraform
```

CodeBuildformato de arquivo de manifesto de provisionamento baseado em:

Com o provisionamento CodeBuild baseado, você especifica os comandos shell de provisionamento e desprovisionamento.

> **Note**
>
> Além do manifesto, seu pacote deve incluir todos os arquivos dos quais seus comandos dependam.

O exemplo de manifesto a seguir usa provisionamento CodeBuild baseado para provisionar (implantar) e desprovisionar (destruir) recursos usando o AWS Cloud Development Kit (AWS CDK) ().AWS CDK O pacote de modelos também deve incluir o código CDK.

Durante o provisionamento, o AWS Proton cria um arquivo de entrada com valores para os parâmetros de entrada que você definiu no esquema do modelo com o nome `proton-input.json`.

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
```

```
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Depois de configurar os diretórios e os arquivos de manifesto para seu ambiente ou pacote de modelos de serviços, você compacta os diretórios em uma bola tar e os carrega em um bucket do Amazon Simple Storage Service (Amazon S3), AWS Proton onde pode recuperá-los, ou em um repositório Git de sincronização de modelos.

Ao criar uma versão secundária de um ambiente ou modelo de serviço com o qual você se registrou AWS Proton, você fornece o caminho para seu ambiente ou pacote de modelos de serviço tar ball que está localizado em seu bucket do S3. AWS Proton salva com a nova versão secundária do modelo. Você pode selecionar a nova versão secundária do modelo para criar ou atualizar ambientes ou serviços AWS Proton.

Conclusão do pacote de modelos de ambiente

Há dois tipos de pacotes de modelos de ambiente para AWS Proton os quais você cria.

- Para criar um pacote de modelo de ambiente para um modelo de ambiente padrão, organize o esquema, os arquivos de infraestrutura como código (IaC) e o arquivo de manifesto em diretórios, conforme mostrado na estrutura de diretórios do pacote de modelos de ambiente a seguir.
- Para criar um pacote de modelos de ambiente para um modelo de ambiente gerenciado pelo cliente, forneça somente o arquivo e o diretório do esquema. Não inclua o diretório e os arquivos da infraestrutura. AWS Proton gerará um erro se o diretório e os arquivos da infraestrutura estiverem incluídos.

Para obter mais informações, consulte Registre e publique modelos.

CloudFormation estrutura de diretório do pacote do modelo de ambiente:

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  cloudformation.yaml
```

Estrutura de diretórios do pacote do modelo de ambiente Terraform:

```
/schema
  schema.yaml
```

```
/infrastructure
  manifest.yaml
  environment.tf
```

Conclusão do pacote de modelos de serviços

Para criar um pacote de modelo de serviço, você deve organizar o esquema, os arquivos de infraestrutura como código (IaC) e os arquivos de manifesto em diretórios, conforme mostrado no exemplo da estrutura de diretórios do pacote de modelos de serviços.

Se você não incluir um pipeline de serviço em seu pacote de modelos, não inclua o diretório e os arquivos do pipeline e defina `"pipelineProvisioning": "CUSTOMER_MANAGED"` ao criar o modelo de serviço que será associado a esse pacote de modelos.

> **Note**
>
> Não é possível modificar o `pipelineProvisioning` após o modelo de serviço ser criado.

Para obter mais informações, consulte Registre e publique modelos.

CloudFormation estrutura de diretórios do pacote de modelos de serviços:

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  cloudformation.yaml
/pipeline_infrastructure
  manifest.yaml
  cloudformation.yaml
```

Estrutura de diretórios do pacote do modelo de serviço Terraform:

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  instance.tf
/pipeline_infrastructure
  manifest.yaml
```

```
    pipeline.tf
```

Considerações sobre o pacote de modelos

- Arquivos de infraestrutura como código (IaC).

  AWS Proton audita os modelos para o formato de arquivo correto. No entanto, AWS Proton não verifica erros de desenvolvimento, dependência e lógica do modelo. Por exemplo, suponha que você especificou a criação de um bucket Amazon S3 em seu arquivo CloudFormation IaC como parte do seu modelo de serviço ou ambiente. Um serviço é criado com base nesses modelos. Agora, suponha que em algum momento você queira excluir o serviço. Se o bucket do S3 especificado não estiver vazio e o arquivo CloudFormation IaC não o marcar como `Retain` no`DeletionPolicy`, haverá AWS Proton falha na operação de exclusão do serviço.
- Limites e formato de tamanho de arquivo do pacote
  - Os limites de tamanho, contagem e tamanho do nome do arquivo do pacote podem ser encontrados em AWS Proton cotas.
  - Os diretórios de arquivos do pacote de modelos são compactados em um tarball e localizados em um bucket do Amazon Simple Storage Service (Amazon S3).
  - Cada arquivo no pacote deve ser um arquivo YAML formatado válido.
- Criptografia do pacote de modelos de bucket do S3

  Se você quiser criptografar dados confidenciais em seus pacotes de modelos em repouso no bucket do S3, use as chaves SSE-S3 ou SSE-KMS para permitir sua recuperação. AWS Proton

AWS Proton modelos

Para adicionar seu pacote de modelos à sua biblioteca AWS Proton de modelos, crie uma versão secundária de modelos e registre-a com AWS Proton. Ao criar o modelo, forneça o nome do bucket do Amazon S3 e o caminho para o pacote de modelo. Depois que os modelos são publicados, eles podem ser selecionados pelos membros da equipe da plataforma e pelos desenvolvedores. Depois de selecionados, AWS Proton usa o modelo para criar e provisionar infraestrutura e aplicativos.

Como administrador, você pode criar e registrar um modelo de ambiente com AWS Proton. Esse modelo de ambiente pode então ser usado para implantar vários ambientes. Por exemplo, ele pode ser usado para implantar ambientes de “desenvolvimento”, “teste” e “produção”. O ambiente “dev” (ambiente de desenvolvimento) pode incluir uma VPC com sub-redes privadas e uma política de acesso restritiva a todos os recursos. As saídas do ambiente podem ser usadas como entradas para serviços.

Você pode criar e registrar modelos de ambiente para criar dois tipos diferentes de ambientes. Você e os desenvolvedores podem usar AWS Proton para implantar serviços em ambos os tipos.

- Registre e publique um modelo de ambiente padrão AWS Proton usado para criar um ambiente padrão que provisione e gerencie a infraestrutura do ambiente.
- Registre e publique um modelo de ambiente gerenciado pelo cliente que AWS Proton use para criar um ambiente gerenciado pelo cliente que se conecte à sua infraestrutura provisionada existente. AWS Proton não gerencia sua infraestrutura provisionada existente.

Você pode criar e registrar modelos de serviço AWS Proton para implantar serviços em ambientes. Um AWS Proton ambiente deve ser criado antes que um serviço possa ser implantado nele.

A lista a seguir descreve como você cria e gerencia modelos com AWS Proton.

- (Opcional) Prepare uma função do IAM para controlar o acesso do desenvolvedor às chamadas de AWS Proton API e às funções de serviço AWS Proton do IAM. Para obter mais informações, consulte the section called “Perfis do IAM”.
- Compor um pacote de modelos. Para obter mais informações, consulte Pacotes de modelos.
- Crie e registre um modelo AWS Proton depois que o pacote de modelos for composto, compactado e salvo em um bucket do Amazon S3. Você pode fazer isso no console ou usando o AWS CLI.

- Teste e use o modelo para criar e gerenciar recursos AWS Proton provisionados após seu registro no. AWS Proton
- Crie e gerencie versões principais e secundárias do modelo durante toda a vida útil do modelo.

Você pode gerenciar versões de modelos manualmente ou com configurações de sincronização de modelos:

- Use o AWS Proton console e AWS CLI crie uma nova versão secundária ou principal.
- Crie uma configuração de sincronização de modelos que permita criar AWS Proton automaticamente uma nova versão secundária ou principal ao detectar uma alteração no seu pacote de modelos em um repositório definido por você.

Para obter informações adicionais, consulte The AWS Proton Service API Reference.

Tópicos

Modelos versionados

Como administrador ou membro de uma equipe da plataforma, você define, cria e gerencia uma biblioteca de modelos versionados que são usados para provisionar recursos de infraestrutura. Há dois tipos de versões de modelo: versões secundárias e versões principais.

- Versões secundárias: Alterações no modelo que têm um esquema compatível com versões anteriores. Essas alterações não exigem que o desenvolvedor forneça novas informações ao atualizar para a nova versão do modelo.

  Ao tentar fazer uma pequena alteração na AWS Proton versão, tente ao máximo determinar se o esquema da nova versão é compatível com versões anteriores do modelo. Se o novo esquema

não for compatível com versões anteriores, o registro da nova versão secundária AWS Proton falhará.

> **Note**
>
> A compatibilidade é determinada exclusivamente com base no esquema. AWS Proton não verifica se o arquivo de infraestrutura como código (IaC) do pacote de modelos é compatível com versões anteriores secundárias. Por exemplo, AWS Proton não verifica se o novo arquivo IaC causa alterações significativas nos aplicativos que estão sendo executados na infraestrutura provisionada por uma versão secundária anterior do modelo.

- Versões principais: Alterações no modelo que podem não ser compatíveis com versões anteriores. Essas mudanças geralmente exigem novas informações do desenvolvedor e geralmente envolvem mudanças no esquema do modelo.

  Às vezes, você pode optar por designar uma alteração compatível com versões anteriores como uma versão principal com base no modelo operacional da sua equipe.

A forma como AWS Proton determina se uma solicitação de versão do modelo é para uma versão secundária ou principal depende da forma como as alterações do modelo são rastreadas:

- Quando você faz uma solicitação explícita para criar uma nova versão de modelo, solicita uma versão principal especificando um número de versão principal e solicita uma versão secundária sem especificar um número de versão principal.
- Quando você usa a sincronização de modelos (e, portanto, não faz solicitações explícitas de versão de modelo), AWS Proton tenta criar novas versões secundárias para alterações de modelo que ocorrem no arquivo YAML existente. AWS Proton cria uma versão principal quando você cria um novo diretório para a nova alteração do modelo (por exemplo, mover da v1 para a v2).

> **Note**
>
> O registro de uma nova versão secundária com base na sincronização de modelos ainda falhará se AWS Proton determinar que a alteração não é compatível com versões anteriores.

Quando você publica uma nova versão de um modelo, ela se torna a versão recomendada se for a versão principal e secundária mais alta. Novos AWS Proton recursos são criados usando a nova versão recomendada e AWS Proton solicita que os administradores usem a nova versão e atualizem os AWS Proton recursos existentes que estão usando uma versão desatualizada.

Registre e publique modelos

Você pode registrar e publicar modelos de ambiente e serviço com AWS Proton, conforme descrito nas seções a seguir.

Você pode criar uma nova versão de um modelo com o console ou AWS CLI.

Como alternativa, você pode usar o console ou AWS CLI criar um modelo e configurar uma sincronização de modelo para ele. Essa configuração permite a AWS Proton sincronização a partir de pacotes de modelos localizados em repositórios git registrados que você definiu. Sempre que um commit é enviado para seu repositório e isso muda um de seus pacotes de modelos, uma nova versão principal ou secundária de seu modelo é criada, se essa versão já não existir. Para saber mais sobre pré-requisitos e requisitos de configuração de sincronização de modelos, consulte Configurações de sincronização de modelo.

Registrar e publicar modelos de ambiente

Você pode registrar e publicar os seguintes tipos de modelos de ambiente.

- Registre e publique um modelo de ambiente padrão AWS Proton usado para implantar e gerenciar a infraestrutura do ambiente.
- Registre e publique um modelo de ambiente gerenciado pelo cliente que AWS Proton use para se conectar à sua infraestrutura provisionada existente que você gerencia. AWS Proton não gerencia sua infraestrutura provisionada existente.

> **Important**
>
> Como administrador, garanta que sua infraestrutura provisionada e gerenciada e todos os parâmetros de saída sejam compatíveis com os modelos de ambiente gerenciado pelo cliente associados. AWS Proton não pode contabilizar as alterações em seu nome porque essas alterações não estão visíveis para AWS Proton. As inconsistências resultam em falhas.

Você pode usar o console ou o AWS CLI para registrar e publicar um modelo de ambiente.

Console de gerenciamento da AWS

Use o console para registrar e publicar um modelo de ambiente novo.

1. No console do AWS Proton, escolha Modelos de ambiente.
2. Selecione Criar modelo de ambiente.
3. Na página Criar modelo de ambiente, na seção Opções de modelo, escolha uma das duas opções de modelo disponíveis.
   - Criar um modelo para provisionar novos ambientes.
   - Criar um modelo para usar a infraestrutura provisionada que você gerencia.
4. Se você escolher Criar um modelo para provisionar novos ambientes, na seção Fonte do pacote de modelos, escolha uma das três opções de origem do pacote de modelos disponíveis. Para saber mais sobre requisitos e pré-requisitos de sincronização de modelos, consulte Configurações de sincronização de modelo.
   - Usar um dos nossos pacotes de modelos de amostra.
   - Usar seu próprio pacote de modelos.
   - Sincronizar modelos do Git.
5. Fornecer um caminho para um pacote de modelos.
   a. Se você escolheu Usar um dos nossos pacotes de modelos de amostra:

      Na seção Exemplo de pacote de modelos, selecione um pacote de modelos de amostra.
   b. Se você escolheu Sincronizar modelos do Git, na seção Código-fonte:
      i. Selecione o repositório para sua configuração de sincronização de modelos.
      ii. Insira o nome da ramificação do repositório a partir da qual sincronizar.
      iii. (Opcional) Insira o nome de um diretório para limitar a pesquisa do seu pacote de modelos.
   c. Caso contrário, na seção de localização do pacote S3, forneça um caminho para seu pacote de modelos.
6. Na seção Detalhes do modelo.
   a. Insira um nome de modelo.

b. (Opcional) Insira um Nome para exibição do modelo.

c. (Opcional) Insira uma Descrição de modelo para o modelo de ambiente.

7. (Opcional) Marque a caixa de seleção Personalizar configurações de criptografia (avançado) na seção Configurações de criptografia para fornecer sua própria chave de criptografia.
8. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
9. Escolha Criar modelo de ambiente.

   Agora você está em uma nova página que exibe o status e os detalhes do seu novo modelo de ambiente. Esses detalhes incluem uma lista de AWS tags gerenciadas pelo cliente. AWS Proton gera automaticamente tags AWS gerenciadas para você quando você cria AWS Proton recursos. Para obter mais informações, consulte AWS Proton recursos e marcação.
10. O status de um novo status de modelo de ambiente começa no estado Rascunho. Você e outras pessoas com permissões de `proton:CreateEnvironment` podem visualizá-lo e acessá-lo. Siga a próxima etapa para disponibilizar o modelo para outras pessoas.
11. Na seção Versões do modelo, escolha o botão de rádio à esquerda da versão secundária do modelo que você acabou de criar (1.0). Como alternativa, você pode escolher Publicar no alerta de informações e pular a próxima etapa.
12. Na seção Versões do modelo, escolha Publicar.
13. O status do modelo muda para Publicado. Por ser a versão mais recente do modelo, é a versão recomendada.
14. No painel de navegação, selecione Modelos de ambiente para ver uma lista dos modelos e detalhes do seu ambiente.

Use o console para registrar novas versões principais e secundárias de um modelo de ambiente.

Para obter mais informações, consulte Modelos versionados.

1. No console do AWS Proton, escolha Modelos de ambiente.
2. Na lista de modelos de ambiente, escolha o nome do modelo de ambiente para o qual você deseja criar uma versão principal ou secundária.
3. Na visualização de detalhes do modelo de ambiente, escolha Criar nova versão na seção Versões do modelo.
4. Na página Criar uma nova versão de modelo de ambiente, na seção Origem do pacote de modelos, escolha uma das duas opções de origem de pacote de modelo disponíveis.

- Usar um dos nossos pacotes de modelos de amostra.
- Usar seu próprio pacote de modelos.

5. Fornecer um caminho para o pacote de modelos selecionado.
   - Se você escolheu Usar um dos nossos pacotes de modelos de amostra, na seção Pacote de modelos de amostra, selecione um pacote de modelos de amostra.
   - Se você escolheu Usar seu próprio pacote de modelos, na seção Localização do pacote S3, escolha o caminho para seu pacote de modelos.
6. Na seção Detalhes do modelo.
   a. (Opcional) Insira um Nome para exibição do modelo.
   b. (Opcional) Insira uma Descrição de modelo para o modelo de serviço.
7. Na seção Detalhes de modelo, escolha uma das seguintes opções.
   - Para criar uma versão secundária, mantenha a caixa de seleção Marcar para criar uma nova versão principal vazia.
   - Para criar uma versão principal, marque a caixa de seleção Marcar para criar uma nova versão principal.
8. Continue com as etapas do console para criar a nova versão secundária ou principal e escolha Criar nova versão.

AWS CLI

Use a CLI para registrar e publicar um novo modelo de ambiente, conforme mostrado nas etapas a seguir.

1. Crie um modelo padrão de ambiente OU um gerenciado pelo cliente especificando a região, o nome, o nome de exibição (opcional) e a descrição (opcional).
   a. Criar um modelo de ambiente padrão.

      Execute o seguinte comando:

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
```

```
    --description "VPC with public access"
```

Resposta:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env"
    }
}
```

b. Crie um modelo de ambiente gerenciado pelo cliente adicionando o parâmetro `provisioning` com o valor `CUSTOMER_MANAGED`.

Execute o seguinte comando:

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access" \
    --provisioning "CUSTOMER_MANAGED"
```

Resposta:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env",
        "provisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Crie uma versão secundária 0 da versão principal 1 do modelo de ambiente

   Essa e as etapas restantes são as mesmas para os modelos de ambiente padrão e gerenciado pelo cliente.

   Inclua o nome do modelo, a versão principal e o nome e a chave do bucket do S3 para o bucket que contém seu pacote de modelos de ambiente.

   Execute o seguinte comando:

   ```
   $ aws proton create-environment-template-version \
       --template-name "simple-env" \
       --description "Version 1" \
       --source s3="{bucket=your_s3_bucket, key=your_s3_key}"
   ```

   Resposta:

   ```
   {
       "environmentTemplateVersion": {
           "arn": "arn:aws:proton:region-id:123456789012:environment-template/
   simple-env:1.0",
           "createdAt": "2020-11-11T23:02:47.763000+00:00",
           "description": "Version 1",
           "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
           "majorVersion": "1",
           "minorVersion": "0",
           "status": "REGISTRATION_IN_PROGRESS",
           "templateName": "simple-env"
       }
   }
   ```

3. Use o comando get para verificar o status dos cadastros.

   Execute o seguinte comando:

   ```
   $ aws proton get-environment-template-version \
       --template-name "simple-env" \
       --major-version "1" \
       --minor-version "0"
   ```

   Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

4. Publique a versão secundária 0 da versão principal 1 do modelo de ambiente fornecendo o nome do modelo e a versão principal e secundária. Esta versão é a versão `Recommended`.

   Execute o seguinte comando:

```
$ aws proton update-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Resposta:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
```

```
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```


Depois de criar um novo modelo usando o AWS CLI, você pode ver uma lista de AWS etiquetas gerenciadas pelo cliente. AWS Proton gera automaticamente tags AWS gerenciadas para você. Você também pode modificar e criar etiquetas gerenciadas pelo cliente usando o AWS CLI. Para obter mais informações, consulte AWS Proton recursos e marcação.

Execute o seguinte comando:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment-
template/simple-env"
```

Registrar e publicar modelos de serviço

Ao criar uma versão de modelo de serviço, você especifica uma lista de modelos de ambiente compatíveis. Dessa forma, quando os desenvolvedores selecionam um modelo de serviço, eles têm opções para o ambiente em que implantar o serviço.

Antes de criar um serviço a partir de um modelo de serviço ou antes de publicar um modelo de serviço, confirme se os ambientes foram implantados a partir dos modelos de ambiente compatíveis listados.

Você não pode atualizar um serviço para a nova versão principal se ele for implantado em um ambiente criado a partir de um modelo de ambiente compatível removido.

Para adicionar ou remover modelos de ambiente compatíveis para uma versão de modelo de serviço, você cria uma nova versão principal dela.

Você pode usar o console ou o AWS CLI para registrar e publicar um modelo de serviço.

Console de gerenciamento da AWS

Use o console para registrar e publicar um novo modelo de serviço.

1. No console do AWS Proton, escolha Modelos de serviço.
2. Selecione Criar modelo de serviço.
3. Na página Criar modelo de serviço, na seção Origem do pacote de modelos, escolha uma das opções de modelo disponíveis.
   - Usar seu próprio pacote de modelos.
   - Sincronizar modelos do Git.
4. Fornecer um caminho para um pacote de modelos.
   a. Se você escolheu Sincronizar modelos do Git, na seção Repositório do código-fonte:
      i. Selecione o repositório para sua configuração de sincronização de modelos.
      ii. Insira o nome da ramificação do repositório a partir da qual sincronizar.
      iii. (Opcional) Insira o nome de um diretório para limitar a pesquisa do seu pacote de modelos.
   b. Caso contrário, na seção de localização do pacote S3, forneça um caminho para seu pacote de modelos.
5. Na seção Detalhes do modelo.
   a. Insira um nome de modelo.
   b. (Opcional) Insira um Nome para exibição do modelo.
   c. (Opcional) Insira uma Descrição de modelo para o modelo de serviço.
6. Na seção Modelos de ambiente compatíveis, escolha de uma lista de modelos de ambiente compatíveis.

7. (Opcional) Na seção Configurações de criptografia, escolha Personalizar configurações de criptografia (avançado) para fornecer sua própria chave de criptografia.
8. (Opcional) Na seção Pipeline:

   Se você não estiver incluindo uma definição de pipeline de serviço em seu modelo de serviço, desmarque a caixa de seleção Pipeline - opcional na parte inferior da página. Não é possível alterar essa configuração após a criação do modelo de serviço. Para obter mais informações, consulte Pacotes de modelos.
9. (Opcional) Na seção Fontes de componentes compatíveis, em Fontes de componentes, escolha Definido diretamente para permitir a anexação de componentes definidos diretamente às suas instâncias de serviço.
10. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
11. Selecione Criar um modelo de serviço.

    Agora você está em uma nova página que exibe o status e os detalhes do seu novo modelo de serviço. Esses detalhes incluem uma lista de AWS tags gerenciadas pelo cliente. AWS Proton gera automaticamente tags AWS gerenciadas para você quando você cria AWS Proton recursos. Para obter mais informações, consulte AWS Proton recursos e marcação.
12. O status de um novo status de modelo de serviço começa no estado Rascunho. Você e outras pessoas com permissões de `proton:CreateService` podem visualizá-lo e acessá-lo. Siga a próxima etapa para disponibilizar o modelo para outras pessoas.
13. Na seção Versões do modelo, escolha o botão de rádio à esquerda da versão secundária do modelo que você acabou de criar (1.0). Como alternativa, você pode escolher Publicar no alerta de informações e pular a próxima etapa.
14. Na seção Versões do modelo, escolha Publicar.
15. O status do modelo muda para Publicado. Por ser a versão mais recente do modelo, é a versão recomendada.
16. No painel de navegação, selecione Modelos de serviço para ver uma lista dos modelos e detalhes do seu serviço.

Use o console para registrar novas versões principais e secundárias de um modelo de serviço.

Para obter mais informações, consulte Modelos versionados.

1. No console do AWS Proton, escolha Modelos de serviço.

2. Na lista de modelos de serviço, escolha o nome do modelo de serviço para o qual você deseja criar uma versão principal ou secundária.
3. Na visualização de detalhes do modelo de serviço, escolha Criar nova versão na seção Versões do modelo.
4. Na página Criar uma nova versão do modelo de serviço, na seção Origem do pacote, selecione Usar seu próprio pacote de modelos.
5. Na seção de localização do pacote S3, escolha um caminho para seu pacote de modelos.
6. Na seção Detalhes do modelo.
   a. (Opcional) Insira um Nome para exibição do modelo.
   b. (Opcional) Insira uma Descrição de modelo para o modelo de serviço.
7. Na seção Detalhes de modelo, escolha uma das seguintes opções.
   - Para criar uma versão secundária, mantenha a caixa de seleção Marcar para criar uma nova versão principal vazia.
   - Para criar uma versão principal, marque a caixa de seleção Marcar para criar uma nova versão principal.
8. Continue com as etapas do console para criar a nova versão secundária ou principal e escolha Criar nova versão.

AWS CLI

Para criar um modelo de serviço que implanta um serviço sem um pipeline de serviço, adicione o parâmetro e o valor `--pipeline-provisioning "CUSTOMER_MANAGED"` ao comando `create-service-template`. Configure seus pacotes de modelos conforme descrito em criação de Pacotes de modelos e Requisitos de esquema para pacotes de modelos de serviço.

> **Note**
>
> Não é possível modificar o `pipelineProvisioning` após o modelo de serviço ser criado.

1. Use a CLI para registrar e publicar um novo modelo de serviço, com ou sem um pipeline de serviço, conforme mostrado nas etapas a seguir.

   a. Crie um modelo de serviço com um pipeline de serviço usando a CLI.

      Especifique o nome, o nome para exibição (opcional) e a descrição (opcional).

      Execute o seguinte comando:

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service"
```

      Resposta:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service"
    }
}
```

   b. Crie um modelo de serviço sem um pipeline de serviço.

      Adicionar `--pipeline-provisioning`.

      Execute o seguinte comando:

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service" \
    --pipeline-provisioning "CUSTOMER_MANAGED"
```

      Resposta:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Crie uma versão secundária 0 da versão principal 1 do modelo de serviço.

   Inclua o nome do modelo, os modelos de versão compatíveis, a versão principal e o nome e a chave do bucket do S3 para o bucket que contém seu pacote de modelos de serviço.

   Execute o seguinte comando:

```
$ aws proton create-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}" \
    --compatible-environment-templates '[{"templateName":"simple-
env","majorVersion":"1"}]'
```

   Resposta:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
```

```
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

3. Use o comando get para verificar o status dos cadastros.

   Execute o seguinte comando:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   Resposta:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
```

```
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

4. Publique o modelo de serviço usando o comando update para alterar o status para `"PUBLISHED"`.

   Execute o seguinte comando:

```
$ aws proton update-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Resposta:

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
```

```
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
       my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello pipeline
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

5. Verifique se a versão 1.0 AWS Proton foi publicada usando o comando get para recuperar dados detalhados do modelo de serviço.

   Execute o seguinte comando:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   Resposta:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
```

```
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:03:04.767000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```


Visualizar dados do modelo

Você pode visualizar listas de modelos com detalhes e visualizar modelos individuais com dados detalhados usando o console do AWS Proton e o AWS CLI.

Os dados do modelo de ambiente gerenciado pelo cliente incluem o parâmetro `provisioned` com o valor `CUSTOMER_MANAGED`.

Se um modelo de serviço não incluir um pipeline de serviço, os dados do modelo de serviço incluirão o parâmetro `pipelineProvisioning` com o valor `CUSTOMER_MANAGED`.

Para obter mais informações, consulte Registre e publique modelos.

Você pode usar o console ou o AWS CLI para listar e visualizar os dados do modelo.

Console de gerenciamento da AWS

Use o console para listar e visualizar modelos.

1. Para ver uma lista de modelos, escolha modelos (Ambiente ou Serviço).
2. Para visualizar dados detalhados, escolha o nome de um modelo.

   Visualize os dados detalhados do modelo, uma lista das versões principais e secundárias do modelo, uma lista dos AWS Proton recursos que foram implantados usando versões de modelo e tags de modelo.

   A versão principal e a versão secundária recomendadas são rotuladas como Recomendadas.

AWS CLI

Use o AWS CLI para listar e visualizar modelos.

Execute o seguinte comando:

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Resposta:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-10T18:35:08.293000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-10T18:35:11.162000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
```

```
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Execute o seguinte comando:

```
$ aws proton list-environment-templates
```

Resposta:

```
{
    "templates": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-3",
            "createdAt": "2020-11-10T18:35:05.763000+00:00",
            "description": "VPC with Public Access",
            "displayName": "VPC",
            "lastModifiedAt": "2020-11-10T18:35:05.763000+00:00",
            "name": "simple-env-3",
            "recommendedVersion": "1.0"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-1",
            "createdAt": "2020-11-10T00:14:06.881000+00:00",
            "description": "Some SSM Parameters",
            "displayName": "simple-env-1",
            "lastModifiedAt": "2020-11-10T00:14:06.881000+00:00",
            "name": "simple-env-1",
            "recommendedVersion": "1.0"
        }
    ]
}
```

Visualizar uma versão secundária de um modelo de serviço.

Execute o seguinte comando:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Resposta:


```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
  my_sample_pipeline_required_input:\n          type: string\n          description:
 \"Another sample input\"\n\n    MyServiceInstanceInputType:\n      type: object
\n      description: \"Service instance input properties\"\n      required:\n
       - my_sample_service_instance_required_input\n      properties:\n
 my_sample_service_instance_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
     my_sample_service_instance_required_input:\n          type: string\n
 description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
```

```
}
```

Visualize um modelo de serviço sem um pipeline de serviço, conforme mostrado no próximo exemplo de comando e resposta.

Execute o seguinte comando:

```
$ aws proton get-service-template \
    --name "simple-svc-template-cli"
```

Resposta:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/simple-svc-
template-cli",
        "createdAt": "2021-02-18T15:38:57.949000+00:00",
        "displayName": "simple-svc-template-cli",
        "lastModifiedAt": "2021-02-18T15:38:57.949000+00:00",
        "status": "DRAFT",
        "name": "simple-svc-template-cli",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

Atualizar um modelo

Você pode atualizar um modelo conforme descrito na lista a seguir.

- Edite o `description` ou o `display name` de um modelo ao usar o console ou o AWS CLI. Você não pode editar o `name` de um modelo.
- Atualize o status de uma versão secundária do modelo ao usar o console ou o AWS CLI. Você só pode alterar o status de `DRAFT` para `PUBLISHED`.
- Edite o nome de exibição e a descrição de uma versão secundária ou principal de um modelo ao usar o AWS CLI.

Console de gerenciamento da AWS

Edite a descrição e o nome de exibição do modelo usando o console conforme descrito nas etapas a seguir.

Na lista de modelos.

1. No console do AWS Proton, escolha Modelos (Ambiente ou Serviço).
2. Na lista de modelos, escolha o botão de rádio à esquerda do modelo para o qual você deseja atualizar a descrição ou o nome de exibição.
3. Escolha Ações e então Editar.
4. Na página Editar modelo (ambiente ou serviço), na seção Detalhes do modelo, insira suas edições no formulário e escolha Salvar alterações.

Altere o status de uma versão secundária de um modelo usando o console para publicar um modelo conforme descrito a seguir. Você só pode alterar o status de `DRAFT` para `PUBLISHED`.

Na página de detalhes do modelo (ambiente ou serviço).

1. No console do AWS Proton, escolha Modelos (Ambiente ou Serviço).
2. Na lista de modelos, escolha o nome do modelo cujo status de uma versão secundária você deseja atualizar de Rascunho para Publicado.
3. Na página de detalhes do modelo (ambiente ou serviço), na seção Versões do modelo, selecione o botão de rádio à esquerda da versão secundária que você deseja publicar.
4. Escolha Publicar na seção Versões de modelo. O status muda de Rascunho para Publicado.

AWS CLI

O exemplo de comando e resposta a seguir mostra como você pode editar a descrição de um modelo de ambiente.

Execute o comando a seguir.

```
$ aws proton update-environment-template \
    --name "simple-env" \
    --description "A single VPC with public access"
```

Resposta:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-28T22:02:10.651000+00:00",
        "description": "A single VPC with public access",
        "displayName": "simple-env",
        "lastModifiedAt": "2020-11-29T16:11:18.956000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```


Você também pode usar o AWS CLI para atualizar modelos de serviço. Consulte o Registrar e publicar modelos de serviço, etapa 5 para obter um exemplo de atualização do status de uma versão secundária de um modelo de serviço.

Excluir modelos

Os modelos podem ser excluídos usando o console e o AWS CLI.

Você pode excluir uma versão secundária de um modelo de ambiente se não houver ambientes implantados nessa versão.

Você pode excluir uma versão secundária de um modelo de serviço se não houver instâncias de serviço ou pipelines implantados nessa versão. Seu pipeline pode ser implantado em uma versão de modelo diferente da sua instância de serviço. Por exemplo, se sua instância de serviço for atualizada da versão 1.0 para a 1.1 e seu pipeline ainda estiver implantado para a versão 1.0, você não poderá excluir o modelo de serviço 1.0.

Console de gerenciamento da AWS

Você pode usar o console para excluir o modelo inteiro ou as versões secundárias e principais individuais de um modelo.

Use o console para excluir modelos da seguinte maneira.

> **Note**
>
> Ao usar o console para excluir modelos.
>
> - Ao excluir o modelo inteiro, você também exclui as versões principais e secundárias do modelo.

Na lista de modelos (ambiente ou serviço).

1. No console do AWS Proton, escolha Modelos (Ambiente ou Serviço).
2. Na lista de modelos, marque a caixa de opção à esquerda do modelo a ser excluído.

   Você só pode excluir um modelo inteiro se não houver AWS Proton recursos implantados em suas versões.
3. Escolha Ações e, em seguida, Excluir para excluir o modelo inteiro.
4. Um modal pede que você confirme a ação de exclusão.
5. Siga as instruções e escolha Sim, excluir.

Na página de detalhes do modelo (ambiente ou serviço).

1. No console do AWS Proton, escolha Modelos (Ambiente ou Serviço).
2. Na lista de modelos, escolha o nome do modelo que deseja excluir por completo ou exclua versões principais ou secundárias individuais.
3. Para excluir o modelo inteiro.

   Você só pode excluir um modelo inteiro se não houver AWS Proton recursos implantados em suas versões.

   a. Escolha Excluir, canto superior direito da página.

b. Um modal pede que você confirme a ação de exclusão.

c. Siga as instruções e escolha Sim, excluir.

4. Para excluir versões principais ou secundárias de um modelo.

   Você só pode excluir uma versão secundária de um modelo se não houver AWS Proton recursos implantados nessa versão.

   a. Na seção Versões do modelo, marque a caixa de opção à esquerda da versão que você deseja excluir.

   b. Escolha Excluir na seção Versões de modelo.

   c. Um modal pede que você confirme a ação de exclusão.

   d. Siga as instruções e escolha Sim, excluir.

AWS CLI

AWS CLI as operações de exclusão de modelos não incluem a exclusão de outras versões de um modelo. Ao usar o AWS CLI, exclua modelos com as seguintes condições.

- Exclua um modelo inteiro se não existirem versões secundárias ou principais do modelo.
- Exclua uma versão principal ao excluir a última versão secundária restante.
- Exclua uma versão secundária de um modelo se não houver AWS Proton recursos implantados nessa versão.
- Exclua a versão secundária recomendada de um modelo se nenhuma outra versão secundária do modelo existir e se não houver AWS Proton recursos implantados nessa versão.

Os exemplos de comandos e respostas a seguir mostram como usar o AWS CLI para excluir modelos.

Execute o seguinte comando:

```
$ aws proton delete-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Resposta:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Execute o seguinte comando:

```
$ aws proton delete-environment-template \
    --name "simple-env"
```

Resposta:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with Public Access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-12T00:23:22.339000+00:00",
        "name": "simple-env",
        "recommendedVersion": "1.0"
    }
}
```

Execute o seguinte comando:

```
$ aws proton delete-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Resposta:

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [{"majorVersion": "1", "templateName":
 "simple-env"}],
        "createdAt": "2020-11-28T22:07:05.798000+00:00",
        "lastModifiedAt": "2020-11-28T22:19:05.368000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

Configurações de sincronização de modelo

Saiba como configurar um modelo para permitir a AWS Proton sincronização a partir de pacotes de modelos localizados nos repositórios git registrados que você define. Quando um commit é enviado para o seu repositório, o AWS Proton verifica se há alterações nos pacotes de modelos do repositório. Se ele detectar uma alteração no pacote de modelos, uma nova versão secundária ou principal do modelo será criada, caso a versão ainda não exista. AWS Proton atualmente suporta GitHub, GitHub Enterprise BitBucket e.

Enviando um commit para um pacote de modelos sincronizados

Quando você envia um commit para uma ramificação que está sendo monitorada por um de seus modelos, o AWS Proton clona seu repositório e determina quais modelos ele precisa sincronizar. Ele verifica os arquivos em seu diretório para encontrar diretórios que correspondam à convenção de `{template-name}/{major-version}/`.

Depois de AWS Proton determinar quais modelos e versões principais estão associados ao seu repositório e ramificação, ele começa a tentar sincronizar todos esses modelos em paralelo.

Durante cada sincronização com um modelo específico, AWS Proton primeiro verifica se o conteúdo do diretório do modelo mudou desde a última sincronização bem-sucedida. Se o conteúdo não mudar, AWS Proton ignora o registro de um pacote duplicado. Isso garante que uma nova versão

secundária do modelo seja criada se o conteúdo do pacote de modelos mudar. Se o conteúdo do pacote de modelos for alterado, o pacote será registrado com. AWS Proton

Depois que o pacote de modelos for registrado, AWS Proton monitora o status do registro até que o registro seja concluído.

Somente uma sincronização pode ocorrer em uma determinada versão secundária e principal do modelo em um determinado momento. Todos os commits que possam ter sido enviados enquanto uma sincronização estava em andamento são agrupados em lotes. Os commits em lote são sincronizados após a conclusão da tentativa de sincronização anterior.

Como sincronizar modelos de serviço

AWS Proton pode sincronizar modelos de ambiente e serviço do seu repositório git. Para sincronizar seus modelos de serviço, você adiciona um arquivo adicional chamado `.template-registration.yaml` a cada diretório de versão principal em seu pacote de modelos. Esse arquivo contém detalhes adicionais AWS Proton necessários ao criar uma versão do modelo de serviço para você após uma confirmação: ambientes compatíveis e fontes de componentes compatíveis.

O caminho completo do arquivo é *`service-template-name`*`/`*`major-version`*`/.template-registration.yaml`. Para obter mais informações, consulte the section called “Como sincronizar modelos de serviço”.

Considerações sobre a configuração de sincronização de modelos

Analise as seguintes considerações sobre o uso de configurações de sincronização de modelos.

- Os repositórios não devem ter mais de 250 MB.
- Para configurar a sincronização de modelos, primeiro vincule o repositório ao AWS Proton. Para obter mais informações, consulte the section called “Criar um link de repositório”.
- Quando uma nova versão do modelo é criada a partir de um modelo sincronizado, ela está no estado de `DRAFT`.
- Uma versão secundária de um modelo será criada se uma das seguintes opções for verdadeira:
  - O conteúdo do pacote de modelos é diferente do conteúdo da última versão secundária do modelo sincronizado.
  - A última versão secundária do modelo sincronizado anteriormente foi excluída.
- A sincronização não pode ser pausada.
- Tanto as novas versões secundárias quanto as principais são sincronizadas automaticamente.

- Novos modelos de nível superior não podem ser criados pelas configurações de sincronização de modelos.
- Você não pode sincronizar com um modelo de vários repositórios usando uma configuração de sincronização de modelos.
- Não é possível usar tags em vez de ramificações.
- Ao criar um modelo de serviço, você especifica modelos de ambiente compatíveis.
- Você pode criar um modelo de ambiente e adicioná-lo como um ambiente compatível para seu modelo de serviço no mesmo commit.
- As sincronizações com uma única versão principal do modelo são executadas uma por vez. Durante uma sincronização, se novos commits forem detectados, eles serão agrupados em lotes e aplicados no final da sincronização ativa. As sincronizações com diferentes versões principais do modelo acontecem paralelamente.
- Se você alterar a ramificação a partir da qual seus modelos estão sendo sincronizados, todas as sincronizações contínuas da ramificação antiga serão concluídas primeiro. Em seguida, a sincronização começa a partir da nova ramificação.
- Se você alterar o repositório a partir do qual seus modelos são sincronizados, qualquer sincronização contínua do repositório antigo poderá falhar ou ser concluída. Depende do estágio da sincronização em que estão.

Para obter mais informações, consulte The AWS Proton Service API Reference.

Tópicos

Criar uma configuração de sincronização de modelos

Saiba como criar uma configuração de sincronização de modelos com AWS Proton o.

Criar os pré-requisitos de uma configuração de sincronização de modelos:

- Você vinculou um repositório com o AWS Proton.
- Um pacote de modelos está localizado no seu repositório.

O link do repositório consiste no seguinte:

- Uma CodeConnections conexão que dá AWS Proton permissão para acessar seu repositório e assinar suas notificações.
- Um perfil vinculado a serviço. Ao vincular o repositório, o perfil vinculado ao serviço será criado para você.

Antes de criar sua primeira configuração de sincronização de modelos, envie um pacote de modelos para seu repositório, conforme mostrado no layout de diretório a seguir.

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
/templates/my-env-template/v1/schema/
```

Depois de criar sua primeira configuração de sincronização de modelos, novas versões de modelo são criadas automaticamente quando você envia um commit que adiciona um pacote de modelos atualizado em uma nova versão (por exemplo, em `/my-env-template/v2/`).

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

Você pode incluir novas versões do pacote de modelos para um ou mais modelos configurados para sincronização em um único commit. AWS Proton cria uma nova versão de modelo para cada nova versão do pacote de modelos que foi incluída no commit.

Depois de criar a configuração de sincronização do modelo, você ainda pode criar manualmente novas versões do modelo no console ou com o AWS CLI fazendo o upload de pacotes de modelos de um bucket do S3. A sincronização de modelos só funciona em uma direção: do seu repositório para o. AWS Proton As versões do modelo criadas manualmente não são sincronizadas.

Depois de definir uma configuração de sincronização de modelos, AWS Proton escuta as alterações no seu repositório. Sempre que uma alteração é enviada, ela procura por qualquer diretório que

tenha o mesmo nome do seu modelo. Em seguida, ele procura dentro desse diretório por quaisquer diretórios que se pareçam com as versões principais. AWS Proton registra o pacote de modelos na versão principal do modelo correspondente. As novas versões estão sempre no estado de DRAFT. Você pode publicar as novas versões com o console ou AWS CLI.

Por exemplo, suponha que você tenha um modelo chamado `my-env-template` configurado para sincronizar a partir do `my-repo/templates` em uma ramificação `main` com o layout a seguir.

```
/code
/code/service.go
README.md
/templates/
/templates/my-env-template/
/templates/my-env-template/v1/
/templates/my-env-template/v1/infrastructure/
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

AWS Proton sincroniza o conteúdo de `/templates/my-env-template/v1/` para `my-env-template:1` e o conteúdo de `/templates/my-env-template/v2/` para`my-env-template:2`. Se eles ainda não existirem, ele cria essas versões principais.

AWS Proton encontrou o primeiro diretório que correspondia ao nome do modelo. Você pode limitar as AWS Proton pesquisas de diretórios especificando um `subdirectoryPath` ao criar ou editar uma configuração de sincronização de modelo. Por exemplo, é possível especificar `/production-templates/` para o `subdirectoryPath`.

Você pode criar um modelo de configuração de sincronização utilizando o console ou o CLI.

Console de gerenciamento da AWS

Criar um modelo e um um modelo de configuração de sincronização utilizando o console.

1. No console do AWS Proton, escolha Modelos de ambiente.
2. Selecione Criar modelo de ambiente.
3. Na página Criar modelo de ambiente, na seção Opções de modelo, escolha Criar um modelo para provisionar novos ambientes.
4. Na seção Fonte do pacote de modelos, escolha Sincronizar modelos do Git.

5. Na seção Repositório do código-fonte:

   a. Para Repositório, selecione o repositório vinculado que contém seu pacote de modelos.

   b. Para Ramificação, selecione a ramificação do repositório de onde sincronizar.

   c. (Opcional) Para Diretório pacotes de modelos, insira o nome de um diretório para analisar a pesquisa do pacote de modelo.

6. Na seção Detalhes do modelo.

   a. Insira um nome de modelo.

   b. (Opcional) Insira um Nome para exibição do modelo.

   c. (Opcional) Insira uma Descrição de modelo para o modelo de ambiente.

7. (Opcional) Marque a caixa de seleção Personalizar configurações de criptografia (avançado) na seção Configurações de criptografia para fornecer sua própria chave de criptografia.

8. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.

9. Escolha Criar modelo de ambiente.

   Agora você está em uma nova página que exibe o status e os detalhes do seu novo modelo de ambiente. Esses detalhes incluem uma lista de tags AWS gerenciadas e gerenciadas pelo cliente. AWS Proton gera automaticamente tags AWS gerenciadas para você quando você cria AWS Proton recursos. Para obter mais informações, consulte AWS Proton recursos e marcação.

10. Na página de detalhes do modelo, escolha a guia Sincronizar para ver os dados detalhados da configuração da sincronização do modelo.

11. Escolha a guia Versões do modelo para visualizar as versões do modelo com detalhes do status.

12. O status de um novo status de modelo de ambiente começa no estado Rascunho. Você e outras pessoas com permissões de `proton:CreateEnvironment` podem visualizá-lo e acessá-lo. Siga a próxima etapa para disponibilizar o modelo para outras pessoas.

13. Na seção Versões do modelo, escolha o botão de rádio à esquerda da versão secundária do modelo que você acabou de criar (1.0). Como alternativa, você pode escolher Publicar no alerta de informações e pular a próxima etapa.

14. Na seção Versões do modelo, escolha Publicar.

15. O status do modelo muda para Publicado. É a versão mais recente e recomendada do modelo.

16. No painel de navegação, selecione Modelos de ambiente para ver uma lista dos modelos e detalhes do seu ambiente.

O procedimento para criar um modelo de serviço e a configuração de sincronização de modelos é semelhante.

AWS CLI

Criar um modelo e um um modelo de configuração de sincronização utilizando o AWS CLI.

1. Criar um modelo. Neste exemplo, um modelo de ambiente é criado.

   Execute o comando a seguir.

   ```
   $ aws proton create-environment-template \
       --name "env-template"
   ```

   A resposta é a que segue.

   ```
   {
       "environmentTemplate": {
           "arn": "arn:aws:proton:us-east-1:123456789012:environment-template/env-template",
           "createdAt": "2021-11-07T23:32:43.045000+00:00",
           "displayName": "env-template",
           "lastModifiedAt": "2021-11-07T23:32:43.045000+00:00",
           "name": "env-template",
           "status": "DRAFT",
           "templateName": "env-template"
       }
   }
   ```

2. Crie sua configuração de sincronização de modelos AWS CLI fornecendo o seguinte:

   - O modelo com o qual deseja sincronizar. Depois de criar a configuração de sincronização do modelo, você ainda pode criar novas versões dele manualmente no console ou com o AWS CLI.
   - O nome do modelo.
   - O tipo de modelo.
   - O repositório vinculado do qual deseja sincronizar.

- O provedor do repositório vinculado.
- A ramificação em que o pacote de modelos está localizado.
- (Opcional) O caminho para o diretório que contém o pacote de modelo. Por padrão, AWS Proton procura o primeiro diretório que corresponda ao nome do seu modelo.

Execute o comando a seguir.

```
$ aws proton create-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-name "myrepos/templates" \
    --repository-provider "GITHUB" \
    --branch "main" \
    --subdirectory "env-template/"
```

A resposta é a que segue.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

3. Para publicar sua versão de modelo, consulte Registre e publique modelos.

Como sincronizar modelos de serviço

Os exemplos anteriores mostram como sincronizar modelos de ambiente. Os modelos de serviço são semelhantes. Para sincronizar modelos de serviço, você adiciona um arquivo adicional chamado `.template-registration.yaml` a cada diretório de versão principal em seu pacote de modelos. Esse arquivo contém detalhes adicionais AWS Proton necessários ao criar uma versão de modelo de serviço para você após uma confirmação. Ao criar explicitamente uma versão de modelo de serviço usando o AWS Proton console ou a API, você fornece esses detalhes como entradas, e esse arquivo substitui essas entradas para sincronização de modelos.

```
./templates/                                                    # subdirectory (optional)
 /templates/my-svc-template/                                    # service template name
 /templates/my-svc-template/v1/                                 # service template version
 /templates/my-svc-template/v1/.template-registration.yaml      # service template version
 properties
 /templates/my-svc-template/v1/instance_infrastructure/         # template bundle
 /templates/my-svc-template/v1/schema/
```

O arquivo `.template-registration.yaml` contém os detalhes a seguir.

- Ambientes compatíveis [obrigatório]: Ambientes baseados nesses modelos de ambiente e versões principais são compatíveis com serviços baseados nessa versão de modelo de serviço.
- Fontes de componentes compatíveis [opcional]: Componentes que usam essas fontes são compatíveis com os serviços baseados nessa versão de modelo de serviço. Se não for especificado, os componentes não poderão ser anexados a esses serviços. Para obter mais informações sobre componentes, consulte Componentes.

A sintaxe YAML do arquivo é a seguinte:

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
  - DIRECTLY_DEFINED
```

Especifique uma ou mais combinações de modelo de ambiente/versão principal. A especificação `supported_component_sources` é opcional e o único valor suportado é `DIRECTLY_DEFINED`.

Example.template-registration.yaml

Neste exemplo, a versão do modelo de serviço é compatível com as versões principais 1 e 2 do modelo de ambiente `my-env-template`. Também é compatível com as versões principais 1 e 3 do modelo de ambiente `another-env-template`. O arquivo não especifica o `supported_component_sources`; portanto, os componentes não podem ser anexados aos serviços com base nessa versão do modelo de serviço.

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
```

```
  - another-env-template:3
```

> **Note**
>
> Anteriormente, AWS Proton definiu um arquivo diferente `.compatible-envs`,, para especificar ambientes compatíveis. AWS Proton ainda suporta esse arquivo e seu formato para compatibilidade com versões anteriores. Não recomendamos mais usá-lo, porque ele não é extensível e não oferece suporte a recursos mais novos, como componentes.

Exibir detalhes da configuração de sincronização do modelo

Visualizar dados detalhados da configuração de sincronização do modelo usando o console ou a CLI.

Console de gerenciamento da AWS

Usar o console para visualizar detalhes da configuração de sincronização de modelo.

1. No painel de navegação, escolha modelos (Ambiente ou Serviço).
2. Para visualizar dados detalhados, escolha o nome de um modelo para o qual você criou uma configuração de sincronização de modelos.
3. Na página de detalhes para o modelo, escolha a guia Sincronizar para ver os dados detalhados da configuração da sincronização do modelo.

AWS CLI

Use o AWS CLI para ver um modelo sincronizado.

Execute o comando a seguir.

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

A resposta é a que segue.

```
{
    "templateSyncConfigDetails": {
```

```
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
        "templateType": "SERVICE"
    }
}
```

Use o AWS CLI para obter o status de sincronização do modelo.

Para `template-version`, insira a versão principal do modelo.

Execute o comando a seguir.

```
$ aws proton get-template-sync-status \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

Editar uma configuração de sincronização de modelos

Você pode editar qualquer um dos parâmetros de configuração de sincronização do modelo, exceto `template-name` e `template-type`.

Saiba como editar um modelo de configuração de sincronização utilizando o console ou o CLI.

Console de gerenciamento da AWS

Editar uma ramificação de modelo de configuração de sincronização utilizando o console.

Na lista de modelos.

1. No console do AWS Proton, escolha Modelos (Ambiente ou Serviço).
2. Na lista de modelos, escolha o nome do modelo com a configuração de sincronização de modelo que deseja editar.
3. Na página de detalhes do modelo, escolha a guia Sincronização de modelos.
4. Na seção Detalhes de sincronização de modelo, escolha Editar.
5. Na página Editar, na seção Repositório de código-fonte, para Ramificação, selecione uma ramificação e escolha Salvar configuração.

AWS CLI

O exemplo de comando e resposta a seguir mostra como você pode editar uma configuração de sincronização de modelo **branch** usando a CLI.

Execute o comando a seguir.

```
$ aws proton update-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-provider "GITHUB" \
    --repository-name "myrepos/templates" \
    --branch "fargate" \
    --subdirectory "env-template"
```

A resposta é a que segue.

```
{
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Você também pode usar o AWS CLI para atualizar modelos de serviço sincronizados.

Excluir uma configuração de sincronização de modelos

Excluir um modelo de configuração de sincronização utilizando o console ou o CLI.

Console de gerenciamento da AWS

Excluir um modelo de configuração de sincronização utilizando o console.

1. Na página de detalhes do modelo, escolha a guia Sincronização.
2. Na seção Detalhes da sincronização, escolha Desconectar.

AWS CLI

Os exemplos de comandos e respostas a seguir mostram como usar o AWS CLI para excluir configurações de modelos sincronizados.

Execute o comando a seguir.

```
$ aws proton delete-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT"
```

A resposta é a que segue.

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Configurações de sincronização de serviços

Com a sincronização de serviços, você pode configurar e implantar seus AWS Proton serviços usando o Git. Você pode usar o service sync para gerenciar implantações e atualizações iniciais do seu AWS Proton serviço com uma configuração definida em um repositório Git. Por meio do Git, você pode usar atributos como controle de versão e pull requests para configurar, gerenciar e implantar seus serviços. O Service Sync combina AWS Proton com o Git para ajudar você a provisionar uma infraestrutura padronizada que é definida e gerenciada por meio de modelos. AWS Proton Ele gerencia as definições de serviço em seu repositório Git e reduz a troca de ferramentas. Em comparação com o uso exclusivo do Git, a padronização de modelos e a implantação AWS Proton ajudam você a gastar menos tempo gerenciando sua infraestrutura. AWS Proton também oferece maior transparência e auditabilidade para desenvolvedores e equipes de plataforma.

AWS Proton Arquivo OPS

O `proton-ops` arquivo define onde AWS Proton encontra o arquivo de especificação usado para atualizar sua instância de serviço. Ela também define em qual ordem atualizar as instâncias de serviço e quando promover mudanças de uma instância para outra.

O arquivo do `proton-ops` suporta a sincronização de uma instância de serviço usando o arquivo de especificação, ou vários arquivos de especificação, encontrados no seu repositório vinculado. Você pode fazer isso definindo um bloco de sincronização no arquivo `proton-ops`, como no exemplo a seguir.

Exemplo ./configuration/proton-ops.yaml:

```
sync:
  services:
      frontend-svc:
          alpha:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          beta:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          gamma:
              branch: pre-prod
              spec: ./frontend-svc/pre-prod/frontend-spec.yaml
          prod-one:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-two:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-three:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

No exemplo anterior, `frontend-svc` é o nome do serviço, e `alpha`, `beta`, `gamma`, `prod-one`, `prod-two`, e `prod-three` são as instâncias.

O arquivo de `spec` pode ser todas as instâncias ou um subconjunto das instâncias definidas no arquivo de `proton-ops`. No entanto, no mínimo, ele deve ter a instância definida na ramificação e a especificação a partir da qual está sincronizando. Se as instâncias não estiverem definidas no arquivo de `proton-ops`, com a ramificação e o local do arquivo de `spec` específicos, a sincronização de serviços não criará nem atualizará essas instâncias.

Os exemplos a seguir mostram a aparência dos arquivos de `spec`. Lembre-se de que o arquivo de `proton-ops` é sincronizado a partir desses arquivos de `spec`.

Exemplo **`./frontend-svc/test/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "alpha"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "beta"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Exemplo **`./frontend-svc/pre-prod/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "gamma"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Exemplo **`./frontend-svc/prod/frontend-spec-second.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "prod-one"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-two"
  environment: "frontend-env"
```

```
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-three"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Se uma instância não sincronizar e houver um problema contínuo ao tentar sincronizá-la, chamar a API de `GetServiceInstanceSyncStatus` pode ajudar a resolver o problema.

> **Note**
>
> Os clientes que usam o service sync ainda estão restritos por AWS Proton limites.

Blockers

Ao sincronizar seu serviço usando o AWS Proton service sync, você pode atualizar sua especificação de serviço e criar e atualizar instâncias de serviço a partir do seu repositório Git. No entanto, pode haver momentos em que você precise atualizar um serviço ou instância manualmente por meio do Console de gerenciamento da AWS ou AWS CLI.

AWS Proton ajuda a evitar a substituição de quaisquer alterações manuais feitas por meio do Console de gerenciamento da AWS ou AWS CLI, como atualizar uma instância de serviço ou excluir uma instância de serviço. Para isso, o AWS Proton cria automaticamente um bloqueador de sincronização de serviços desativando a sincronização de serviços ao detectar uma alteração manual.

Para obter todos os bloqueadores associados a um serviço, você deve fazer o seguinte na ordem de cada `serviceInstance` associada ao serviço:

- Você só deve chamar a `getServiceSyncBlockerSummary` API com o `serviceName`.
- Chamar a API `getServiceSyncBlockerSummary` com o `serviceName` e o `serviceInstanceName`.

Ela retorna uma lista dos bloqueadores mais recentes e o status associado a eles. Se algum bloqueador estiver marcado como ATIVO, você deverá resolvê-lo chamando a API `UpdateServiceSyncBlocker` com `blockerId` e `resolvedReason` para cada um.

Se você atualizar ou criar manualmente uma instância de serviço, AWS Proton cria um bloqueador de sincronização de serviço na instância de serviço. AWS Proton continua sincronizando todas as outras instâncias de serviço, mas desativa a sincronização dessa instância de serviço até que o bloqueador seja resolvido. Se você excluir uma instância de serviço de um serviço, AWS Proton cria um bloqueador de sincronização de serviço no serviço. Isso AWS Proton impede a sincronização de qualquer uma das instâncias do serviço até que o bloqueador seja resolvido.

Depois de ter todos os bloqueadores ativos, você deve resolvê-los chamando a API `UpdateServiceSyncBlocker` com `blockerId` e `resolvedReason` para cada um dos bloqueadores ativos.

Usando o Console de gerenciamento da AWS, você pode determinar se uma sincronização de serviço está desativada navegando até a guia Sincronização de serviços AWS Proton e escolhendo-a. Se o serviço ou as instâncias do serviço estiverem bloqueados, um botão Habilitar será exibido. Para habilitar a sincronização de serviços, escolha Habilitar.

Tópicos

Criar uma configuração de sincronização de serviço

Você pode criar uma configuração de sincronização de serviço usando o console ou AWS CLI.

Console de gerenciamento da AWS

1. Na página Escolher um modelo de serviço, selecione um modelo e escolha Configurar.
2. Na página Configurar serviço, na seção Detalhes do serviço, insira um novo Nome de serviço.
3. (Opcional) Insira uma descrição para o serviço.

4. Na seção Repositório de código-fonte do aplicativo, escolha Escolha um repositório Git vinculado para selecionar um repositório ao qual você já tenha vinculado. AWS Proton Se você ainda não tiver um repositório vinculado, escolha Vincular outro repositório Git e siga as instruções em Criar um link para seu repositório.
5. Para Repositório, escolha o nome do seu repositório de códigos-fonte da lista.
6. Em Ramificação, escolha o nome da ramificação do repositório de código-fonte na lista.
7. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
8. Escolha Próximo.
9. Na página Configurar instâncias de serviço, na seção Fonte de definição de serviço, selecione Sincronizar seu serviço a partir do Git.
10. Na seção Arquivos de definição de serviço, se você quiser que o AWS Proton crie seu arquivo do `proton-ops`, selecione Quero que o AWS Proton crie os arquivos. Com essa opção, AWS Proton cria o `proton-ops` arquivo `spec` and nos locais que você especificar. Selecione Estou fornecendo meus próprios arquivos para criar seu próprio arquivo OPS.
11. Na seção Repositório de definição de serviço, escolha Escolher um repositório Git vinculado para selecionar um repositório ao qual você já tenha vinculado. AWS Proton
12. Para Nome do repositório, escolha o nome do seu repositório de códigos-fonte da lista.
13. Para ramificação de **`proton-ops`** arquivo, escolha o nome de sua ramificação na lista onde AWS Proton colocará seu OPS e arquivo de especificação.
14. Na seção Instâncias do serviço, cada campo é preenchido automaticamente com base nos valores do arquivo de `proton-ops`.
15. Escolha Avançar e revise suas entradas.
16. Escolha Criar.

AWS CLI

Crie uma configuração de sincronização de serviço usando o AWS CLI

- Execute o comando a seguir.

```
$ aws proton create-service-sync-config \
    --resource "service-arn" \
    --repository-provider "GITHUB" \
```

```
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --proton-ops-file "./configuration/custom-proton-ops.yaml" (optional)
```

A resposta é a que segue.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Visualizar detalhes da configuração de uma sincronização de serviço

É possível exibir os dados de detalhes de configuração para uma sincronização do serviço usando o console do ou o AWS CLI.

Console de gerenciamento da AWS

Use o console do para visualizar detalhes da configuração de uma sincronização de serviço

1. No painel de navegação, escolha Serviços.
2. Para visualizar dados detalhados, escolha o nome de um serviço para o qual você criou uma configuração de sincronização de serviços.
3. Na página de detalhes do serviço, selecione a guia Sincronização do serviço para ver os dados detalhados da configuração da sincronização do serviço.

AWS CLI

Use o AWS CLI para obter um serviço sincronizado.

Execute o comando a seguir.

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

A resposta é a que segue.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Use o AWS CLI para obter o status de sincronização do serviço.

Execute o comando a seguir.

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

Editar uma configuração de sincronização de serviço

Você pode editar uma configuração de sincronização de serviço usando o console ou AWS CLI.

Console de gerenciamento da AWS

Edite uma configuração de sincronização do serviço usando o console.

1. No painel de navegação, escolha Serviços.
2. Para visualizar dados detalhados, escolha o nome de um serviço para o qual você criou uma configuração de sincronização de serviços.
3. Na página de detalhes de serviço, escolha a guia Sincronização de serviço.
4. Na seção Sincronização de serviços, escolha Editar.
5. Na página Editar, atualize as informações que deseja editar e escolha Salvar.

AWS CLI

O exemplo de comando e resposta a seguir mostra como você pode editar uma configuração de sincronização de serviço usando a AWS CLI.

Execute o comando a seguir.

```
$ aws proton update-service-sync-config \
    --service-name "service name" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --ops-file "./configuration/custom-proton-ops.yaml"
```

A resposta é a que segue.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Excluir uma configuração de sincronização de serviço

Você pode excluir uma configuração de sincronização de serviço usando o console ou AWS CLI.

Console de gerenciamento da AWS

Exclua uma configuração de sincronização do serviço usando o console

1. Na página de detalhes de serviço, escolha a guia Sincronização de serviço.
2. Na seção Detalhes da sincronização do serviço, escolha Desconectar para desconectar seu repositório. Depois que seu repositório for desconectado, não sincronizaremos mais o serviço desse repositório.

AWS CLI

Os exemplos de comandos e respostas a seguir mostram como usar o AWS CLI para excluir configurações sincronizadas do serviço.

Execute o comando a seguir.

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

A resposta é a que segue.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> A sincronização de serviços não exclui instâncias de serviço. Ele só exclui a configuração.

AWS Proton ambientes

Pois AWS Proton, um ambiente representa o conjunto de recursos e políticas compartilhados nos quais os AWS Proton serviços são implantados. Eles podem conter quaisquer recursos que se espera que sejam compartilhados entre instâncias AWS Proton de serviço. Esses recursos podem incluir VPCs clusters e balanceadores de carga compartilhados ou gateways de API. Um AWS Proton ambiente deve ser criado antes que um serviço possa ser implantado nele.

Esta seção descreve como gerenciar ambientes utilizando as operações de criar, visualizar, atualizar e excluir. Para obter mais informações, consulte The AWS Proton Service API Reference.

Tópicos

Perfis do IAM

Com AWS Proton, você fornece as funções e AWS KMS chaves do IAM para os AWS recursos que você possui e gerencia. Posteriormente, eles são aplicados e usados por recursos pertencentes e gerenciados por desenvolvedores. Você cria uma função do IAM para controlar o acesso da sua equipe de desenvolvedores à AWS Proton API.

AWS Proton função de serviço

Ao criar um novo ambiente, você fornece um perfil de serviço do IAM relacionado. O perfil contém todas as permissões necessárias para atualizar toda a infraestrutura provisionada definida nos modelos de ambiente e nos modelos de serviço. Para exemplos de perfis, consulteAWS Proton função de serviço para provisionamento usando CloudFormation. Se você usar conexões de conta

de ambiente e contas de ambiente, você cria o perfil em uma conta de ambiente selecionada. Para obter mais informações, consulte Criar um ambiente em uma conta e provisionar em outra conta e Conexões de conta de ambiente.

A forma como você fornece esse perfil de serviço e quem assume o perfil depende do método de provisionamento do seu ambiente.

- AWS-provisionamento gerenciado — você fornece a função diretamente ao AWS Proton criar um ambiente ou indiretamente por meio de conexões de conta. AWS Proton assume a função na conta relevante para provisionar o ambiente e a infraestrutura de serviços.
- Provisionamento autogerenciado: É sua responsabilidade configurar sua automação de provisionamento para assumir um perfil apropriado usando as credenciais apropriadas quando um pull request (PR) aciona uma ação de provisionamento. Para ver um exemplo de GitHub ação que assume uma função, consulte Assumindo uma função na documentação “Configurar AWS credenciais” da ação para GitHub ações.

Para obter mais informações sobre métodos de provisionamento, consulte the section called “Métodos de provisionamento”.

Criar um ambiente

Aprenda a criar AWS Proton ambientes.

Você pode criar um AWS Proton ambiente de duas maneiras:

- Crie, gerencie e provisione um ambiente padrão usando um modelo de ambiente padrão. AWS Proton provisiona a infraestrutura para seu ambiente.
- Conecte-se AWS Proton à infraestrutura gerenciada pelo cliente usando um modelo de ambiente gerenciado pelo cliente. Você provisiona seus próprios recursos compartilhados fora do e AWS Proton, em seguida, fornece saídas de provisionamento que AWS Proton podem ser usadas.

Você pode escolher uma das várias abordagens de provisionamento ao criar um ambiente.

- AWS provisionamento gerenciado — crie, gerencie e provisione um ambiente em uma única conta. AWS Proton provisiona seu ambiente.

  Esse método só oferece suporte a modelos de código de CloudFormation infraestrutura (IaC).

- AWS provisionamento gerenciado para outra conta — Em uma única conta de gerenciamento, crie e gerencie um ambiente provisionado em outra conta com conexões de conta de ambiente. AWS Proton provisiona seu ambiente na outra conta. Para obter mais informações, consulte Criar um ambiente em uma conta e provisionar em outra conta e Conexões de conta de ambiente.

  Esse método suporta apenas modelos de CloudFormation IaC.
- Provisionamento autogerenciado — AWS Proton envia pull requests de provisionamento para um repositório vinculado com sua própria infraestrutura de provisionamento.

  Esse método oferece suporte somente a modelos do Terraform IaC.
- CodeBuild provisionamento — AWS Proton usa AWS CodeBuild para executar comandos de shell fornecidos por você. Seus comandos podem ler as entradas que AWS Proton fornecem e são responsáveis por provisionar ou desprovisionar a infraestrutura e gerar valores de saída. Um pacote de modelos para esse método inclui seus comandos em um arquivo de manifesto e quaisquer programas, scripts ou outros arquivos que esses comandos possam precisar.

  Como exemplo do uso do CodeBuild provisionamento, você pode incluir um código que usa o AWS Cloud Development Kit (AWS CDK) para provisionar AWS recursos e um manifesto que instala o CDK e executa seu código CDK.

  Para obter mais informações, consulte the section called “CodeBuild pacote”.

  > **Note**
  >
  > Você pode usar o CodeBuild provisionamento com ambientes e serviços. No momento, você não pode provisionar componentes dessa forma.

Com o provisionamento AWS gerenciado (na mesma conta e em outra conta), AWS Proton faz chamadas diretas para provisionar seus recursos.

Com o provisionamento autogerenciado, AWS Proton faz pull requests para fornecer arquivos IaC compilados que seu mecanismo de IaC usa para provisionar recursos.

Para obter mais informações, consulte the section called “Métodos de provisionamento”, the section called “Pacotes de modelos” e the section called “Requisitos do esquema do ambiente”.

Tópicos

- Crie e provisione um ambiente padrão na mesma conta

- Criar um ambiente em uma conta e provisionar em outra conta
- Crie e provisione um ambiente usando o provisionamento autogerenciado

Crie e provisione um ambiente padrão na mesma conta

Use o console ou AWS CLI para criar e provisionar um ambiente em uma única conta. O provisionamento é gerenciado por. AWS

Console de gerenciamento da AWS

Use o console para criar e provisionar um ambiente em uma única conta.

1. No console do AWS Proton, escolha Ambientes.
2. Selecione Criar ambiente.
3. Na página Escolher um modelo de ambiente, selecione um modelo e escolha Configurar.
4. Na página Configurar ambiente, na seção Provisionamento, escolha Provisionamento gerenciado pela AWS .
5. Na seção Conta de implantação, escolha Esta Conta da AWS.
6. Na página Configurar ambiente, na seção Configurações do ambiente, insira um Nome de ambiente.
7. (Opcional) Insira uma descrição do ambiente.
8. Na seção Perfis de ambiente, selecione o perfil de serviço do AWS Proton que você criou como parte de Configurando funções AWS Proton de serviço.
9. (Opcional) Na seção Perfil do componente, selecione um perfil de serviço que permita que componentes diretamente definidos sejam executados no ambiente e defina o escopo dos recursos que eles podem provisionar. Para obter mais informações, consulte Componentes.
10. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
11. Escolha Próximo.
12. Na página Definir configurações personalizadas do ambiente, você deve inserir valores para os parâmetros `required`. Você pode inserir valores para os parâmetros `optional` ou usar os padrões quando fornecidos.
13. Escolha Avançar e revise suas entradas.
14. Escolha Criar.

Visualize os detalhes e o status do ambiente, bem como as tags gerenciadas pela AWS e as tags gerenciadas pelo cliente para seu ambiente.

15. No painel de navegação, escolha Ambientes.

    Uma nova página exibe uma lista de seus ambientes junto com o status e outros detalhes do ambiente.

AWS CLI

Use o AWS CLI para criar e provisionar um ambiente em uma única conta.

Para criar um ambiente, você especifica o ARN do perfil de serviço do AWS Proton , o caminho para o arquivo de especificação, o nome do ambiente, o ARN do modelo de ambiente, as versões principal e secundária e a descrição (opcional).

Os próximos exemplos mostram um arquivo de especificação formatado YAML que especifica valores para duas entradas definidas no arquivo de esquema do modelo de ambiente. Você pode usar o comando `get-environment-template-minor-version` para visualizar o esquema de modelo do ambiente.

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

Crie um ambiente executando o comando a seguir.

```
$ aws proton create-environment \
    --name "MySimpleEnv" \
    --template-name simple-env \
    --template-major-version 1 \
    --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
 \
    --spec "file://env-spec.yaml"
```

Resposta:

```
{
```

```
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateName": "simple-env"
    }
}
```

Depois de criar um novo ambiente, você pode ver uma lista de AWS tags gerenciadas pelo cliente, conforme mostrado no exemplo de comando a seguir. AWS Proton gera automaticamente tags AWS gerenciadas para você. Você também pode modificar e criar etiquetas gerenciadas pelo cliente usando o AWS CLI. Para obter mais informações, consulte AWS Proton recursos e marcação.

Comando:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv"
```

Criar um ambiente em uma conta e provisionar em outra conta

Use o console ou AWS CLI crie um ambiente padrão em uma conta de gerenciamento que provisione a infraestrutura do ambiente em outra conta. O provisionamento é gerenciado pelo AWS.

Antes de usar o console ou a CLI, conclua as seguintes etapas.

1. Identifique-os Conta da AWS IDs para a conta de gerenciamento e ambiente e copie-os para uso posterior.
2. Na conta do ambiente, crie uma função AWS Proton de serviço com permissões mínimas para o ambiente criar. Para obter mais informações, consulte AWS Proton função de serviço para provisionamento usando CloudFormation.

Console de gerenciamento da AWS

Use o console para um ambiente em uma conta e provisionar em outra.

1. Na conta de ambiente, crie uma conexão de conta de ambiente e use-a para enviar uma solicitação de conexão com a conta de gerenciamento.

   a. No console do AWS Proton, escolha Conexões de conta de ambiente no painel de navegação.

   b. Na página Conexões da conta de ambiente, escolha Solicitar conexão.

   > **Note**
   >
   > Verifique se a ID da conta listada no título da página de Conexão da conta de ambiente corresponde à sua ID de conta de ambiente pré-identificada.

   c. Na página Solicitação de conexão, na seção Perfil do ambiente, selecione Perfil de serviço existente e o nome do perfil de serviço que você criou para o ambiente.

   d. Na seção Conectar à conta de gerenciamento, insira o ID da conta de gerenciamento e um nome de ambiente para seu AWS Proton ambiente. Copie o nome para uso posterior.

   e. Escolha Solicitar conexão no canto inferior direito da página.

   f. Sua solicitação aparece como pendente na tabela de conexões de ambiente enviadas para contas de gerenciamento e um modal mostra como aceitar a solicitação da conta de gerenciamento.

2. Na conta de gerenciamento, aceite uma solicitação para se conectar a partir da conta do ambiente.

   a. Faça login na sua conta de gerenciamento e escolha Conexões da conta de ambiente no AWS Proton console.

   b. Na página Conexões da conta de ambiente, na tabela Solicitações de conexão de conta de ambiente, selecione a conexão da conta de ambiente com a ID da conta de ambiente que corresponde à sua ID de conta de ambiente pré-identificada.

> **Note**
>
> Verifique se a ID da conta listada no título da página de Conexão da conta de ambiente corresponde à sua ID de conta de gerenciamento pré-identificada.

c. Escolha Accept (Aceitar). O status muda de PENDENTE para CONECTADO.

3. Na conta de gerenciamento, crie um ambiente.

a. No painel de navegação, escolha Modelos de ambiente.

b. Na página Modelos de ambiente, escolha Criar modelo de ambiente.

c. Na página Escolher um modelo de ambiente, escolha um modelo de ambiente.

d. Na página Configurar ambiente, na seção Provisionamento, escolha Provisionamento gerenciado pela AWS .

e. Na seção Conta de implantação, escolha Outra AWS conta;.

f. Na seção Detalhes do ambiente, selecione a Conexão da conta do ambiente e o Nome do ambiente.

g. Escolha Próximo.

h. Preencha os formulários e escolha Avançar até chegar à página Revisar e criar.

i. Revise e escolha Criar ambiente.

AWS CLI

Use o AWS CLI para criar um ambiente em uma conta e provisionar em outra.

Na conta de ambiente, crie uma conexão de conta de ambiente e solicite a conexão executando o comando a seguir.

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Resposta:

```
{
```

```
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

Na conta de gerenciamento, aceite o pedido de conexão de conta de ambiente conexão executando o comando a seguir.

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Visualize a conexão da sua conta de ambiente executando o comando a seguir.

```
$ aws proton get-environment-account-connection \
```

```
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Na conta de gerenciamento, crie um ambiente executando o comando a seguir.

```
$ aws proton create-environment \
    --name "simple-env-connected" \
    --template-name simple-env-template \
    --template-major-version "1" \
    --template-minor-version "1" \
    --spec "file://simple-env-template/specs/original.yaml" \
    --environment-account-connection-id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-
connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
```

```
        "templateName": "simple-env-template"
    }
}
```

Crie e provisione um ambiente usando o provisionamento autogerenciado

Quando você usa o provisionamento autogerenciado, AWS Proton envia pull requests de provisionamento para um repositório vinculado com sua própria infraestrutura de provisionamento. As pull requests iniciam seu próprio fluxo de trabalho, que chama AWS serviços; para provisionar a infraestrutura.

Considerações sobre provisionamento autogerenciado:

- Antes de criar um ambiente, configure um diretório de recursos do repositório para provisionamento autogerenciado. Para obter mais informações, consulte AWS Proton infraestrutura como arquivos de código.
- Depois de criar o ambiente, AWS Proton espera receber notificações assíncronas sobre o status do provisionamento da sua infraestrutura. Seu código de provisionamento deve usar a AWS Proton `NotifyResourceStateChange` API para enviar essas notificações assíncronas para o. AWS Proton

Você pode usar o provisionamento autogerenciado no console ou com a AWS CLI. Os exemplos a seguir mostram como você pode usar o provisionamento autogerenciado com o Terraform.

Console de gerenciamento da AWS

Use o console para criar um ambiente do Terraform usando provisionamento autogerenciado.

1. No console do AWS Proton, escolha Ambientes.
2. Selecione Criar ambiente.
3. Na página Escolher um modelo de ambiente, selecione um modelo do Terraform e escolha Configurar.
4. Na página Configurar ambiente, na seção Provisionamento, escolha Provisionamento autogerenciado.
5. Na seção Detalhes do repositório de provisionamento:

a. Se você ainda não vinculou seu repositório de aprovisionamento AWS Proton, escolha Novo repositório, escolha um dos provedores de repositório e, em seguida, para CodeStarconexão, escolha uma de suas conexões.

> **Note**
>
> Se você ainda não tiver uma conexão com a conta do provedor de repositório relevante, escolha Adicionar uma nova CodeStar conexão. Em seguida, crie uma conexão e escolha o botão Atualizar ao lado do menu de CodeStar conexão. Agora você deve ser capaz de escolher sua nova conexão no menu.

Se você já vinculou seu repositório ao AWS Proton, escolha Repositório existente.

b. Em Nome do repositório, escolha um repositório. O menu suspenso mostra os repositórios vinculados para o Repositório existente ou a lista de repositórios na conta do provedor para o Novo repositório.

c. Em Nome da ramificação, escolha uma das ramificações do repositório.

6. Na seção Configurações do ambiente, insira um Nome de ambiente.
7. (Opcional) Insira uma descrição do ambiente.
8. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
9. Escolha Próximo.
10. Na página Definir configurações personalizadas do ambiente, você deve inserir valores para os parâmetros `required`. Você pode inserir valores para os parâmetros `optional` ou usar os padrões quando fornecidos.
11. Escolha Avançar e revise suas entradas.
12. Escolha Criar para enviar uma pull request.

    - Se você aprovar a pull request, a implantação está em andamento.
    - Se você rejeitar a pull request, a criação do ambiente será cancelada.
    - Se o tempo limite do pull request expirar, a criação do ambiente não será concluída.
13. Visualize os detalhes e o status do ambiente, bem como as tags AWS gerenciadas e as tags gerenciadas pelo cliente do seu ambiente.
14. No painel de navegação, escolha Ambientes.

Uma nova página exibe uma lista de seus ambientes junto com o status e outros detalhes do ambiente.

AWS CLI

Ao criar um ambiente usando o provisionamento autogerenciado, você adiciona o parâmetro `provisioningRepository` e omite os parâmetros `ProtonServiceRoleArn` e `environmentAccountConnectionId`.

Use o AWS CLI para criar um ambiente Terraform com provisionamento autogerenciado.

1. Crie um ambiente e envie uma pull request ao repositório para análise e aprovação.

   Os próximos exemplos mostram um arquivo de especificação formatado pelo YAML que define os valores para duas entradas com base no arquivo de esquema do modelo de ambiente. Você pode usar o comando `get-environment-template-minor-version` para visualizar o esquema de modelo do ambiente.

   Especificação:

   ```
   proton: EnvironmentSpec
   spec:
     ssm_parameter_value: "test"
   ```

   Crie um ambiente executando o comando a seguir.

   ```
   $ aws proton create-environment \
       --name "pr-environment" \
       --template-name "pr-env-template" \
       --template-major-version "1" \
       --provisioning-repository="branch=main,name=myrepos/env-
   repo,provider=GITHUB" \
       --spec "file://env-spec.yaml"
   ```

   Response:>

   ```
   {
       "environment": {
   ```

```
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T17:06:58.679000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T17:06:58.679000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateName": "pr-env-template"
    }
```

2. Revisar a solicitação

   - Se você aprovar a solicitação, o provisionamento está em andamento.
   - Se você rejeitar a solicitação, a criação do ambiente será cancelada.
   - Se o tempo limite do pull request expirar, a criação do ambiente não será concluída.

3. Forneça de forma assíncrona o status de provisionamento para. AWS Proton O exemplo a seguir notifica sobre AWS Proton um provisionamento bem-sucedido.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

Exibir dados de ambiente

Você pode visualizar os dados detalhados do ambiente usando o AWS Proton console ou AWS CLI o.

Console de gerenciamento da AWS

Você pode visualizar listas de ambientes com detalhes e ambientes individuais com dados detalhados usando o console do AWS Proton.

1. Para ver uma lista dos seus ambientes, escolha Ambientes no painel de navegação.

2. Para exibir dados detalhados, escolha o nome de um ambiente.

   Visualize os dados detalhados do seu ambiente.

AWS CLI

Use os detalhes do ambiente AWS CLI get ou list.

Execute este comando: .

```
$ aws proton get-environment \
    --name "MySimpleEnv"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "lastDeploymentSucceededAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\nspec:\n  my_sample_input: \"the first\"\n
 my_other_sample_input: \"the second\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

Atualizar um ambiente

Se o AWS Proton ambiente estiver associado a uma conexão de conta de ambiente, não atualize nem inclua o `protonServiceRoleArn` parâmetro para atualizar ou conectar-se a uma conexão de conta de ambiente.

Você só pode atualizar para uma nova conexão de conta de ambiente se ambas as condições a seguir forem verdadeiras:

- A conexão da conta de ambiente foi criada na mesma conta de ambiente em que a conexão da conta de ambiente atual foi criada.
- >A conexão da conta do ambiente está associada ao ambiente atual.

Se o ambiente não estiver associado a uma conexão de conta de ambiente, não atualize e nem inclua o parâmetro `environmentAccountConnectionId`.

Você pode atualizar o parâmetro e valor `environmentAccountConnectionId` ou `protonServiceRoleArn`. Não é possível atualizar ambos.

Se seu ambiente usa provisionamento autogerenciado, não atualize o parâmetro `provisioning-repository` e omita os parâmetros `environmentAccountConnectionId` e `protonServiceRoleArn`.

Há quatro modos para atualizar um ambiente, conforme descrito na lista a seguir. Ao usar o AWS CLI, o `deployment-type` campo define o modo. Ao usar o console, esses modos são mapeados para as ações Editar, Atualizar, Atualizar secundárias e Atualizar principais que aparecem no menu em cascata Ações.

`NONE`

Nesse modo, uma implantação não ocorre. Somente as ramificações de metadata solicitadas do são atualizadas.

`CURRENT_VERSION`

Nesse modo, o ambiente é implantado e atualizado com a nova especificação fornecida por você. Somente ramificações solicitadas do são atualizadas. Não inclua parâmetros de versão secundária ou principal ao usar este `deployment-type`.

`MINOR_VERSION`

Nesse modo, o ambiente é implantado e atualizado com a versão secundária publicada e recomendada (mais recente) da versão principal atual em uso por padrão. Você também pode especificar uma versão secundária diferente da versão principal atual em uso.

`MAJOR_VERSION`

Nesse modo, o ambiente é implantado e atualizado com a versão principal e secundária publicada e recomendada (mais recente) do modelo atual por padrão. Você também pode

especificar uma versão principal diferente que seja superior à versão principal em uso e uma versão secundária (opcional).

Tópicos

Atualizar um ambiente de provisionamento AWS gerenciado

O provisionamento padrão só é suportado por ambientes que provisionam com o CloudFormation.

Use o console ou AWS CLI atualize seu ambiente.

Console de gerenciamento da AWS

Execute o comando a seguir para visualizar os parâmetros de ambiente conforme mostrado na tabela a seguir.

1. Escolha uma das duas etapas a seguir.
   a. Na lista de ambientes.
      i. No console do AWS Proton, escolha Ambientes.
      ii. Na lista de ambientes, escolha o botão de rádio à esquerda do ambiente você deseja atualizar.
   b. Na página de detalhes de ambiente do console.
      i. No console do AWS Proton, escolha Ambientes.
      ii. Na lista de ambientes, escolha o nome do ambiente você deseja atualizar.
2. Escolha uma das próximas 4 etapas para atualizar seu ambiente.
   a. Para fazer uma edição que não exija a implantação do ambiente.
      i. Por exemplo, para alterar uma descrição.

         Escolha Editar.
      ii. Preencha o formulário e escolha Avançar.

iii. Revise sua edição e escolha Atualizar.

b. Para fazer atualizações somente nas entradas de metadados.

i. Escolha Ações e, em seguida, Atualizar.

ii. Preencha o formulário e escolha Editar.

iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.

iv. Revise suas atualizações e escolha Atualizar.

c. Para fazer uma atualização para uma nova versão secundária de seu modelo de ambiente.

i. Escolha Ações e, em seguida, Atualizar secundária.

ii. Preencha o formulário e escolha Avançar.

iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.

iv. Revise suas atualizações e escolha Atualizar.

d. Para fazer uma atualização para uma nova versão principal de seu modelo de ambiente.

i. Escolha Ações e, em seguida, Atualizar principal.

ii. Preencha o formulário e escolha Avançar.

iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.

iv. Revise suas atualizações e escolha Atualizar.

AWS CLI

Use o AWS Proton AWS CLI para atualizar um ambiente para uma nova versão secundária.

Execute o seguinte comando para atualizar seu ambiente:

```
$ aws proton update-environment \
        --name "MySimpleEnv" \
        --deployment-type "MINOR_VERSION" \
        --template-major-version "1" \
        --template-minor-version "1" \
        --proton-service-role-arn arn:aws:iam::123456789012:role/service-
role/ProtonServiceRole \
        --spec "file:///spec.yaml"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:29:55.472000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

Execute o comando a seguir para obter e confirmar o status:

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "MySimpleEnv",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Atualizar um ambiente de provisionamento autogerenciado

O provisionamento autogerenciado só é suportado por ambientes que provisionam com o Terraform.

Use o console ou AWS CLI atualize seu ambiente.

Console de gerenciamento da AWS

Execute o comando a seguir para visualizar os parâmetros de ambiente conforme mostrado na tabela a seguir.

1. Escolha uma das duas etapas a seguir.
   a. Na lista de ambientes.
      i. No console do AWS Proton, escolha Ambientes.
      ii. Na lista de ambientes, escolha o botão de rádio à esquerda do modelo de ambiente você deseja atualizar.
   b. Na página de detalhes de ambiente do console.
      i. No console do AWS Proton, escolha Ambientes.
      ii. Na lista de ambientes, escolha o nome do ambiente você deseja atualizar.
2. Escolha uma das próximas 4 etapas para atualizar seu ambiente.
   a. Para fazer uma edição que não exija a implantação do ambiente.
      i. Por exemplo, para alterar uma descrição.

         Escolha Editar.
      ii. Preencha o formulário e escolha Avançar.
      iii. Revise sua edição e escolha Atualizar.
   b. Para fazer atualizações somente nas entradas de metadados.
      i. Escolha Ações e, em seguida, Atualizar.
      ii. Preencha o formulário e escolha Editar.
      iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.
      iv. Revise suas atualizações e escolha Atualizar.

c. Para fazer uma atualização para uma nova versão secundária de seu modelo de ambiente.

   i. Escolha Ações e, em seguida, Atualizar secundária.

   ii. Preencha o formulário e escolha Avançar.

   iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.

   iv. Revise suas atualizações e escolha Atualizar.

d. Para fazer uma atualização para uma nova versão principal de seu modelo de ambiente.

   i. Escolha Ações e, em seguida, Atualizar principal.

   ii. Preencha o formulário e escolha Avançar.

   iii. Preencha os formulários e escolha Avançar até chegar à página Revisar.

   iv. Revise suas atualizações e escolha Atualizar.

AWS CLI

Use o AWS CLI para atualizar um ambiente Terraform para uma nova versão secundária com provisionamento autogerenciado.

1. Execute o seguinte comando para atualizar seu ambiente:

```
$ aws proton update-environment \
    --name "pr-environment" \
    --deployment-type "MINOR_VERSION" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --provisioning-repository "branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec-mod.yaml"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "IN_PROGRESS",
```

```
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:09:15.745000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "pr-env-template"
    }
}
```

2. Execute o comando a seguir para obter e confirmar o status:

```
$ aws proton get-environment \
    --name "pr-environment"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:25:41.998000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "spec": "proton: EnvironmentSpec\nspec:\n   ssm_parameter_value: \"test
\"\n ssm_another_parameter_value: \"update\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
```

```
        "templateName": "pr-env-template"
    }
}
```

3. Revise a pull request que foi enviada por AWS Proton.

   - Se você aprovar a solicitação, o provisionamento está em andamento.
   - Se você rejeitar a solicitação, a criação do ambiente será cancelada.
   - Se o tempo limite do pull request expirar, a criação do ambiente não será concluída.

4. Forneça o status de provisionamento para. AWS Proton

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Cancela uma implantação de ambiente em progresso

Você pode tentar cancelar a implantação de uma atualização de ambiente se a `deploymentStatus` estiver ativada`IN_PROGRESS`. AWS Proton tentativas de cancelar a implantação. O cancelamento bem-sucedido não é garantido.

Quando você cancela uma implantação de atualização, AWS Proton tenta cancelar a implantação conforme listado nas etapas a seguir.

Com o provisionamento AWS gerenciado, AWS Proton faça o seguinte:

- Define o estado de implantação como `CANCELLING`.
- Interrompe a implantação em andamento e exclui todos os novos recursos que foram criados pela implantação quando `IN_PROGRESS`.
- Define o estado de implantação como `CANCELLED`.
- Reverte o estado do recurso para o que era antes do início da implantação.

Com o provisionamento autogerenciado, AWS Proton faça o seguinte:

- Tenta fechar a pull request para evitar a mesclagem das alterações no seu repositório.
- Define o estado de implantação para `CANCELLED` se a pull request foi fechada com sucesso.

Para obter instruções sobre como cancelar a implantação de um ambiente, consulte CancelEnvironmentDeployment a Referência AWS Proton da API.

Você pode usar o console ou a CLI para cancelar ambientes que estão em andamento.

Console de gerenciamento da AWS

Use o console para cancelar uma implantação de atualização de ambiente, conforme mostrado nas etapas seguintes.

1. No console do AWS Proton, escolha Ambientes no painel de navegação.
2. Na lista de ambientes, escolha o nome do ambiente com a atualização de implantação que você deseja cancelar.
3. Se o status de implantação da atualização estiver Em andamento, na página de detalhes do ambiente, escolha Ações e, em seguida, Cancelar implantação.
4. Você será solicitado a confirmar que você deseja cancelar esses objetos. Escolha Cancelar implantação.
5. O status de implantação da atualização será definido como Cancelando e, em seguida, Cancelado para concluir o cancelamento.

AWS CLI

Use o AWS Proton AWS CLI para cancelar uma implantação de atualização do ambiente IN_PROGRESS para uma nova versão secundária 2.

Uma condição de espera é incluída no modelo usado neste exemplo para que o cancelamento comece antes que a implantação da atualização seja bem-sucedida.

Execute o comando a seguir para cancelar a atualização.

```
$ aws proton cancel-environment-deployment \
        --environment-name "MySimpleEnv"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
```

```
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Execute o comando a seguir para obter e confirmar o status:”

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Exclua um ambiente

Você pode excluir um AWS Proton ambiente usando o AWS Proton console ou AWS CLI o.

> **Note**
>
> Você não pode excluir um ambiente que tenha qualquer componente associado. Para excluir esse ambiente, você deve primeiro excluir todos os componentes que estão sendo executados no ambiente. Para obter mais informações sobre componentes, consulte Componentes.

Console de gerenciamento da AWS

Exclua um ambiente usando o console conforme descrito nas duas opções a seguir.

Na lista de ambientes.

1. No console do AWS Proton, escolha Ambientes.
2. Na lista de ambientes, selecione o botão de rádio à esquerda do ambiente você deseja excluir.
3. Escolha Ações e então Excluir.
4. Um modal pede que você confirme a ação de exclusão.
5. Siga as instruções e escolha Sim, excluir.

Na página de detalhes do ambiente.

1. No console do AWS Proton, escolha Ambientes.
2. Na lista de ambientes, escolha o nome do ambiente você deseja excluir.
3. Na página de detalhes do ambiente, escolha Ações e, em seguida, Excluir.
4. Você será solicitado a confirmar que você deseja excluir esses objetos.
5. Siga as instruções e escolha Sim, excluir.

AWS CLI

Use o AWS CLI para excluir um ambiente.

Não exclua um ambiente se serviços ou instâncias de serviço forem implantados no ambiente.

Execute este comando: .

```
$ aws proton delete-environment \
    --name "MySimpleEnv"
```

Resposta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "DELETE_IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Conexões de conta de ambiente

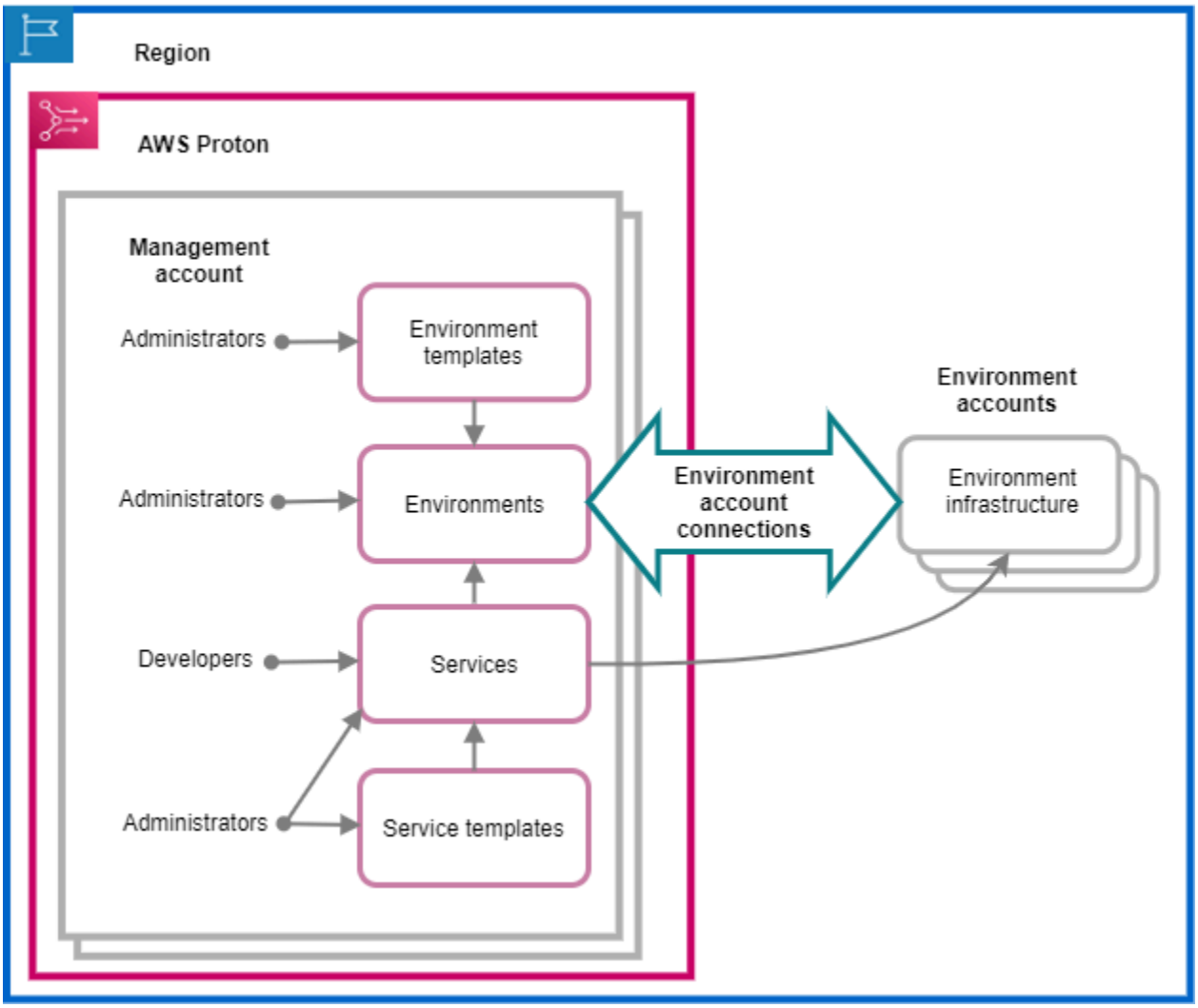
Visão geral

Saiba como criar e gerenciar um AWS Proton ambiente em uma conta e provisionar seus recursos de infraestrutura em outra conta. Isso pode ajudar a melhorar a visibilidade e a eficiência em grande escala. As conexões de conta do ambiente oferecem suporte apenas ao provisionamento padrão com infraestrutura do CloudFormation como código.

> **Note**
>
> As informações neste tópico são relevantes para ambientes configurados com provisionamento gerenciado pelo AWS . Com ambientes configurados com provisionamento autogerenciado, AWS Proton não provisiona diretamente sua infraestrutura. Em vez disso, ele envia pull requests (PRs) ao seu repositório para provisionamento. É sua

responsabilidade garantir que seu código de automação assuma a identidade e o perfil corretos.
Para obter mais informações sobre métodos de provisionamento, consulte the section called "Métodos de provisionamento".

Terminologia



Com conexões de conta de AWS Proton ambiente, você pode criar um AWS Proton ambiente a partir de uma conta e provisionar sua infraestrutura em outra conta.

conta gerencial

A conta única em que você, como administrador, cria um AWS Proton ambiente que provisiona recursos de infraestrutura em outra conta de ambiente.

Conta do ambiente

Uma conta na qual a infraestrutura de ambiente é provisionada, quando você cria um ambiente de AWS Proton em outra conta.

Conexão de conta de ambiente

Uma conexão bidirecional segura entre uma conta de gerenciamento e uma conta de ambiente. Mantém a autorização e as permissões conforme descritas mais adiante nas seções seguintes.

Quando você cria uma conexão de conta de ambiente em uma conta de ambiente em uma região específica, somente as contas de gerenciamento na mesma região podem ver e usar a conexão da conta de ambiente. Isso significa que o AWS Proton ambiente criado na conta de gerenciamento e a infraestrutura ambiental provisionada na conta de ambiente devem estar na mesma região.

Considerações sobre a conexão da conta de ambiente

- Você precisa de uma conexão de conta de ambiente para cada ambiente que você deseja provisionar em uma conta de ambiente.
- Para obter informações sobre as cotas de conexão da conta de ambiente, consulte AWS Proton cotas.

Tags

Na conta do ambiente, use o console ou o AWS CLI para visualizar e gerenciar as tags gerenciadas pelo cliente da conexão da conta do ambiente. AWS tags gerenciadas não são geradas para conexões de contas de ambiente. Para obter mais informações, consulte Tags.

Criar um ambiente em uma conta e provisionar sua infraestrutura em outra conta

Para criar e provisionar um ambiente a partir de uma única conta de gerenciamento, configure uma conta de ambiente para um ambiente que você planeja criar.

Inicie na conta do ambiente e crie uma conexão.

Na conta de ambiente, crie uma função AWS Proton de serviço com escopo reduzido somente às permissões necessárias para provisionar os recursos de infraestrutura do seu ambiente. Para obter mais informações, consulte AWS Proton função de serviço para provisionamento usando CloudFormation.

Em seguida, crie e envie uma solicitação de conexão de conta de ambiente para sua conta de gerenciamento. Quando a solicitação for aceita, AWS Proton pode usar a função do IAM associada que permite o provisionamento de recursos do ambiente na conta do ambiente associada.

Na conta de gerenciamento, aceite ou rejeite a conexão da conta do ambiente.

Na conta de gerenciamento, aceite ou rejeite a solicitação de conexão da conta do ambiente. Você não pode excluir uma conexão de conta de ambiente da sua conta de gerenciamento.

Se você aceitar a solicitação, eles AWS Proton poderão usar a função do IAM associada que permite o provisionamento de recursos na conta do ambiente associada.

Os recursos da infraestrutura do ambiente são provisionados na conta do ambiente associada. Você só pode usar AWS Proton APIs para acessar e gerenciar seu ambiente e seus recursos de infraestrutura, a partir da sua conta de gerenciamento. Para obter mais informações, consulte Criar um ambiente em uma conta e provisionar em outra conta e Atualizar um ambiente.

Depois de rejeitar uma solicitação, você não poderá aceitar nem usar a conexão da conta de ambiente rejeitada.

> **Note**
>
> Você não pode rejeitar uma conexão de conta de ambiente conectada a um ambiente. Para rejeitar a conexão da conta do ambiente, você deve primeiro excluir o ambiente associado.

Na conta do ambiente, acesse os recursos de infraestrutura provisionados.

Na conta do ambiente, você pode visualizar e acessar os recursos de infraestrutura provisionados. Por exemplo, você pode usar ações de CloudFormation API para monitorar e limpar pilhas, se necessário. Você não pode usar as ações da AWS Proton API para acessar ou gerenciar o AWS Proton ambiente usado para provisionar os recursos de infraestrutura.

Na conta do ambiente, você pode excluir as conexões da conta do ambiente que você criou na conta do ambiente. Você não pode aceitá-los ou rejeitá-los. Se você excluir uma conexão de conta de ambiente que está sendo usada por um AWS Proton ambiente, AWS Proton não poderá gerenciar os recursos de infraestrutura do ambiente até que uma nova conexão de ambiente seja aceita para a conta do ambiente e o ambiente nomeado. Você é responsável por limpar os recursos provisionados que permanecem sem uma conexão com o ambiente.

Use o console ou a CLI para gerenciar as conexões da conta do ambiente

Você pode usar o console ou a CLI para criar e gerenciar as conexões da conta do ambiente.

Console de gerenciamento da AWS

Use o console para criar uma conexão de conta do ambiente e enviar uma solicitação à conta de gerenciamento, conforme mostrado nas próximas etapas.

1. Escolha um nome para o ambiente que você planeja criar em sua conta de gerenciamento ou escolha o nome de um ambiente existente que exija uma conexão de conta de ambiente.
2. Em uma conta de ambiente, no console do AWS Proton, escolha Conexões de conta de ambiente no painel de navegação.
3. Na página Conexões da conta de ambiente, escolha Solicitar conexão.

> **Note**
>
> Verifique a ID da conta que está listada no título da página de Conexão da conta de ambiente. Verifique se ele corresponde ao ID da conta de ambiente na qual você deseja que seu ambiente nomeado seja provisionado.

4. Na página Solicitação de conexão:

   a. Na seção Conectar à conta de gerenciamento, insira a ID da conta de gerenciamento e o Nome de ambiente que inseriu na etapa 1.

   b. Na seção Função de ambiente, escolha Nova função de serviço e cria AWS Proton automaticamente uma nova função para você. Ou selecione Perfil de serviço existente e o nome do perfil de serviço que você criou anteriormente.

> **Note**
>
> A função criada AWS Proton automaticamente para você tem amplas permissões. Recomendamos que você defina o perfil de acordo com as permissões necessárias para provisionar os recursos de infraestrutura do seu ambiente. Para obter mais informações, consulte AWS Proton função de serviço para provisionamento usando CloudFormation.

c. (Opcional) Na seção Tags, escolha Adicionar nova tag para criar uma tag gerenciada pelo cliente para sua conexão com a conta do ambiente.

d. Escolha Solicitar conexão.

5. Sua solicitação aparece como pendente na tabela de conexões de ambiente enviadas para contas de gerenciamento e um modal informa como aceitar a solicitação da conta de gerenciamento.

Aceite ou rejeite uma solicitação de conexão da conta do ambiente.

1. Em uma conta de gerenciamento, no console do AWS Proton, escolha Conexões de conta de ambiente no painel de navegação.
2. Na página Conexões de conta de ambiente, na tabela Solicitações de conexão de conta de ambiente, escolha a solicitação de conexão de ambiente a ser aceita ou rejeitada.

> **Note**
>
> Verifique a ID da conta que está listada no título da página de Conexão da conta de ambiente. Verifique se ele corresponde à ID da conta de gerenciamento associada à conexão da conta de ambiente a ser rejeitada. Depois que você rejeitar essa conexão de conta de ambiente, não poderá aceitar ou usar a conexão de conta de ambiente rejeitada.

3. Escolha Rejeitar ou Aceitar.

- Se você selecionou Rejeitar, o status mudará de pendente para rejeitado.
- Se você selecionou Aceitar, o status mudará de pendente para conectado.

Excluir uma conexão de conta de ambiente

1. Em uma conta de ambiente, no console do AWS Proton, escolha Conexões de conta de ambiente no painel de navegação.

> **Note**
>
> Verifique a ID da conta que está listada no título da página de Conexão da conta de ambiente. Verifique se ele corresponde à ID da conta de gerenciamento associada à conexão da conta de ambiente a ser rejeitada. Depois de excluir essa conexão

de conta de ambiente, não é AWS Proton possível gerenciar os recursos de infraestrutura de ambiente na conta de ambiente. Ele só pode gerenciá-lo depois que uma nova conexão de conta de ambiente para a conta de ambiente e o ambiente nomeado forem aceitos pela conta de gerenciamento.

2. Na página Conexões da conta de ambiente, na seção Solicitações enviadas para se conectar à conta de gerenciamento, escolha Excluir.
3. Você será solicitado a confirmar que você deseja excluir esses objetos. Escolha Excluir.

AWS CLI

Escolha um nome para o ambiente que você planeja criar em sua conta de gerenciamento ou escolha o nome de um ambiente existente que exija uma conexão de conta de ambiente.

Crie uma conexão de conta de ambiente em uma conta de ambiente.

Execute este comando: .

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role",
        "status": "PENDING"
    }
```

```
}
```

Aceite ou rejeite uma conexão de conta de ambiente em uma conta de gerenciamento, conforme mostrado no comando e na resposta a seguir.

> **Note**
>
> Se você rejeitar essa conexão de conta de ambiente, não poderá aceitar ou usar a conexão de conta de ambiente rejeitada.

Se você especificar Rejeitar, o status mudará de pendente para rejeitado.

Se você especificar Aceitar, o status mudará de pendente para conectado.

Execute o comando a seguir para aceitar a conexão da conta do ambiente:

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Execute o comando a seguir para rejeitar a conexão da conta do ambiente:

```
$ aws proton reject-environment-account-connection \
```

```
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-reject",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role"
    }
}
```

Visualizar as conexões de conta de um ambiente. Você pode obter ou listar conexões de contas do ambiente.

Execute o seguinte comando get:

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
```

```
        "status": "CONNECTED"
    }
}
```

Exclua uma conexão de conta de ambiente em uma conta de ambiente.

> **Note**
>
> Se você excluir essa conexão de conta de ambiente, AWS Proton não poderá gerenciar os recursos de infraestrutura de ambiente na conta de ambiente até que uma nova conexão de ambiente tenha sido aceita para a conta de ambiente e o ambiente nomeado. Você é responsável por limpar os recursos provisionados que permanecem sem uma conexão com o ambiente.

Execute este comando: .

```
$ aws proton delete-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Resposta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Ambientes gerenciados pelo cliente

Com ambientes gerenciados pelo cliente, você pode usar a infraestrutura existente, como uma VPC, que você já implantou como seu ambiente. AWS Proton Ao usar ambientes gerenciados pelo cliente, você pode provisionar seus próprios recursos compartilhados fora do. AWS Proton No entanto, você ainda pode AWS Proton permitir o consumo de saídas de provisionamento relevantes como entradas para AWS Proton serviços quando eles forem implantados. Se as saídas puderem mudar, AWS Proton é capaz de aceitar atualizações. AWS Proton No entanto, não é capaz de alterar o ambiente diretamente, pois o provisionamento é gerenciado externamente. AWS Proton

Depois que o ambiente é criado, você é responsável por fornecer as mesmas saídas AWS Proton que teriam sido criadas se AWS Proton tivesse criado o ambiente, como nomes de cluster do Amazon ECS ou Amazon VPC. IDs

Com essa funcionalidade, você pode implantar e atualizar recursos de AWS Proton serviço de um modelo de AWS Proton serviço para esse ambiente. No entanto, o ambiente em si não é modificado por meio de atualizações de modelos em AWS Proton. Você é responsável por executar atualizações no ambiente e atualizar essas saídas no. AWS Proton

Você pode ter vários ambientes em uma única conta que são uma combinação de ambientes AWS Proton gerenciados e gerenciados pelo cliente. Você também pode vincular uma segunda conta e usar um AWS Proton modelo na conta principal para executar implantações e atualizações em ambientes e serviços nessa segunda conta vinculada.

Como usar ambientes gerenciados pelo cliente

A primeira coisa que os administradores precisam fazer é registrar um modelo de ambiente importado e gerenciado pelo cliente. Não forneça manifestos ou arquivos de infraestrutura no pacote de modelos. Forneça somente o esquema.

O esquema abaixo descreve uma lista de saídas usando o formato de API aberta e replica as saídas de um modelo. CloudFormation

> **Important**
>
> Somente entradas de string são permitidas para as saídas.

O exemplo a seguir é um trecho das seções de saída de um CloudFormation modelo para um modelo Fargate correspondente.

```
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

O esquema do ambiente AWS Proton importado correspondente é semelhante ao seguinte. Não forneça valores padrão no esquema.

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentOutput"
  types:
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:
          type: string
          description: "The ARN of the ECS role"
        VpcId:
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

No momento do registro do modelo, você indica que esse modelo foi importado e fornece a localização do bucket do Amazon S3 para o pacote. AWS Proton valida se o esquema contém somente `environment_input_type` e nenhum parâmetro de CloudFormation modelo antes de colocar o modelo no rascunho.

Você fornece o seguinte para criar um ambiente importado.

- Um perfil do IAM para usar ao fazer implantações.
- Uma especificação com os valores das saídas necessárias.

Você pode fornecer os dois por meio do console ou AWS CLI usando um processo semelhante à implantação de um ambiente normal.

CodeBuild criação de função de provisionamento

Ferramentas de infraestrutura como código (IAAC), como CloudFormation o Terraform, exigem permissões para os diversos tipos de recursos. AWS Por exemplo, se um modelo do IAAC declarar um bucket do Amazon S3, ele precisará de permissões para criar, ler, atualizar e excluir buckets do Amazon S3. É considerado uma prática recomendada de segurança limitar os perfis às permissões mínimas necessárias. Dada a variedade de AWS recursos, é difícil criar políticas de privilégios mínimos para modelos de IAAC, especialmente quando os recursos gerenciados por esses modelos podem ser alterados posteriormente. Por exemplo, em suas últimas edições em um modelo que está sendo gerenciado por AWS Proton, você adiciona um recurso de banco de dados do RDS.

Configurar as permissões corretas ajuda a facilitar as implantações do seu IaC. AWS Proton CodeBuild O provisionamento executa comandos CLI arbitrários fornecidos pelo cliente em CodeBuild um projeto localizado na conta do cliente. Normalmente, esses comandos criam e excluem a infraestrutura usando uma ferramenta de Infraestrutura como Código (IAAC), como AWS CDK. Quando um AWS recurso é implantado cujo modelo usa CodeBuild Provisioning, AWS iniciará uma construção em um CodeBuild projeto gerenciado por. AWS Uma função é passada para CodeBuild, que CodeBuild pressupõe a execução de comandos. Essa função, chamada de função de CodeBuild provisionamento, é fornecida pelo cliente e contém as permissões necessárias para provisionar a infraestrutura. É para ser assumido apenas por CodeBuild e nem mesmo AWS Proton posso presumir.

Criar a função

A função CodeBuild de aprovisionamento pode ser criada no console do IAM ou no. AWS CLI Para criá-lo no AWS CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AWSProtonCodeBuildProvisioningBasicAccess
```

Isso também anexa o`AWSProtonCodeBuildProvisioningBasicAccess`, que contém as permissões mínimas necessárias para o CodeBuild serviço executar uma compilação.

Se você preferir usar o console, certifique-se do seguinte ao criar o perfil:

1. Para entidade confiável, selecione AWS serviço e, em seguida, selecione CodeBuild.
2. Na etapa Adicionar permissões, selecione `AWSProtonCodeBuildProvisioningBasicAccess` e quaisquer outras políticas que você deseja anexar.

Acesso de administrador

Se você anexar a `AdministratorAccess` política à função de CodeBuild provisionamento, isso garantirá que qualquer modelo do IAAC não falhe devido à falta de permissões. Isso também significa que qualquer pessoa que possa criar um modelo de ambiente ou modelo de serviço pode realizar ações em nível de administrador, mesmo que esse usuário não seja administrador. AWS Proton não recomenda o uso `AdministatorAccess` com a função de CodeBuild aprovisionamento. Se você decidir usar `AdministratorAccess` com a função de CodeBuild provisionamento, faça isso em um ambiente sandbox.

Você pode criar um perfil com o `AdministratorAccess` no console do IAM ou executando este comando:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

Criação de um perfil com escopo mínimo

Se você quer criar um perfil com o mínimo de permissões, há várias abordagens:

- Implante com permissões de administrador e, em seguida, defina o perfil. Recomendamos usar o IAM Access Analyzer.
- Use políticas gerenciadas para dar acesso aos serviços que você planeja usar.

AWS CDK

Se você está usando AWS CDK com AWS Proton e já executou `cdk bootstrap` em cada conta/região do ambiente, então já existe uma função para. `cdk deploy` Nesse caso, anexe a seguinte política à função de CodeBuild aprovisionamento:

```
{
    "Action": "sts:AssumeRole",
    "Resource": [
        "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
        "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
    ],
    "Effect": "Allow"
}
```

VPC personalizado

Se você decidir executar CodeBuild em uma VPC personalizada, precisará das seguintes permissões em sua CodeBuild função:

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:network-interface/*",
        "arn:aws:ec2:region:account-id:subnet/*",
        "arn:aws:ec2:region:account-id:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:*/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
```

```
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:AuthorizedService": "codebuild.amazonaws.com"
        }
    }
}
```

Você também pode usar a política gerenciada pelo `AmazonEC2FullAccess`, embora isso inclua permissões que talvez você não precise. Para anexar a política gerenciada isando a CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

AWS Proton serviços

Um AWS Proton serviço é uma instanciação de um modelo de serviço, normalmente incluindo várias instâncias de serviço e um pipeline. Uma instância AWS Proton de serviço é uma instanciação de um modelo de serviço em um ambiente específico. Um modelo de serviço é uma definição completa da infraestrutura e do pipeline de serviços opcional para um AWS Proton serviço.

Depois de implantar suas instâncias de serviço, você pode atualizá-las por meio de push de código-fonte que acionam o CI/CD pipeline ou atualizando o serviço para novas versões de seu modelo de serviço. AWS Proton avisa quando novas versões de seu modelo de serviço são disponibilizadas para que você possa atualizar seus serviços para uma nova versão. Quando seu serviço é atualizado, AWS Proton reimplanta o serviço e as instâncias do serviço.

Este capítulo mostra como gerenciar serviços usando operações de criação, visualização, atualização e exclusão. Para obter informações adicionais, consulte The AWS Proton Service API Reference.

Tópicos

Criar um serviço

Para implantar um aplicativo com AWS Proton, como desenvolvedor, você cria um serviço e fornece as seguintes entradas.

1. O nome de um modelo de AWS Proton serviço publicado pela equipe da plataforma.
2. Um nome para o serviço.
3. O número de instâncias de serviço que você deseja implantar.

4. Uma seleção de ambientes que você deseja usar.
5. Uma conexão com seu repositório de código se você estiver usando um modelo de serviço que inclua um pipeline de serviço (opcional).

O que há em um serviço?

Ao criar um AWS Proton serviço, você pode escolher entre dois tipos diferentes de modelos de serviço:

- Um modelo de serviço que inclui um pipeline de serviço (padrão).
- Um modelo de serviço que não inclui um pipeline de serviço.

É necessário criar pelo menos uma instância de serviço ao criar o serviço.

Uma instância de serviço e um pipeline opcional estão associados a um serviço. Você só pode criar ou excluir um pipeline dentro do contexto das ações de criação e exclusão do serviço. Para saber como adicionar e remover instâncias de um serviço, consulte Editar um serviço.

> **Note**
>
> Seu ambiente está configurado para provisionamento autogerenciado ou autogerenciado. AWS AWS Proton provisiona serviços em um ambiente usando o mesmo método de provisionamento usado pelo ambiente. O desenvolvedor que cria ou atualiza instâncias de serviço não vê a diferença e sua experiência é a mesma em ambos os casos.
> Para obter mais informações sobre métodos de provisionamento, consulte the section called “Métodos de provisionamento”.

Modelos de serviço

As versões principais e secundárias dos modelos de serviço estão disponíveis. Ao usar o console, você seleciona a versão principal do `Recommended` e secundária mais recente do modelo de serviço. Ao usar o AWS CLI e especificar somente a versão principal do modelo de serviço, você especifica implicitamente sua versão `Recommended` secundária mais recente.

A seguir, descrevemos a diferença entre as versões principais e secundárias do modelo e seu uso.

- Novas versões de um modelo se tornam `Recommended` assim que são aprovadas por um membro da equipe da plataforma. Isso significa que novos serviços são criados usando essa versão e você é solicitado a atualizar os serviços existentes para a nova versão.
- Por meio AWS Proton disso, a equipe da plataforma pode atualizar automaticamente as instâncias de serviço para uma nova versão secundária de um modelo de serviço. As versões secundárias devem ser compatíveis com versões anteriores.
- Como as versões principais exigem que você forneça novas entradas como parte do processo de atualização, você precisa atualizar seu serviço para uma versão principal do modelo de serviço. As versões principais não são compatíveis com versões anteriores.

Criar um serviço

Os procedimentos a seguir mostram como usar o AWS Proton console ou AWS CLI criar um serviço com ou sem um pipeline de serviço.

Console de gerenciamento da AWS

Crie um serviço conforme mostrado nas etapas do console a seguir.

1. No console do AWS Proton, selecione Serviços.
2. Escolha Create service.
3. Na página Escolher um modelo de serviço, selecione um modelo e escolha Configurar.

   Quando você não quiser usar um pipeline habilitado, escolha um modelo marcado com Exclui pipeline para seu serviço.
4. Na página Configurar serviço, na seção Configurações do serviço, insira o Nome do serviço.
5. (Opcional) Insira uma descrição para o serviço.
6. Na seção Configurações do repositório de serviços:

   a. Em CodeStar Conexão, escolha sua conexão na lista.

   b. Para ID do Repositório, escolha o nome do seu repositório de códigos-fonte da lista.

   c. Em Nome da ramificação, escolha o nome da ramificação do repositório de código-fonte na lista.
7. (Opcional) Na seção Tags, escolha Adicionar nova tag e insira uma chave e um valor para criar uma tag gerenciada pelo cliente.
8. Escolha Próximo.

9. Na página Definir configurações personalizadas, na seção Instâncias de serviço, escolha a seção Nova instância. Você deve inserir valores para os parâmetros do `required`. Você pode inserir valores para os parâmetros `optional` ou usar os padrões quando fornecidos.
10. Na seção Entradas do pipeline, você deve inserir valores para os parâmetros do `required`. Você pode inserir valores para os parâmetros `optional` ou usar os padrões quando fornecidos.
11. Escolha Avançar e revise suas entradas.
12. Escolha Criar.

    Visualize os detalhes e o status do serviço, bem como as tags AWS gerenciadas e as tags gerenciadas pelo cliente do seu serviço.
13. No painel de navegação, escolha Serviços.

    Uma nova página exibe uma lista de seus serviços junto com o status e outros detalhes do serviço.

AWS CLI

Ao usar o AWS CLI, você especifica as entradas de serviço em um `spec` arquivo formatado em YAML,`.aws-proton/service.yaml`, localizado no diretório do código-fonte.

Você pode usar o comando CLI `get-service-template-minor-version` para visualizar os parâmetros obrigatórios e opcionais do esquema para os quais você fornece valores em seu arquivo de especificação.

Se você quiser usar um modelo de serviço que tenha `pipelineProvisioning: "CUSTOMER_MANAGED"`, não inclua a seção `pipeline:` em sua especificação e não incluaos parâmetros `-repository-connection-arn`, `-repository-id` e `-branch-name` em seu comando `create-service`.

Crie um serviço com um pipeline de serviço, conforme mostrado nas etapas da CLI a seguir.

1. Configure o perfil de serviço para o pipeline conforme mostrado no comando de exemplo da CLI a seguir.

   Comando:

   ```
   $ aws proton update-account-settings \
   ```

```
        --pipeline-service-role-arn "arn:aws:iam::123456789012:role/AWS
 ProtonServiceRole"
```

2. A lista a seguir mostra um exemplo de especificação, com base no esquema do modelo de serviço, que inclui o pipeline de serviço e as entradas da instância.

   Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_required_input: "hello"
  my_sample_pipeline_optional_input: "bye"

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

   Crie um serviço com um pipeline, conforme mostrado no exemplo de comando e resposta da CLI a seguir.

   Comando:

```
$ aws proton create-service \
        --name "MySimpleService" \
        --branch-name "mainline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --repository-connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
        --repository-id "myorg/myapp" \
        --spec "file://spec.yaml"
```

   Resposta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
```

```
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Crie um serviço sem um pipeline de serviço, conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Veja a seguir um exemplo de especificação que não inclui entradas do pipeline de serviço.

Especificação:

```
proton: ServiceSpec

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

Para criar um serviço sem um pipeline de serviço provisionado, você fornece o caminho para um **`spec.yaml`** e não inclui os parâmetros do repositório, conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Comando:

```
$ aws proton create-service \
        --name "MySimpleServiceNoPipeline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --spec "file://spec-no-pipeline.yaml"
```

Resposta:

```
{
```

```
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/
MySimpleServiceNoPipeline",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleServiceNoPipeline",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service-no-pipeline"
    }
}
```

Exibir dados do serviço

Você pode visualizar e listar dados detalhados do serviço usando o AWS Proton console ou AWS CLI o.

Console de gerenciamento da AWS

Liste e visualize os detalhes do serviço usando o console do AWS Proton, conforme mostrado nas etapas a seguir.

1. Para ver uma lista dos seus serviços, escolha Serviços no painel de navegação.
2. Para visualizar dados detalhados, escolha o nome de um serviço.

   Visualize os dados detalhados do seu serviço.

AWS CLI

Visualize os detalhes de um serviço com um pipeline de serviço, conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Comando:

```
$ aws proton get-service \
    --name "simple-svc"
```

Resposta:

```
{
    "service": {
```

```
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "repositoryId": "myorg/myapp",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```


Visualize os detalhes de um serviço sem um pipeline de serviço, conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Comando:

```
$ aws proton get-service \
    --name "simple-svc-no-pipeline"
```

Resposta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc-without-
pipeline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc-without-pipeline",
        "spec": "proton: ServiceSpec\ninstances:\n- name: instance-svc-simple\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_required_input:
 hi\n    my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple-no-pipeline"
    }
}
```


Editar um serviço

Você pode fazer as seguintes edições em um AWS Proton serviço.

- Editar a descrição do serviço.
- Edite um serviço adicionando e removendo instâncias de serviço.

Editar a descrição do serviço.

Você pode usar o console ou o AWS CLI para editar a descrição de um serviço.

Console de gerenciamento da AWS

Edite um serviço usando o console conforme descrito nas etapas a seguir.

Na lista de serviços.

1. No console do AWS Proton, selecione Serviços.
2. Na lista de serviços, escolha o botão de rádio à esquerda do serviço que deseja atualizar.
3. Escolha Editar.
4. Na página Configurar serviço, preencha o formulário e escolha Avançar.
5. Na página Definir configurações personalizadas, escolha Avançar.
6. Revise suas edições e escolha Salvar alterações.

Na página de detalhes do serviço.

1. No console do AWS Proton, selecione Serviços.
2. Na lista de serviços, escolha o nome do serviço que deseja editar.
3. Na página de detalhes do serviço, escolha Editar.
4. Na página Configurar serviço, preencha o formulário e escolha Avançar.
5. Na página Configurar ajustes personalizados, preencha o formulário e escolha Avançar.
6. Revise suas edições e escolha Salvar alterações.

AWS CLI

Edite uma descrição conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Comando:

```
$ aws proton update-service \
    --name "MySimpleService" \
    --description "Edit by updating description"
```

Resposta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2021-03-12T22:44:21.975000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "ACTIVE",
        "templateName": "fargate-service"
    }
}
```

Editar um serviço para adicionar ou remover instâncias de serviço

Para um AWS Proton serviço, você pode adicionar ou excluir instâncias de serviço enviando uma especificação editada. As seguintes condições devem ser satisfeitas para que uma solicitação seja bem-sucedida:

- Seu serviço e pipeline ainda não estão sendo editados ou excluídos quando você envia a solicitação de edição.
- Sua especificação editada não inclui edições que modifiquem o pipeline de serviço ou edições em instâncias de serviço existentes que não devem ser excluídas.
- Sua especificação editada não remove nenhuma instância de serviço existente que tenha um componente anexado. Para excluir essa instância de serviço, você deve primeiro atualizar o componente para desanexá-lo de sua instância de serviço. Para obter mais informações sobre componentes, consulte Componentes.

As instâncias com falha de exclusão são instâncias de serviço no estado `DELETE_FAILED`. Quando você solicita uma edição de serviço, AWS Proton tenta remover as instâncias que falharam na exclusão para você, como parte do processo de edição. Se alguma de suas instâncias de serviço falhar na exclusão, ainda poderá haver recursos associados às instâncias, mesmo que eles não estejam visíveis no console ou no AWS CLI. Verifique os recursos de infraestrutura da instância que falharam na exclusão e limpe-os para que AWS Proton você possa removê-los.

Para a cota de instâncias de serviço de um serviço, consulte AWS Proton cotas. Você também deve manter pelo menos uma instância de serviço para seu serviço após sua criação. Durante o processo de atualização, AWS Proton faz uma contagem das instâncias de serviço existentes e das instâncias a serem adicionadas ou removidas. As instâncias com falha de exclusão estão incluídas nessa contagem e você deve contabilizá-las ao editar suas `spec`.

Use o console ou AWS CLI para adicionar ou remover instâncias de serviço

Console de gerenciamento da AWS

Edite seu serviço para adicionar ou remover instâncias de serviço usando o console.

No console do AWS Proton

1. No painel de navegação, escolha Serviços.
2. Selecione o serviço a ser editado.

3. Escolha Editar.
4. (Opcional) Na página Configurar serviço, edite o nome ou a descrição do serviço e escolha Avançar.
5. Na página Definir configurações personalizadas, escolha Excluir para excluir uma instância de serviço e escolha Adicionar nova instância para adicionar uma instância de serviço e preencher o formulário.
6. Escolha Próximo.
7. Revise a atualização e escolha Salvar alterações.
8. Um modal solicita que você verifique a exclusão das instâncias de serviço. Siga as instruções e escolha Sim, excluir.
9. Na página de detalhes do serviço, veja os detalhes do status do seu serviço.

AWS CLI

Adicione e exclua instâncias de serviço com uma versão editada**`spec`**, conforme mostrado nos AWS CLI exemplos de comandos e respostas a seguir.

Ao usar a CLI, sua você `spec` deve excluir as instâncias de serviço a serem excluídas e incluir as instâncias de serviço a serem adicionadas e as instâncias de serviço existentes que você não marcou para exclusão.

A lista a seguir mostra o exemplo de `spec` antes da edição e uma lista das instâncias de serviço implantadas pela especificação. Essa especificação foi usada no exemplo anterior para editar uma descrição de serviço.

Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
```

```
      my_sample_service_instance_optional_input: "def"
      my_sample_service_instance_required_input: "456"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

O exemplo de comando `list-service-instances` e resposta da CLI a seguir mostra as instâncias ativas antes de adicionar ou excluir uma instância de serviço.

Comando:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Resposta:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "example-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.160000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.160000+00:00",
            "name": "my-instance",
```

```
            "serviceName": "example-svc",
            "serviceTemplateArn": "arn:aws:proton:region-id:123456789012:service-
template/fargate-service",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

A lista a seguir mostra o exemplo editado `spec` usado para excluir e adicionar uma instância. A instância existente chamada `my-instance` é removida e uma nova instância chamada `yet-another-instance` é adicionada.

Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Você pode usar `"${Proton::CURRENT_VAL}"` para indicar quais valores de parâmetros devem ser preservados da `spec` original, se os valores existirem na `spec`. Use `get-service` para visualizar a `spec` original de um serviço, conforme descrito em Exibir dados do serviço.

A lista a seguir mostra como você pode usar `"${Proton::CURRENT_VAL}"` para garantir que sua `spec` não inclua alterações nos valores dos parâmetros para que as instâncias de serviços existentes permaneçam.

Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

A próxima listagem mostra o comando e a resposta da CLI para editar o serviço.

Comando:

```
$ aws proton update-service
    --name "MySimpleService" \
    --description "Edit by adding and deleting a service instance" \
    --spec "file://spec.yaml"
```

Resposta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by adding and deleting a service instance",
        "lastModifiedAt": "2021-03-12T22:55:48.169000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "UPDATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

O comando `list-service-instances` e a resposta a seguir confirmam que a instância existente chamada `my-instance` foi removida e uma nova instância chamada `yet-another-instance` foi adicionada.

Comando:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Resposta:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/yet-another-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:56:01.565000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:56:01.565000+00:00",
            "name": "yet-another-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
```

```
}
```

O que acontece quando você adiciona ou remove instâncias de serviço

Depois de enviar uma edição de serviço para excluir e adicionar instâncias de serviço AWS Proton , execute as seguintes ações.

- Configura o serviço em `UPDATE_IN_PROGRESS`
- Se o serviço tiver um pipeline, definirá seu status como `IN_PROGRESS` e bloqueará as ações do pipeline.
- Define todas as instâncias de serviço que devem ser excluídas para `DELETE_IN_PROGRESS`.
- Bloqueia as ações do serviço.
- Bloqueia ações em instâncias de serviço marcadas para exclusão.
- Cria novas instâncias de serviço.
- Exclui instâncias que você listou para exclusão.
- Tenta remover instâncias cuja exclusão falhou.
- Depois que as adições e exclusões forem concluídas, reprovisiona o pipeline de serviços (se houver), configura seu serviço para `ACTIVE` e habilita ações de serviço e pipeline.

AWS Proton tenta remediar os modos de falha da seguinte forma.

- Se uma ou mais instâncias de serviço falharem na criação, AWS Proton tente desprovisionar todas as instâncias de serviço recém-criadas e reverta `spec` para o estado anterior. Ele não exclui nenhuma instância de serviço e não modifica o pipeline de forma alguma.
- Se uma ou mais instâncias de serviço não forem excluídas, AWS Proton reprovisione o pipeline sem as instâncias excluídas. O `spec` é atualizado para incluir as instâncias adicionadas e excluir as instâncias que foram marcadas para exclusão.
- Se o pipeline falhar no provisionamento, não haverá uma tentativa de reversão e tanto o serviço quanto o pipeline refletirão um estado de falha na atualização.

Marcação e edições de serviço

Quando você adiciona instâncias de serviço como parte da edição do serviço, as tags AWS gerenciadas se propagam e são criadas automaticamente para as novas instâncias e recursos

provisionados. Se você criar novas tags, essas tags serão aplicadas somente às novas instâncias. As tags de serviço gerenciadas pelo cliente existentes também se propagam para as novas instâncias. Para obter mais informações, consulte AWS Proton recursos e marcação.

Excluir um serviço

Você pode excluir um AWS Proton serviço, com suas instâncias e pipeline, usando o AWS Proton console ou AWS CLI o.

Você não pode excluir um serviço que tenha qualquer instância de serviço com um componente anexado. Para excluir esse serviço, você deve primeiro atualizar todos os componentes anexados para desanexá-los de suas instâncias de serviço. Para obter mais informações sobre componentes, consulte Componentes.

Console de gerenciamento da AWS

Exclua um serviço usando o console conforme descrito nas etapas a seguir.

Na página de detalhes do serviço.

1. No console do AWS Proton, selecione Serviços.
2. Na lista de serviços, escolha o nome do serviço que deseja excluir.
3. Na página de detalhes de serviço, escolha Ações e, em seguida, Excluir.
4. Um modal pede que você confirme a ação de exclusão.
5. Siga as instruções e escolha Sim, excluir.

AWS CLI

Exclua um serviço conforme mostrado no exemplo de comando e resposta da CLI a seguir.

Comando:

```
$ aws proton delete-service \
    --name "simple-svc"
```

Resposta:

```
{
    "service": {
```

```
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2020-11-29T00:30:39.248000+00:00",
        "name": "simple-svc",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "DELETE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Exibir dados da instância de serviço

Aprenda a visualizar os dados detalhados da instância de AWS Proton serviço. É possível o console ou a AWS CLI.

Uma instância de serviço pertence a um serviço. Você só pode criar ou excluir uma instância dentro do contexto das ações de editar, criar e excluir do serviço. Para saber como adicionar e remover instâncias de um serviço, consulte Editar um serviço.

Console de gerenciamento da AWS

Liste e visualize os detalhes de instância do serviço usando o console do AWS Proton, conforme mostrado nas etapas a seguir.

1. Para ver uma lista das suas instâncias de serviço, escolha Instâncias de serviço no painel de navegação.
2. Para visualizar dados detalhados, escolha o nome de uma instância de serviço.

   Visualize os dados detalhados da sua instância de serviço.

AWS CLI

Liste e visualize os detalhes da instância de serviço conforme mostrado nos seguintes exemplos de comandos e respostas da CLI.

Comando:

```
$ aws proton list-service-instances
```

Resposta:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
service-instance/instance-one",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentArn": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "name": "instance-one",
            "serviceName": "simple-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Comando:

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Resposta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
        "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
```

```
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: hello world\n
 my_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_optional_input:
 Ola\n    my_sample_service_instance_required_input: Ciao\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Atualize uma instância de serviço

Aprenda a atualizar uma instância AWS Proton de serviço e cancelar a atualização.

Uma instância de serviço pertence a um serviço. Você só pode criar ou excluir uma instância dentro do contexto das ações de editar, criar e excluir do serviço. Para saber como adicionar e remover instâncias de um serviço, consulte Editar um serviço.

Há quatro modos para atualizar uma instância de serviço, conforme descrito na lista a seguir. Ao usar o AWS CLI, o `deployment-type` campo define o modo. Ao usar o console, esses modos são mapeados para as ações Editar, Atualizar para a versão secundária mais recente e Atualizar para a versão principal mais recente, que aparecem em Ações na página de detalhes da instância de serviço.

`NONE`

Nesse modo, uma implantação não ocorre. Somente as ramificações de metadata solicitadas do são atualizadas.

`CURRENT_VERSION`

Nesse modo, a instância de serviço é implantada e atualizada com a nova especificação fornecida por você. Somente ramificações solicitadas do são atualizadas. Não inclua parâmetros de versão secundária ou principal ao usar este `deployment-type`.

`MINOR_VERSION`

Nesse modo, a instância de serviço é implantada e atualizada com a versão secundária publicada e recomendada (mais recente) da versão principal atual em uso por padrão. Você também pode especificar uma versão secundária diferente da versão principal atual em uso.

`MAJOR_VERSION`

Nesse modo, a instância de serviço é implantada e atualizada com a versão principal e secundária publicada e recomendada (mais recente) do modelo atual por padrão. Você também pode especificar uma versão principal diferente que seja superior à versão principal em uso e uma versão secundária (opcional).

Você pode tentar cancelar a implantação de uma atualização de instância de serviço, se `deploymentStatus` for o caso`IN_PROGRESS`. AWS Proton tentativas de cancelar a implantação. O cancelamento bem-sucedido não é garantido.

Quando você cancela uma implantação de atualização, AWS Proton tenta cancelar a implantação conforme listado nas etapas a seguir.

- Define o estado de implantação como `CANCELLING`.
- Interrompe a implantação em andamento e exclui todos os novos recursos que foram criados pela implantação quando `IN_PROGRESS`.
- Define o estado de implantação como `CANCELLED`.
- Reverte o estado do recurso para o que era antes do início da implantação.

Para obter mais informações sobre o cancelamento da implantação de uma instância de serviço, consulte CancelServiceInstanceDeployment a Referência da AWS Proton API.

Use o console ou AWS CLI para fazer atualizações ou cancelar implantações de atualizações.

Console de gerenciamento da AWS

Atualize uma instância de serviço usando o console seguindo estas etapas.

1. No console do AWS Proton, escolha Instâncias de serviço no painel de navegação.
2. Na lista de instâncias de serviço, escolha o nome da instância de serviço que você deseja atualizar.

3. Escolha Ações e, em seguida, escolha uma das opções de atualização, Editar para atualizar as especificações ou Ações e, em seguida, Atualizar para a versão secundária mais recente ou Atualizar para a versão principal mais recente.
4. Preencha cada formulário e escolha Avançar até chegar à página Revisar.
5. Revise suas edições e escolha Atualizar.

AWS CLI

Atualize uma instância de serviço para uma nova versão secundária, conforme mostrado nos comandos e respostas de exemplo da CLI.

Quando você atualiza sua instância de serviço com uma `spec` modificada, você pode usar `"${Proton::CURRENT_VAL}"` para indicar quais valores de parâmetros devem ser preservados da `spec` original, se os valores existirem na `spec`. Use `get-service` para visualizar a `spec` original de uma instância de serviço, conforme descrito em Exibir dados do serviço.

O exemplo a seguir mostra como você pode usar a `"${Proton::CURRENT_VAL}"` em uma `spec`.

Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Comando: atualizar

```
$ aws proton update-service-instance \
    --name "instance-one" \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Resposta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentName": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Comando: obter e confirmar o status

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Resposta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
```

```
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Console de gerenciamento da AWS

Cancele a implantação de uma instância de serviço usando o console, conforme mostrado nas etapas a seguir.

1. No console do AWS Proton, escolha Instâncias de serviço no painel de navegação.
2. Na lista de instâncias de serviço, escolha o nome da instância de serviço com a atualização de implantação que você deseja cancelar.
3. Se o status de implantação da atualização estiver Em andamento, na página de detalhes da instância de serviço, escolha Ações e, em seguida, Cancelar implantação.
4. Um modal pede que você confirme o cancelamento. Escolha Cancelar implantação.
5. O status de implantação da atualização será definido como Cancelando e, em seguida, Cancelado para concluir o cancelamento.

AWS CLI

Cancele uma atualização de implantação da instância de serviço IN_PROGRESS para a nova versão secundária 2, conforme mostrado nos seguintes exemplos de comandos e respostas da CLI.

Uma condição de espera é incluída no modelo usado neste exemplo para que o cancelamento comece antes que a implantação da atualização seja bem-sucedida.

Comando: cancelar

```
$ aws proton cancel-service-instance-deployment \
    --service-instance-name "instance-one" \
    --service-name "simple-svc"
```

Resposta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: abc\n  my_sample_pipeline_required_input:
 '123'\ninstances:\n- name: my-instance\n  environment: MySimpleEnv
\n  spec:\n    my_sample_service_instance_optional_input: def\n
 my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
 environment: MySimpleEnv\n  spec:\n    my_sample_service_instance_required_input:
 '789'\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Comando: obter e confirmar o status

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Resposta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```


Atualizar um pipeline de serviço

Aprenda a atualizar um pipeline AWS Proton de serviços e cancelar a atualização.

Um pipeline de serviços pertence a um serviço. Você só pode criar ou excluir um pipeline dentro do contexto das ações de criação e exclusão do serviço.

Há quatro modos para atualizar um pipeline de serviço, conforme descrito na lista a seguir. Ao usar o AWS CLI, o `deployment-type` campo define o modo. Quando você usa o console, esses modos são mapeados para Editar pipeline e Atualizar para a versão recomendada.

`NONE`

Nesse modo, uma implantação não ocorre. Somente as ramificações de metadata solicitadas do são atualizadas.

`CURRENT_VERSION`

Nesse modo, o pipeline de serviço é implantado e atualizado com a nova especificação fornecida por você. Somente ramificações solicitadas do são atualizadas. Não inclua parâmetros de versão secundária ou principal ao usar este `deployment-type`.

`MINOR_VERSION`

Nesse modo, o pipeline de serviço é implantado e atualizado com a versão secundária publicada e recomendada (mais recente) da versão principal atual em uso por padrão. Você também pode especificar uma versão secundária diferente da versão principal atual em uso.

`MAJOR_VERSION`

Nesse modo, o pipeline de serviço é implantado e atualizado com a versão principal e secundária publicada e recomendada (mais recente) do modelo atual por padrão. Você também pode especificar uma versão principal diferente que seja superior à versão principal em uso e uma versão secundária (opcional).

Você pode tentar cancelar a implantação de uma atualização do pipeline de serviço, se `deploymentStatus` for o caso`IN_PROGRESS`. AWS Proton tentativas de cancelar a implantação. O cancelamento bem-sucedido não é garantido.

Quando você cancela uma implantação de atualização, AWS Proton tenta cancelar a implantação conforme listado nas etapas a seguir.

- Define o estado de implantação como `CANCELLING`.
- Interrompe a implantação em andamento e exclui todos os novos recursos que foram criados pela implantação quando `IN_PROGRESS`.
- Define o estado de implantação como `CANCELLED`.
- Reverte o estado do recurso para o que era antes do início da implantação.

Para obter mais informações sobre o cancelamento da implantação de um pipeline de serviço, consulte CancelServicePipelineDeployment a Referência da AWS Proton API.

Use o console ou AWS CLI para fazer atualizações ou cancelar implantações de atualizações.

Console de gerenciamento da AWS

Atualize um pipeline de serviço usando o console conforme descrito nas etapas a seguir.

1. No console do AWS Proton, selecione Serviços.
2. Na lista de serviços, escolha o nome do serviço para o qual deseja atualizar o pipeline.
3. Há duas guias na página de detalhes do serviço, Visão geral e Pipeline. Escolher Pipeline.
4. Se você quiser atualizar as especificações, escolha Editar pipeline, preencha cada formulário e escolha Avançar até concluir o formulário final e, em seguida, escolha Atualizar pipeline.

   Se você quiser atualizar para uma nova versão e houver um ícone de informações que indica que uma nova versão está disponível no Modelo de pipeline, escolha o nome da nova versão do modelo.

   a. Escolha Atualizar para a versão recomendada.
   b. Preencha cada formulário e escolha Avançar até preencher o formulário final e escolher Atualizar.

AWS CLI

Atualize um pipeline de serviço para uma nova versão secundária, conforme mostrado nos seguintes exemplos de comandos e respostas da CLI.

Quando você atualiza seu pipeline de serviço com uma `spec` modificada, você pode usar `"${Proton::CURRENT_VAL}"` para indicar quais valores de parâmetros devem ser preservados da `spec` original, se os valores existirem na `spec`. Use `get-service` para visualizar a `spec` original de um pipeline de serviço, conforme descrito em Exibir dados do serviço.

O exemplo a seguir mostra como você pode usar a `"${Proton::CURRENT_VAL}"` em uma `spec`.

Especificação:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"
```

```
instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Comando: atualizar

```
$ aws proton update-service-pipeline \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Resposta:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"my-instance\"\n    environment: \"MySimpleEnv
\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"MySimpleEnv\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Comando: obter e confirmar o status

```
$ aws proton get-service \
    --name "simple-svc"
```

Resposta:


```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
```

```
        "templateName": "svc-simple"
    }
}
```

Console de gerenciamento da AWS

Cancele a implantação de um pipeline de serviço usando o console, conforme mostrado nas etapas a seguir.

1. No console do AWS Proton, escolha Serviços no painel de navegação.
2. Na lista de serviços, escolha o nome do serviço que tem o pipeline com a atualização de implantação que você deseja cancelar.
3. Na página de detalhes do serviço selecione a guia Pipeline.
4. Se o status de implantação da atualização estiver Em andamento, na página de detalhes do pipeline de serviço, escolha Cancelar implantação.
5. Um modal pede que você confirme o cancelamento. Escolha Cancelar implantação.
6. O status de implantação da atualização será definido como Cancelando e, em seguida, Cancelado para concluir o cancelamento.

AWS CLI

Cancele uma atualização de implantação do pipeline de serviço IN_PROGRESS para a versão secundária 2, conforme mostrado nos seguintes exemplos de comandos e respostas da CLI.

Uma condição de espera é incluída no modelo usado neste exemplo para que o cancelamento comece antes que a implantação da atualização seja bem-sucedida.

Comando: cancelar

```
$ aws proton cancel-service-pipeline-deployment \
    --service-name "simple-svc"
```

Resposta:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",
```

```
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Comando: obter e confirmar o status

```
$ aws proton get-service \
    --name "simple-svc"
```

Resposta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "CANCELLED",
            "deploymentStatusMessage": "User initiated cancellation.",
            "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
```

```
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

AWS Proton componentes

Os componentes são um tipo de AWS Proton recurso. Eles adicionam flexibilidade aos modelos de serviço. Os componentes fornecem às equipes da plataforma um mecanismo para ampliar os principais padrões de infraestrutura e definir salvaguardas que capacitam os desenvolvedores a gerenciar aspectos de sua infraestrutura de aplicações.

Em, AWS Proton os administradores definem a infraestrutura padrão que é usada em equipes de desenvolvimento e aplicativos. No entanto, as equipes de desenvolvimento podem precisar incluir recursos adicionais para seus casos de uso específicos, como filas do Amazon Simple Queue Service (Amazon SQS) ou tabelas do Amazon DynamoDB. Esses recursos específicos da aplicação podem mudar com frequência, especialmente durante o desenvolvimento inicial da aplicação. Manter essas mudanças frequentes nos modelos criados pelo administrador pode ser difícil de gerenciar e escalar — os administradores precisariam manter muitos outros modelos sem o real valor agregado do administrador. A alternativa — permitir que os desenvolvedores de aplicativos criem modelos para seus aplicativos — também não é ideal, pois retira a capacidade dos administradores de padronizar os principais componentes da arquitetura, como tarefas. AWS Fargate É aqui que entram os componentes.

Com um componente, um desenvolvedor pode adicionar recursos suplementares à aplicação, além do que os administradores definiram nos modelos de ambiente e serviço. Em seguida, o desenvolvedor anexa o componente a uma instância de serviço. AWS Proton provisiona recursos de infraestrutura definidos pelo componente da mesma forma que provisiona recursos para ambientes e instâncias de serviço.

Um componente pode ler as entradas da instância de serviço e fornecer saídas para a instância de serviço, para uma experiência totalmente integrada. Por exemplo, se o componente adiciona um bucket do Amazon Simple Storage Service (Amazon S3) para uso por uma instância de serviço, o modelo de componente pode levar os nomes de instância do ambiente e do serviço em conta para nomear o bucket. Quando AWS Proton renderiza o modelo de serviço para provisionar uma instância de serviço, a instância de serviço pode se referir ao bucket e usá-lo.

Os componentes que AWS Proton atualmente oferecem suporte são componentes definidos diretamente. Você passa o arquivo Infrastructure as Code (IaC) que define a infraestrutura do componente diretamente para a AWS Proton API ou o console. Isso é diferente de um ambiente ou serviço, em que você define o IaC em um pacote de modelo e registra o pacote como um recurso de modelo e, em seguida, usa um recurso de modelo para criar o ambiente ou serviço.

> **Note**
>
> Componentes definidos diretamente permitem que os desenvolvedores definam uma infraestrutura extra e a provisionem. AWS Proton provisiona todos os componentes diretamente definidos em execução no mesmo ambiente usando a mesma função AWS Identity and Access Management (IAM).

Um administrador pode controlar o que os desenvolvedores podem fazer com os componentes de duas maneiras:

- Fontes de componentes compatíveis — Um administrador pode permitir a anexação de componentes às instâncias de serviço com base em uma propriedade das versões do modelo de AWS Proton serviço. Por padrão, os desenvolvedores não podem anexar componentes às instâncias de serviço.

  Para obter mais informações sobre essa propriedade, consulte o supportedComponentSourcesparâmetro da ação da CreateServiceTemplateVersionAPI na Referência da AWS Proton API.

  > **Note**
  >
  > Quando você usa a sincronização de modelos, AWS Proton cria versões de modelo de serviço implicitamente ao confirmar alterações em um pacote de modelos de serviço em um repositório. Nesse caso, em vez de especificar as fontes de componentes compatíveis durante a criação da versão do modelo de serviço, você especifica essa propriedade em um arquivo associado a cada versão principal do modelo de serviço. Para obter mais informações, consulte the section called “Como sincronizar modelos de serviço”.

- Funções de componentes — Um administrador pode atribuir uma função de componente a um ambiente. AWS Proton assume esse papel quando provisiona a infraestrutura definida por componente diretamente definido no ambiente. Portanto, o perfil do componente define a infraestrutura que os desenvolvedores podem adicionar usando componentes diretamente definidos no ambiente. Na ausência do perfil de componente, os desenvolvedores não podem criar componentes diretamente definidos no ambiente.

Para obter mais informações sobre a atribuição de uma função de componente, consulte o componentRoleArnparâmetro da ação da CreateEnvironmentAPI na Referência da AWS Proton API.

> **Note**
>
> Os perfis dos componentes não são usados em ambientes de Provisionamento autogerenciado.

Tópicos

Como os componentes se comparam a outros AWS Proton recursos?

De muitas maneiras, os componentes são semelhantes a outros AWS Proton recursos. Sua infraestrutura é definida em um arquivo de modelo IaC, criado no formato CloudFormation YAML ou Terraform HCL. AWS Proton pode provisionar a infraestrutura de componentes usando provisionamento AWS gerenciado ou autogerenciado.

No entanto, os componentes são diferentes de outros AWS Proton recursos de algumas maneiras:

- Estado desanexado: os componentes são projetados para serem anexados às instâncias de serviço e para ampliar sua infraestrutura, mas também podem estar em um estado desanexado, no qual não estão conectados a nenhuma instância de serviço. Para obter mais informações sobre estados de componentes, consulte the section called “Status de componentes”.

- Sem esquema: Os componentes não têm um esquema associado, como os pacotes de modelos. As entradas dos componentes são definidas por um serviço. Um componente pode consumir entradas quando está conectado a uma instância de serviço.
- Sem componentes gerenciados pelo cliente — AWS Proton sempre provisiona a infraestrutura de componentes para você. Não há uma versão dos componentes para trazer seus próprios recursos. Para obter mais informações sobre ambientes gerenciados pelo cliente, consulte the section called "Criar"
- Sem recurso de modelo: componentes diretamente definidos não têm um recurso de modelo associado semelhante aos modelos de ambiente e serviço. Você fornece um arquivo de modelo de IaC diretamente para o componente. Da mesma forma, você fornece diretamente um manifesto que define a linguagem do modelo e o mecanismo de renderização para provisionar a infraestrutura do componente. Você cria o arquivo de modelo e o manifesto de forma semelhante à criação de um pacote de modelos. No entanto, com componentes definidos diretamente, não há necessidade de armazenar arquivos IaC como pacotes em locais específicos, e você não cria um recurso de modelo a AWS Proton partir de arquivos IaC.
- Sem provisionamento CodeBuild baseado — você não pode provisionar componentes diretamente definidos usando seu próprio script de provisionamento personalizado, conhecido como provisionamento baseado. CodeBuild Para obter mais informações, consulte the section called "CodeBuild aprovisionamento".

Componentes no AWS Proton console

Use o AWS Proton console para criar, atualizar, visualizar e usar AWS Proton componentes.

As páginas do console a seguir estão relacionadas aos componentes. Incluímos links diretos para páginas de console de nível superior.

- Componentes — Visualize a lista de componentes em sua AWS conta. Você pode criar novos componentes e atualizar ou excluir componentes existentes. Escolha um nome de componente na lista para ver sua página de detalhes.

  Listas semelhantes também existem nas páginas de Detalhes do ambiente e Detalhes da instância de serviço. Essas listas mostram somente os componentes associados ao recurso que está sendo visualizado. Ao criar um componente a partir de uma dessas listas, AWS Proton pré-seleciona o ambiente associado na página Criar componente.

- Detalhes do componente: para visualizar a página de detalhes do componente, escolha um nome de componente na lista Componentes.

  Na página de detalhes, visualize os detalhes e o status do componente e atualize ou exclua o componente. Visualize e gerencie listas de saídas (por exemplo, recursos provisionados ARNs), CloudFormation pilhas provisionadas e tags atribuídas.
- Criar componente: criar um componente. Insira o nome e a descrição do componente, escolha os recursos associados, especifique o arquivo de IaC de origem do componente e atribua tags.
- Atualizar componente: para atualizar um componente, selecione o componente na lista Componentes e, no menu Ações, escolha Atualizar componente. Como alternativa, nas páginas de Detalhes do componente, escolha Atualizar.

  Você pode atualizar a maioria dos detalhes do componente. Você não pode atualizar o nome do componente. E você pode escolher se deseja ou não reimplantar o componente após uma atualização bem-sucedida.
- Configurar ambiente: Ao criar ou atualizar um ambiente, você pode especificar um perfil de componente. Esse perfil controla a capacidade de executar componentes diretamente definidos no ambiente e fornece permissões para provisioná-los.
- Criar nova versão do modelo de serviço: Ao criar uma versão do modelo de serviço, você pode especificar as Fontes de componentes compatíveis para a versão do modelo. Isso controla a capacidade de anexar componentes às instâncias de serviços com base nessa versão do modelo.

Componentes na AWS Proton API e AWS CLI

Use a AWS Proton API ou a AWS CLI para criar, atualizar, visualizar e usar AWS Proton componentes.

As ações de API a seguir gerenciam diretamente os recursos dos AWS Proton componentes.

- CreateComponent— Crie um AWS Proton componente.
- DeleteComponent— Exclua um AWS Proton componente.
- GetComponent— Obtenha dados detalhados de um componente.
- ListComponentOutputs— Obtenha uma lista de saídas de componentes de infraestrutura como código (IaC).
- ListComponentProvisionedResources— Liste os recursos provisionados para um componente com detalhes.

- ListComponents— Listar componentes com dados resumidos. Você pode filtrar a lista de resultados por ambiente, serviço ou uma única instância de serviço.

As seguintes ações de API de outros AWS Proton recursos têm algumas funcionalidades relacionadas aos componentes.

- CreateEnvironment, UpdateEnvironment— Use `componentRoleArn` para especificar o Amazon Resource Name (ARN) da função de serviço do IAM que é AWS Proton usada ao provisionar componentes diretamente definidos nesse ambiente. Ela determina o escopo da infraestrutura que um componente diretamente definido pode provisionar.
- CreateServiceTemplateVersion— Use `supportedComponentSources` para especificar fontes de componentes compatíveis. Componentes com fontes compatíveis podem ser anexados às instâncias de serviço com base nessa versão do modelo de serviço.

Perguntas frequentes sobre o componente

Qual é o ciclo de vida de um componente?

Os componentes podem estar em um estado conectado ou desconectado. Eles são projetados para serem conectados a uma instância de serviço e aprimorar sua infraestrutura na maioria das vezes. Os componentes desanexados estão em um estado de transição que permite excluir um componente ou anexá-lo a outra instância de serviço de forma controlada e segura. Para obter mais informações, consulte the section called “Status de componentes”.

Por que não consigo excluir meus componentes anexados?

Solução: para excluir um componente anexado, atualize o componente para desanexá-lo da instância de serviço, valide a estabilidade da instância de serviço e, em seguida, exclua o componente.

Por que isso é necessário? Os componentes conectados fornecem a infraestrutura extra de que sua aplicação precisa para executar suas funções de runtime. A instância de serviço pode estar usando saídas de componentes para detectar e usar recursos dessa infraestrutura. Excluir o componente e, assim, remover seus recursos de infraestrutura, pode causar interrupções na instância de serviço conectada.

Como medida de segurança adicional, AWS Proton exige que você atualize o componente e o desconecte de sua instância de serviço antes de excluí-lo. Em seguida, você pode validar sua

instância de serviço para garantir que ela continue sendo implantada e funcionando adequadamente. Se você detectar um problema, poderá reconectar rapidamente o componente à instância de serviço e, em seguida, trabalhar para corrigir o problema. Quando tiver certeza de que sua instância de serviço está livre de qualquer dependência do componente, você pode excluir o componente com segurança.

Por que não consigo alterar diretamente a instância de serviço conectada a um componente?

Solução: para mudar a anexação, atualize o componente para desanexá-lo da instância de serviço, valide a estabilidade do componente e da instância de serviço e então anexe o componente à nova instância de serviço.

Por que isso é necessário? Um componente foi projetado para ser anexado a uma instância de serviço. Seu componente pode usar entradas de instância de serviço para nomeação e configuração de recursos de infraestrutura. Alterar a instância de serviço conectada pode causar interrupções no componente (além de uma possível interrupção na instância de serviço, conforme descrito nas perguntas frequentes anteriores, Por que não consigo excluir meus componentes conectados?). Por exemplo, isso pode causar a renomeação e possivelmente até a substituição dos recursos definidos no modelo de IaC do componente.

Como medida de segurança adicional, AWS Proton exige que você atualize o componente e o desconecte de sua instância de serviço antes de poder anexá-lo a outra instância de serviço. Em seguida, você pode validar a estabilidade do componente e da instância de serviço antes de anexar o componente à nova instância de serviço.

Status de componentes

AWS Proton os componentes podem estar em dois estados fundamentalmente diferentes:

- Anexado: O componente é anexado a uma instância de serviço. Ele define a infraestrutura que suporta a funcionalidade de runtime da instância de serviço. O componente estende a infraestrutura definida nos modelos de ambiente e serviço com a infraestrutura definida pelo desenvolvedor.

  Um componente típico está no estado conectado durante a maior parte da parte útil de seu ciclo de vida.
- Desanexado — O componente está associado a um AWS Proton ambiente e não está vinculado a nenhuma instância de serviço no ambiente.

Esse é um estado de transição para estender a vida útil de um componente além de uma única instância de serviço.

A tabela a seguir fornece uma comparação de alto nível dos diferentes estados dos componentes.

	Attached	Desanexados
Objetivo principal do estado	Estender a infraestrutura de uma instância de serviço.	Para manter a infraestrutura do componente entre os anexos da instância de serviço.
Associado a	Uma instância de serviço e um ambiente	Um ambiente do
Principais propriedades específicas	• Nome do serviço • Nome da instância de serviço • Especificação	• Nome do ambiente
Pode ser excluído	✗ Não	✓ Sim
Pode ser atualizado para outra instância de serviço	✗ Não	✓ Sim
Pode ler entradas	✓ Sim	✗ Não

O objetivo principal de um componente é ser anexado a uma instância de serviço e ampliar sua infraestrutura com recursos adicionais. Um componente anexado pode ler as entradas da instância de serviço de acordo com a especificação. Você não pode excluir diretamente o componente nem anexá-lo a uma instância de serviço diferente. Você também não pode excluir sua instância de serviço nem o serviço e o ambiente relacionados. Para fazer qualquer uma dessas coisas, primeiro atualize o componente para desconectá-lo de sua instância de serviço.

Para manter a infraestrutura do componente além da vida útil de uma única instância de serviço, você atualiza o componente e o separa de sua instância de serviço removendo o serviço e os nomes da instância de serviço. Esse estado separado é um estado de transição. O componente não tem entradas. Sua infraestrutura permanece provisionada e você pode atualizá-la. Você pode excluir recursos aos quais o componente estava associado quando foi anexado (instância de serviço, serviço). Você pode excluir o componente ou atualizá-lo para ser anexado novamente a uma instância de serviço.

Infraestrutura de componentes como arquivos de código

Os arquivos de infraestrutura de componentes como código (IaC) são semelhantes aos de outros AWS Proton recursos. Conheça aqui alguns detalhes específicos dos componentes. Para obter informações completas sobre a criação de arquivos IaC para AWS Proton, consulte. Criação e pacotes de modelos

Usando parâmetros com componentes

O namespace do AWS Proton parâmetro inclui alguns parâmetros que um arquivo IaC de serviço pode referenciar para obter o nome e as saídas de um componente associado. O namespace também inclui parâmetros que um arquivo IaC do componente pode referenciar para obter entradas, saídas e valores de recursos do ambiente, serviço e instância de serviço aos quais o componente está associado.

Um componente não tem entradas próprias — ele obtém suas entradas da instância de serviço à qual está conectado. Um componente também pode ler as saídas do ambiente.

Para obter mais informações sobre o uso de parâmetros em arquivos IaC de componentes e serviços associados, consulte the section called “Parâmetros do componente CloudFormation IaC”. Para obter informações gerais sobre AWS Proton parâmetros e uma referência completa do namespace do parâmetro, consulte. the section called “Parâmetros”

Criação de arquivos IaC robustos

Como administrador, ao criar uma versão do modelo de serviço, você pode decidir se deseja permitir que as instâncias de serviço criadas a partir da versão do modelo tenham componentes anexados. Veja o supportedComponentSourcesparâmetro da ação da CreateServiceTemplateVersionAPI na Referência da AWS Proton API. No entanto, para qualquer instância de serviço futura, a pessoa que cria a instância, decide se deseja ou não anexar um componente a ela e (no caso de componentes

definidos diretamente) é a autora do componente IaC normalmente é uma pessoa diferente — um desenvolvedor usando seu modelo de serviço. Portanto, você não pode garantir que um componente seja anexado a uma instância de serviço. Você também não pode garantir a existência de nomes de saída de componentes específicos ou a validade e segurança dos valores dessas saídas.

AWS Proton e a sintaxe do Jinja ajuda você a contornar esses problemas e criar modelos de serviço robustos que são renderizados sem falhas das seguintes maneiras:

- AWS Proton filtros de parâmetros — Ao se referir às propriedades de saída do componente, você pode usar filtros de parâmetros — modificadores que validam, filtram e formatam valores de parâmetros. Para ter mais informações e exemplos, consulte the section called “CloudFormation filtros de parâmetros”.
- Padrão de propriedade única: Ao se referir a um único recurso ou propriedade de saída de um componente, você pode garantir que a renderização do seu modelo de serviço não falhe usando o filtro do `default`, com ou sem um valor padrão. Se o componente ou um parâmetro de saída específico ao qual você está se referindo não existir, o valor padrão (ou uma string vazia, se você não tiver especificado um valor padrão) será renderizado e a renderização será bem-sucedida. Para obter mais informações, consulte the section called “Fornecer valores padrão”.

  Exemplos:

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

  > **Note**
  >
  > Não confunda a parte `.default` do namespace, que designa componentes diretamente definidos, com o filtro `default`, que fornece um valor padrão quando a propriedade referenciada não existe.

- Referência de objeto inteiro: Quando você se refere ao componente inteiro ou à coleção das saídas de um componente, o AWS Proton retorna um objeto vazio, `{}`, e, portanto, garante que a renderização do seu modelo de serviço não falhe. Você não precisa utilizar qualquer filtro Certifique-se de fazer a referência em um contexto que possa usar um objeto vazio ou use uma condição de `{{ if .. }}` para testar um objeto vazio.

  Exemplos:

  - `{{ service_instance.components.default }}`

- `{{ service_instance.components.default.outputs }}`

CloudFormation Exemplo de componente

Aqui está um exemplo completo de um componente definido AWS Proton diretamente e como você pode usá-lo em um AWS Proton serviço. O componente diretamente definido provisiona um bucket do Amazon Simple Storage Service (Amazon S3) e uma política de acesso relacionada. A instância de serviço pode se referir a esse bucket e usá-lo. O nome do bucket é baseado nos nomes do ambiente, serviço, instância de serviço e componente, o que significa que o bucket é acoplado a uma instância específica do modelo de componente que estende uma instância de serviço específica. Os desenvolvedores podem criar vários componentes com base nesse modelo de componente para provisionar buckets do Amazon S3 para diferentes instâncias de serviço e necessidades funcionais.

O exemplo abrange a criação das várias CloudFormation infraestruturas necessárias como arquivos de código (IaC) e a criação de uma função necessária AWS Identity and Access Management (IAM). O exemplo agrupa etapas de acordo com os perfis das pessoas proprietárias.

Etapas do administrador

Para permitir que os desenvolvedores usem componentes com um serviço

1. Crie uma função AWS Identity and Access Management (IAM) que defina os recursos que os componentes diretamente definidos em execução em seu ambiente podem provisionar. AWS Proton assume essa função posteriormente para provisionar componentes diretamente definidos no ambiente.

   Para este exemplo, use a seguinte política:

   Example perfil de componente diretamente definido

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "cloudformation:CancelUpdateStack",
   ```

```
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

2. Forneça o perfil que você criou na etapa anterior ao criar ou atualizar o ambiente. No AWS Proton console, especifique uma função de componente na página Configurar ambiente. Se você estiver usando a AWS Proton API ou AWS CLI, especifique as ações `componentRoleArn` da CreateEnvironmentou UpdateEnvironmentda API.

3. Crie um modelo de serviço que se refira a um componente diretamente definido anexado à instância de serviço.

   O exemplo mostra como escrever um modelo de serviço robusto que não falha se um componente não estiver anexado à instância do serviço.

   Example arquivo CloudFormation IaC de serviço usando um componente

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

4. Crie uma nova versão secundária do modelo de serviço que declare os componentes diretamente definidos como compatíveis.

- Pacote de modelos no Amazon S3 — No AWS Proton console, ao criar uma versão de modelo de serviço, para Fontes de componentes compatíveis, escolha Definido diretamente. Se você estiver usando a AWS Proton API ou AWS CLI, especifique `DIRECTLY_DEFINED` no `supportedComponentSources` parâmetro das ações da UpdateServiceTemplateVersionAPI CreateServiceTemplateVersionou.
- Sincronização de modelos: Confirme uma alteração no repositório do pacote de modelos de serviço, onde você especifica `DIRECTLY_DEFINED` como um item `supported_component_sources:` no arquivo `.template-registration.yaml` no diretório da versão principal. Para obter mais informações sobre esse arquivo, consulte the section called “Como sincronizar modelos de serviço”.

5. Concede permissão para criar uma versão secundária do modelo de serviço. Para obter mais informações, consulte the section called “Publicar”.
6. Certifique-se de permitir no `proton:CreateComponent` o perfil do IAM dos desenvolvedores que usam esse modelo de serviço.

Etapas do desenvolvedor

Para usar um componente definido diretamente com uma instância de serviço

1. Crie um serviço que use a versão do modelo de serviço que o administrador criou com suporte a componentes. Como alternativa, atualize uma de suas instâncias de serviço existentes para usar a versão mais recente do modelo.
2. Escreva um arquivo de modelo de componente IaC que provisione um bucket do Amazon S3 e uma política de acesso relacionada e exponha esses recursos como saídas.

   Example arquivo CloudFormation IaC do componente

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
```

```
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

3. Se você estiver usando a AWS Proton API ou AWS CLI, escreva um arquivo de manifesto para o componente.

   Example manifesto de componente definido diretamente

   ```
   infrastructure:
     templates:
       - file: "cloudformation.yaml"
         rendering_engine: jinja
         template_language: cloudformation
   ```

4. Crie um componente definido diretamente. AWS Proton assume a função do componente que o administrador definiu para provisionar o componente.

   No AWS Proton console, na página Componentes, escolha Criar componente. Em Configurações do componente, insira o Nome do componente e opcionalmente uma Descrição do componente. Em Anexação de componente, escolha Anexar o componente a uma instância de serviço. Selecione seu ambiente, serviço e instância de serviço. Em Origem do componente, escolha e CloudFormation, em seguida, escolha o arquivo IaC do componente.

   > **Note**
   >
   > Você não precisa fornecer um manifesto — o console cria um para você.

Se você estiver usando a AWS Proton API ou AWS CLI, use a ação CreateComponentda API. Defina um componente `name` e opcionalmente `description`. Definir `environmentName`, `serviceName`, e `serviceInstanceName`. Defina `templateSource` e `manifest` para os caminhos dos arquivos que você criou.

> **Note**
>
> Especificar um nome de ambiente é opcional quando você especifica nomes de serviço e instância de serviço. A combinação desses dois é exclusiva em sua AWS conta e AWS Proton pode determinar o ambiente a partir da instância de serviço.

5. Atualize sua instância de serviço para reimplantá-la. AWS Proton usa saídas do seu componente no modelo de instância de serviço renderizado para permitir que seu aplicativo use o bucket do Amazon S3 que o componente provisionou.

Usando repositórios git com AWS Proton

AWS Proton usa repositórios git para diversas finalidades. A lista a seguir categoriza os tipos de repositório associados AWS Proton aos recursos. Para AWS Proton recursos que se conectam repetidamente ao seu repositório para enviar conteúdo para ele ou extrair conteúdo dele, você precisa registrar um link de repositório AWS Proton em sua AWS conta. Um link de repositório é um conjunto de propriedades que AWS Proton podem ser usadas quando se conecta a um repositório. AWS Proton atualmente suporta GitHub, GitHub Enterprise BitBucket e.

Repositórios para desenvolvedores

Repositório de código: Um repositório que os desenvolvedores usam para armazenar o código da aplicação. Usado para implantação de código. AWS Proton não interage diretamente com esse repositório. Quando um desenvolvedor provisiona um serviço que inclui um pipeline, ele fornece o nome do repositório e a ramificação para ler o código da aplicação. O AWS Proton passa essas informações para o pipeline que ele provisiona.

Para obter mais informações, consulte the section called “Criar”.

Repositórios do administrador

Repositório de modelos — Um repositório em que os administradores AWS Proton armazenam pacotes de modelos. Usado para sincronização de modelos. Quando um administrador cria um modelo AWS Proton, ele pode apontar para um repositório de modelos e AWS Proton manter o novo modelo sincronizado com ele. Quando o administrador atualiza o pacote de modelos no repositório, cria AWS Proton automaticamente uma nova versão do modelo. Vincule um repositório de modelos AWS Proton antes de poder usá-lo para sincronização.

Para obter mais informações, consulte the section called “Configurações de sincronização de modelo”.

> **Note**
>
> Não é necessário um repositório de modelos se você continuar a carregar seus modelos para o Amazon Simple Storage Service (Amazon S3) e chamar o gerenciamento de modelos APIs para criar novos AWS Proton modelos ou versões de modelos.

Repositórios de provisionamento autogerenciados

Repositório de infraestrutura: Um repositório que hospeda modelos de infraestrutura renderizados. Usado para provisionamento autogerenciado da infraestrutura de recursos. Quando um administrador cria um ambiente para provisionamento autogerenciado, ele fornece um repositório. AWS Proton envia pull requests (PRs) para esse repositório para criar a infraestrutura para o ambiente e para qualquer instância de serviço implantada no ambiente. Vincule um repositório de infraestrutura ao AWS Proton antes de usá-lo para provisionamento de infraestrutura autogerenciada.

Repositório de pipelines: Um repositório usado para criar pipelines. Usado para provisionamento autogerenciado de pipelines. O uso de um repositório adicional para provisionar pipelines AWS Proton permite armazenar configurações de pipeline independentemente de qualquer ambiente ou serviço individual. Você só precisa fornecer um único repositório de pipeline para todos os seus serviços de provisionamento autogerenciados. Vincule um repositório de pipeline AWS Proton antes de usá-lo para provisionamento de pipeline autogerenciado.

Para obter mais informações, consulte the section called “Provisionamento gerenciado pela AWS”.

Tópicos

Crie um link para seu repositório

Você pode criar um link para seu repositório usando o console ou a CLI. Quando você cria um link de repositório, AWS Proton cria uma função vinculada ao serviço para você.

Console de gerenciamento da AWS

Crie um link para seu repositório conforme mostrado nas etapas do console a seguir.

1. No console do AWS Proton, escolha Repositórios.
2. Escolha Criar repositório.
3. Na página Vincular novo repositório, na seção Detalhes do repositório:

a. Escolha seu provedor de repositório.

b. Escolha uma de suas conexões existentes. Se você não tiver uma, escolha Adicionar uma nova CodeStar conexão para criar uma conexão e, em seguida, volte ao AWS Proton console, atualize a lista de conexões e escolha sua nova conexão.

c. Escolha entre seus repositórios de código-fonte conectados.

4. [opcional] Na seção Tags, escolha Adicionar nova tag uma ou mais vezes e insira os pares de chave e valor.
5. Escolha Criar repositório.
6. Visualize os dados de detalhes do seu repositório vinculado.

AWS CLI

Crie e registre um link para seu repositório.

Execute o seguinte comando:

```
$ aws proton create-repository \
    --name myrepos/environments \
    --connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
    --provider "GITHUB" \
    --encryption-key "arn:aws:kms:region-id:123456789012:key/bPxRfiCYEXAMPLEKEY" \
    --tags key=mytag1,value=value1 key=mytag2,value=value2
```

Os dois últimos parâmetros, --encryption-key e --tags, são opcionais.

Resposta:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
        "connectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/2ad03b28-a7c4-EXAMPLE11111",
        "encryptionKey": "arn:aws:kms:region-id:123456789012:key/
bPxRfiCYEXAMPLEKEY",
        "name": "myrepos/environments",
        "provider": "GITHUB"
    }
```

```
}
```

Depois de criar um link de repositório, você pode ver uma lista de AWS tags gerenciadas pelo cliente, conforme mostrado no exemplo de comando a seguir. AWS Proton gera automaticamente tags AWS gerenciadas para você. Você também pode modificar e criar etiquetas gerenciadas pelo cliente usando o AWS CLI. Para obter mais informações, consulte AWS Proton recursos e marcação.

Comando:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments"
```

Exibir dados do repositório vinculado

Você pode listar e visualizar detalhes do repositório vinculado usando o console ou o AWS CLI. Para links de repositório usados para sincronizar repositórios git com AWS Proton, você pode recuperar a definição e o status da sincronização do repositório usando o. AWS CLI

Console de gerenciamento da AWS

Listar e visualizar detalhes do repositório vinculado usando o console do AWS Proton.

1. Para listar seus repositórios vinculados, escolha Repositórios no painel de navegação.
2. Para visualizar dados detalhados, escolha o nome de um repositório.

AWS CLI

Listar seus repositórios vinculados.

Execute o seguinte comando:

```
$ aws proton list-repositories
```

Resposta:

```
{
```

```
    "repositories": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
            "name": "myrepos/templates",
            "provider": "GITHUB"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
            "name": "myrepos/environments",
            "provider": "GITHUB"
        }
    ]
}
```

Visualizar os detalhes de um repositório vinculado.

Execute o seguinte comando:

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

Resposta:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Listar seus repositórios sincronizados.

O exemplo a seguir lista os repositórios que você configurou para sincronização de modelos.

Execute o seguinte comando:

```
$ aws proton list-repository-sync-definitions \
```

```
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Visualizar o status de sincronização do repositório.

O exemplo a seguir recupera o status de sincronização de um repositório de sincronização de modelos.

Execute o seguinte comando:

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Resposta:

```
{
    "latestSync": {
        "events": [
            {
                "event": "Clone started",
                "time": "2021-11-21T00:26:35.883000+00:00",
                "type": "CLONE_STARTED"
            },
            {
                "event": "Updated configuration",
                "time": "2021-11-21T00:26:41.894000+00:00",
                "type": "CONFIG_UPDATED"
            },
            {
                "event": "Starting syncs for commit 62c03ff86eEXAMPLE1111111",
                "externalId": "62c03ff86eEXAMPLE1111111",
                "time": "2021-11-21T00:26:44.861000+00:00",
                "type": "STARTING_SYNC"
            }
        ],
        "startedAt": "2021-11-21T00:26:29.728000+00:00",
        "status": "SUCCEEDED"
    }
```

```
}
```

Excluir um link de repositório

É possível excluir um link de repositório usando o console do ou a AWS CLI.

> **Note**
>
> A exclusão de um link do repositório remove somente o link registrado que AWS Proton tem em sua AWS conta. Ela não exclui informações do seu repositório.

Console de gerenciamento da AWS

Excluir um link do repositório usando o console.

Na página de detalhes do repositório.

1. No console do AWS Proton, escolha Repositórios.
2. Na lista de instantâneos, escolha o botão de rádio à esquerda do instantâneo que você deseja excluir.
3. Escolha Excluir.
4. Um modal pede que você confirme a ação de exclusão.
5. Siga as instruções e escolha Sim, excluir.

AWS CLI

Excluir um link de repositório.

Execute o seguinte comando:

```
$ aws proton delete-repository \
    --name myrepos/templates \
    --provider"GITHUB"
```

Resposta:

```
{
```

```
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Monitoramento AWS Proton

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS Proton suas AWS soluções. A seção a seguir descreve as ferramentas de monitoramento que você pode usar com AWS Proton.

Automatize AWS Proton com EventBridge

Você pode monitorar AWS Proton eventos na Amazon EventBridge. EventBridge fornece um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos software-as-a-service (SaaS) e. Serviços da AWS Você pode configurar eventos para responder às mudanças no estado do AWS recurso. EventBridge encaminha esses dados para serviços de destino, como o AWS Lambda Amazon Simple Notification Service. Esses eventos são os mesmos que aparecem nos CloudWatch Eventos da Amazon. CloudWatch O Events fornece um fluxo quase em tempo real de eventos do sistema que descrevem mudanças nos AWS recursos. Para obter mais informações, consulte O que é a Amazon EventBridge? no Guia do EventBridge usuário da Amazon.

Use EventBridge para ser notificado sobre mudanças de estado nos fluxos de trabalho de AWS Proton provisionamento.

Tipos de eventos

Os eventos são compostos por regras que incluem um padrão de eventos e metas. Você configura uma regra escolhendo o padrão do evento e os objetos de destino:

Padrão de evento

> Cada regra é expressa como um padrão de evento com a origem e o tipo de eventos a serem monitorados e os alvos do evento. Para monitorar eventos, você cria uma regra com o serviço que você está monitorando como fonte do evento. Por exemplo, você pode criar uma regra com um padrão de evento que usa o AWS Proton como fonte de eventos para acionar uma regra quando há alterações em um estado de implantação.

Destinos

> A regra recebe um serviço selecionado como o destino do evento. Você pode configurar um serviço de destino para enviar notificações, capturar informações de status, tomar medidas corretivas, iniciar eventos ou realizar outras ações.

Os objetos de evento contêm campos padrão de ID, conta Região da AWS, tipo de detalhe, fonte, versão, recurso e hora (opcional). O campo de detalhes é um objeto aninhado contendo campos personalizados para o evento.

AWS Proton os eventos são emitidos com base no melhor esforço. O melhor esforço de entrega significa que o serviço tenta enviar todos os eventos para EventBridge, mas em alguns casos raros, um evento pode não ser entregue.

Para cada AWS Proton recurso que pode emitir eventos, a tabela a seguir lista o valor do tipo de detalhe, os campos de detalhes e (quando disponível) uma referência a uma lista de valores para os campos de detalhes `status` e`previousStatus`. Quando um recurso é excluído, o valor do campo de detalhes do `status` é `DELETED`.

Recurso	Valor do tipo de detalhe	Campos de detalhes

`EnvironmentTemplate`	AWS Proton Alteração do status do modelo de ambiente	`name`
		
		`status`
		
		`previousStatus`

`EnvironmentTemplateVersion`	AWS Proton Alteração do status da versão do modelo de ambiente	`name`
		
		`majorVersion`
		
		`minorVersion`
		
		`status`
		
		`previousStatus`
		
		valores de status

Recurso	Valor do tipo de detalhe	Campos de detalhes

`ServiceTemplate`	AWS Proton Alteração do status do modelo de serviço	`name`
		`status`
		`previousStatus`

`ServiceTemplateVersion`	AWS Proton Alteração do status da versão do modelo de serviço	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		valores de status

`Environment`	AWS Proton Alteração do status do ambiente	`name`
		`status`
		`previousStatus`

Recurso	Valor do tipo de detalhe	Campos de detalhes

`Service`	AWS Proton Alteração do status do serviço	`name`
		`status`
		`previousStatus`
		valores de status

`ServiceInstance`	AWS Proton Alteração do status da instância de serviço	`name`
		`serviceName`
		`status`
		`previousStatus`

`ServicePipeline`	AWS Proton Alteração do status do pipeline de serviços	`serviceName`
		`status`
		`previousStatus`

`EnvironmentAccountConnection`	AWS Proton Alteração do status de conexão da conta de ambiente	`id`
		`status`
		`previousStatus`
		valores de status

Recurso	Valor do tipo de detalhe	Campos de detalhes

`Component`	AWS Proton Alteração do status do componente	`name`
		
		`status`
		
		`previousStatus`

AWS Proton exemplos de eventos

Os exemplos a seguir mostram as formas de AWS Proton enviar eventos para EventBridge o.

Modelo de serviço

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name"],
    "detail": {
        "name": "sample-service-template-name",
        "status": "PUBLISHED",
        "previousStatus": "DRAFT"
    }
}
```

Versão do modelo de serviço

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Version Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name:1.0"],
    "detail": {
        "name": "sample-service-template-name",
        "majorVersion": "1",
        "minorVersion": "0",
```

```
        "status": "REGISTRATION_FAILED",
        "previousStatus": "REGISTRATION_IN_PROGRESS"
    }
}
```

Ambiente

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Environment Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:environment/sample-
environment"],
    "detail": {
        "name": "sample-environment",
        "status": "DELETE_FAILED",
        "previousStatus": "DELETE_IN_PROGRESS"
    }
}
```

EventBridgeTutorial: Envie alertas do Amazon Simple Notification AWS Proton Service para alterações no status do serviço

Neste tutorial, você usa uma regra de evento AWS Proton pré-configurada que captura as alterações de status do seu AWS Proton serviço. EventBridgeenvia as alterações de status para um tópico do Amazon SNS. Você assina o tópico e o Amazon SNS envia e-mails de alteração de status para o seu AWS Proton serviço.

Pré-requisitos

Você tem um AWS Proton serviço existente com um `Active` status. Como parte deste tutorial, você pode adicionar instâncias de serviço a esse serviço e, em seguida, excluí-las.

Se você precisar criar um AWS Proton serviço, consulteIntrodução. Para obter mais informações, consulte AWS Proton cotas e the section called “Edite”.

Etapa 1: criar e se inscrever em um tópico do Amazon SNS

Crie um tópico do Amazon SNS para funcionar como um destino de evento para a regra de evento que você criou na Etapa 2.

Crie um tópico do Amazon SNS

1. Faça o login e abra o console do Amazon SNS.
2. No painel de navegação, selecione Tópicos, Criar tópico.
3. Em Criar página de tópico:
   a. Escolha Tipo Padrão.
   b. Em Nome, insira **`tutorial-service-status-change`** e selecione Criar tópico.
4. Na página de tutorial-service-status-changedetalhes, escolha Criar assinatura.
5. Na página Criar assinatura:
   a. Em Protocolo, escolha Email.
   b. Em Endpoint, insira um endereço de e-mail ao qual tenha acesso e escolha Criar assinatura.
6. Verifique sua conta de e-mail e espere para receber uma mensagem de e-mail de confirmação de assinatura. Quando você recebê-la, abra-a escolha Confirmar assinatura.

Etapa 2: registrar uma regra de evento

Registre uma regra de evento que capture as alterações de status do seu AWS Proton serviço. Para obter mais informações, consulte Pré-requisitos.

Crie uma regra de evento.

1. Abra o EventBridge console da Amazon.
2. No painel de navegação, escolha Events (Eventos), Rules (Regras).
3. Na página Regras, na seção Regras, escolha Criar regra.
4. Na página Criar regra:
   a. Na seção Nome e descrição, em Nome, insira **`tutorial-rule`**.
   b. Na seção Definir padrão, escolha Padrão de evento.
      i. Em Event matching pattern (Padrão de correspondência de eventos), escolha Pre-defined by service (Predefinido por serviço).
      ii. Em Provedor de serviços, escolha AWS.
      iii. Em Service Name (Nome do serviço), escolha AWS Proton.

iv. Em Tipo de evento, selecione Mudança de status do AWS Proton .

O padrão de evento aparece em um editor de texto.

v. Abra o console do AWS Proton.

vi. No painel de navegação, escolha Serviços.

vii. Na página Serviços, escolha o nome do seu AWS Proton serviço.

viii. Na página Detalhes do serviço, copie o nome do recurso da Amazon (ARN) do serviço.

ix. Navegue de volta ao EventBridge console e à regra do tutorial e escolha Editar no editor de texto.

x. No editor de texto, para `"resources":`, insira o ARN do serviço que você copiou na etapa viii.

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

xi. Salve o padrão do evento.

c. Na seção Selecionar alvos:

i. Em Target (Destino), selecione SNS topic (Tópico do SNS).

ii. Em Tópico, escolha tutorial-service-status-change.

d. Escolha Criar.

Etapa 3: Testar sua regra de evento

Verifique se sua regra de evento está funcionando adicionando uma instância ao seu AWS Proton serviço.

1. Vá para o Console do AWS Proton.
2. No painel de navegação, escolha Serviços.
3. Na página de Serviços, selecione o nome do seu serviço.
4. Na página Detalhes do serviço, escolha Editar.
5. Na página Configurar serviço, escolha Avançar.

6. Na página Definir configurações personalizadas, na seção Instâncias de serviço, escolha Adicionar nova instância.
7. Preencha o formulário para sua nova instância:
   a. Digite um Nome para sua nova instância.
   b. Selecione os mesmos ambientes compatíveis que você escolheu para suas instâncias existentes.
   c. Insira valores para as entradas necessárias.
   d. Escolha Próximo.
8. Revise suas entradas e escolha Atualizar.
9. Depois que o status do serviço for`Active`, verifique seu e-mail para verificar se você recebeu AWS notificações que fornecem atualizações de status.

```
{
    "version": "0",
    "id": "af76c382-2b3c-7a0a-cf01-936dff228276",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:40:16Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "ACTIVE",
        "status": "UPDATE_IN_PROGRESS",
        "name": "your-service"
    }
}
```

```
{
    "version": "0",
    "id": "87131e29-ad95-bda2-cd30-0ce825dfb0cd",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:42:27Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
```

```
        "previousStatus": "UPDATE_IN_PROGRESS",
        "status": "ACTIVE",
        "name": "your-service"
    }
}
```

Etapa 4: limpar

Exclua seu tópico e assinatura do Amazon SNS e exclua sua EventBridge regra.

Exclua o tópico e a assinatura do Amazon SNS.

1. Navegue até o console do Amazon SNS.
2. No painel de navegação, selecione Subscriptions (Assinaturas).
3. Na página Assinaturas, escolha a assinatura que você fez para o tópico nomeado `tutorial-service-status-change` e escolha Excluir.
4. No painel de navegação, escolha Tópicos.
5. Na página Tópicos, escolha o tópico chamado `tutorial-service-status-change` e então Excluir..
6. Um modal solicita que você verifique a exclusão. Siga as instruções e escolha Excluir.

Exclua sua EventBridge regra.

1. Navegue até o EventBridge console da Amazon.
2. No painel de navegação, escolha Events (Eventos), Rules (Regras).
3. Na página Regras, escolha a regra chamada `tutorial-rule` e, em seguida, escolha Excluir.
4. Um modal solicita que você verifique a exclusão. Escolha Excluir.

Exclua a instância de serviço adicionada.

1. Navegue até o console do AWS Proton.
2. No painel de navegação, escolha Serviços.
3. Na página de Serviços, selecione o nome do seu serviço.
4. Na página Detalhes do serviço, escolha Editar e, em seguida, Avançar.

5. Na página Definir configurações personalizadas, na seção Instâncias de serviço, escolha Excluir para a instância de serviço que você criou como parte deste tutorial e, em seguida, escolha Avançar.
6. Revise suas entradas e escolha Atualizar.
7. Um modal solicita que você verifique a exclusão. Siga as instruções e escolha Sim, excluir.

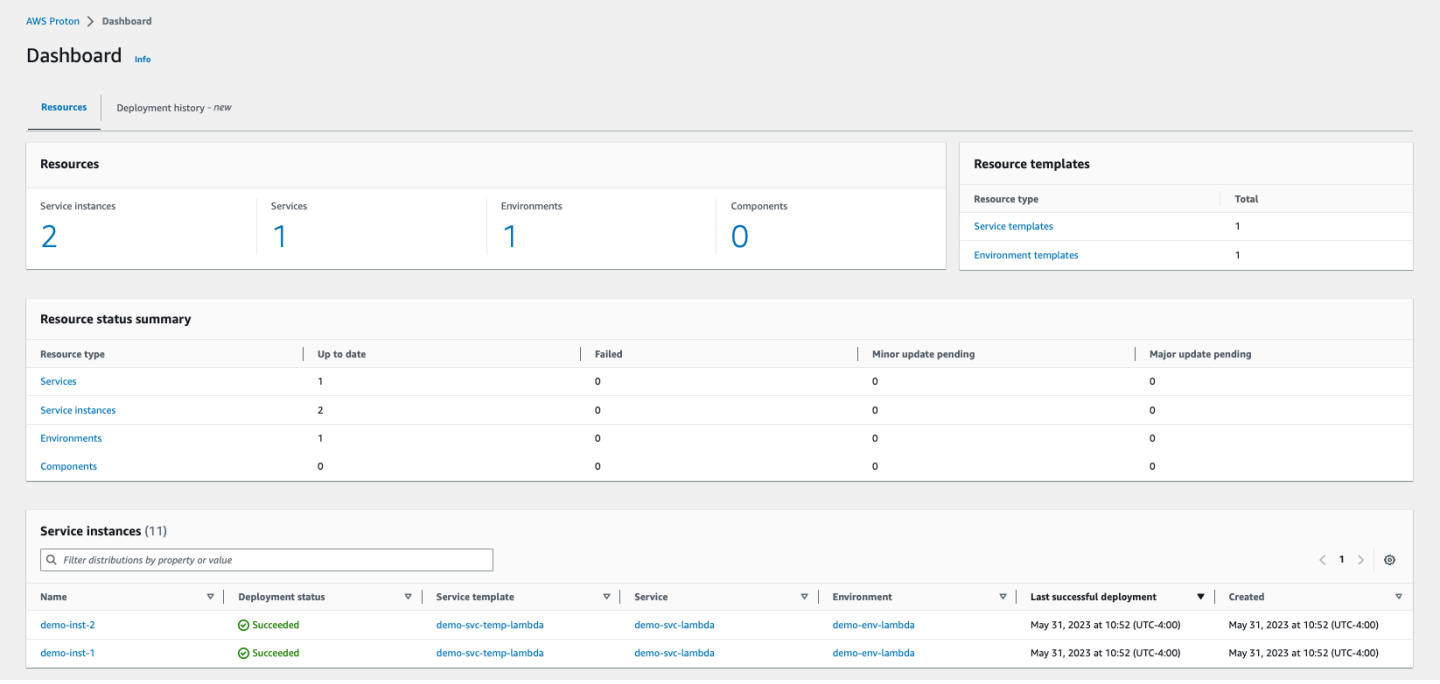
Mantenha a infraestrutura atualizada com o AWS Proton painel

O AWS Proton painel fornece um resumo dos AWS Proton recursos em sua AWS conta, com um foco particular na obsolescência — na atualização dos recursos implantados. Um recurso implantado está atualizado quando usa a versão recomendada do modelo associado. Um recurso out-of-date implantado pode precisar de uma atualização principal ou secundária da versão do modelo.

Visualize o painel no AWS Proton console

Para visualizar o AWS Proton painel, abra o AWS Proton console e, no painel de navegação, escolha Painel.

Recursos



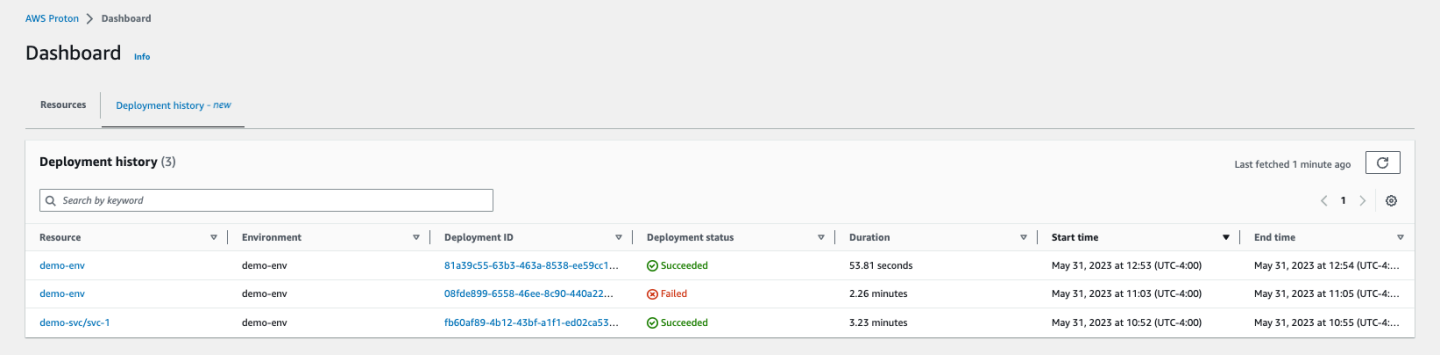
A primeira guia do painel exibe as contagens de todos os recursos em sua conta. A guia de recursos mostra o número de suas instâncias de serviço, serviços, ambientes e componentes, bem como

seus modelos de recursos. Ele também divide as contagens de recursos para cada tipo de recurso implantado pelo status dos recursos desse tipo. Uma tabela de instâncias de serviço mostra detalhes de cada instância de serviço: seu status de implantação, os AWS Proton recursos aos quais ela está associada, as atualizações que estão disponíveis e alguns registros de data e hora.

Você pode filtrar a lista de instâncias de serviço por qualquer propriedade da tabela. Por exemplo, você pode filtrar para ver instâncias de serviço com implantações dentro de uma janela de tempo específica ou instâncias de serviço que estão desatualizadas em relação às recomendações de versões principais ou secundárias.

Escolha um nome de instância de serviço para navegar até a página de detalhes da instância de serviço, onde você pode agir para fazer as atualizações de versão apropriadas. Escolha qualquer outro nome de AWS Proton recurso para navegar até sua página de detalhes ou escolha um tipo de recurso para navegar até a respectiva lista de recursos.

Histórico de implantação



A guia Histórico de implantação permite que você veja detalhes sobre suas implantações. Na tabela do histórico de implantação, você pode acompanhar o status da implantação, bem como o ambiente e a ID da implantação. Você pode escolher o nome do recurso ou o ID de implantação para ver ainda mais detalhes, como uma mensagem de status de implantação e saídas de recursos. A tabela também permite filtrar por qualquer propriedade da tabela.

Segurança em AWS Proton

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que é executada Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade aplicáveis AWS Proton, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS em Escopo por Programa .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS service (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.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar AWS Proton. Os tópicos a seguir mostram como configurar o AWS Proton para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros Serviços da AWS que o ajudem a monitorar e proteger seus AWS Proton recursos.

Tópicos

Identity and Access Management para AWS Proton

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar AWS Proton os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS Proton de identidade e acesso).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como AWS Proton funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Exemplos de políticas baseadas em identidade para AWS Proton)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como AWS Proton funciona com o IAM

Antes de usar o IAM para gerenciar o acesso AWS Proton, saiba com quais recursos do IAM estão disponíveis para uso AWS Proton.

Recursos do IAM que você pode usar com AWS Proton

Recurso do IAM	AWS Proton apoio
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de políticas	Sim
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Sim

Para ter uma visão de alto nível de como AWS Proton e outros Serviços da AWS funcionam com a maioria dos recursos do IAM, consulte Serviços da AWS esse trabalho com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para AWS Proton

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas

políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para AWS Proton

Para ver exemplos de políticas AWS Proton baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Proton

Políticas baseadas em recursos dentro AWS Proton

Compatibilidade com políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações políticas para AWS Proton

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de AWS Proton ações, consulte Ações definidas por AWS Proton na Referência de Autorização de Serviço.

As ações de política AWS Proton usam o seguinte prefixo antes da ação:

```
proton
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `List`, inclua a seguinte ação:

```
"Action": "proton:List*"
```

Para ver exemplos de políticas AWS Proton baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Proton

Recursos políticos para AWS Proton

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de AWS Proton recursos e seus ARNs, consulte Recursos definidos por AWS Proton na Referência de Autorização de Serviço. Para saber com quais ações é possível especificar o ARN de cada atributo, consulte Ações definidas pelo AWS Proton.

Para ver exemplos de políticas AWS Proton baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Proton

Chaves de condição de política para AWS Proton

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista de chaves de AWS Proton condição, consulte Chaves de condição AWS Proton na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas por AWS Proton.

Para ver um exemplo condition-key-based de política para limitar o acesso a um recurso, consulteExemplos de políticas baseadas em chaves de condição para AWS Proton.

Listas de controle de acesso (ACLs) em AWS Proton

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

As listas de controle de acesso (ACLs) são listas de beneficiários que você pode anexar aos recursos. Elas concedem às contas permissões para acessar o recurso ao qual são anexadas.

Controle de acesso baseado em atributos (ABAC) com AWS Proton

Compatível com ABAC (tags em políticas): sim

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Para obter mais informações sobre a marcação de AWS Proton recursos, consulteAWS Proton recursos e marcação.

Usando credenciais temporárias com AWS Proton

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Permissões principais entre serviços para AWS Proton

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Funções de serviço para AWS Proton

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

Para obter mais informações, consulte AWS Proton Exemplos de políticas de função de serviço do IAM.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper AWS Proton a funcionalidade. Edite as funções de serviço somente quando AWS Proton fornecer orientação para fazer isso.

Funções vinculadas a serviços para AWS Proton

Compatibilidade com perfis vinculados a serviços: sim

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter mais informações, consulte Usando funções vinculadas a serviços para AWS Proton.

Exemplos de políticas para AWS Proton

Encontre exemplos de políticas AWS Proton do IAM nas seções a seguir.

Tópicos

- Exemplos de políticas baseadas em identidade para AWS Proton
- AWS Proton Exemplos de políticas de função de serviço do IAM
- Exemplos de políticas baseadas em chaves de condição para AWS Proton

Exemplos de políticas baseadas em identidade para AWS Proton

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do AWS Proton . Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos por AWS Proton, incluindo o formato do ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição AWS Proton na Referência de Autorização de Serviço.

Tópicos

- Práticas recomendadas de política
- Links para exemplos de políticas baseadas em identidade para AWS Proton

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir AWS Proton recursos em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access

Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Links para exemplos de políticas baseadas em identidade para AWS Proton

Links para exemplos de políticas baseadas em identidade para AWS Proton

- AWS políticas gerenciadas para AWS Proton
- AWS Proton Exemplos de políticas de função de serviço do IAM
- Exemplos de políticas baseadas em chaves de condição para AWS Proton

AWS Proton Exemplos de políticas de função de serviço do IAM

Os administradores possuem e gerenciam os recursos AWS Proton criados conforme definido pelos modelos de ambiente e serviço. Eles atribuem funções de serviço do IAM à conta que permitem AWS Proton criar recursos em seu nome. Os administradores fornecem as funções e AWS Key Management Service as chaves do IAM para recursos que, posteriormente, pertencem e são gerenciados pelos desenvolvedores quando AWS Proton implantam seu aplicativo como um AWS Proton serviço em um AWS Proton ambiente. Para obter mais informações sobre criptografia de dados AWS KMS e criptografia de dados, consulteProteção de dados em AWS Proton.

Uma função de serviço é uma função da Amazon Web Services (IAM) que permite AWS Proton fazer chamadas para recursos em seu nome. Se você especificar uma função de serviço, o AWS Proton usará as credenciais da função. Use uma função de serviço para especificar explicitamente as ações que AWS Proton podem ser executadas.

Você cria o perfil de serviço e a respectiva política de permissão com o serviço do IAM. Para obter mais informações sobre a criação de uma função de serviço, consulte Criação de uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.

AWS Proton função de serviço para provisionamento usando CloudFormation

Como membro da equipe da plataforma, você pode, como administrador, criar uma função de AWS Proton serviço e fornecê-la AWS Proton ao criar um ambiente como função de CloudFormation serviço do ambiente (o `protonServiceRoleArn` parâmetro da ação da CreateEnvironmentAPI). Essa função permite AWS Proton fazer chamadas de API para outros serviços em seu nome quando o ambiente ou qualquer uma das instâncias de serviço em execução nele usa o provisionamento AWS gerenciado e AWS CloudFormation a infraestrutura de provisionamento.

Recomendamos que você use a seguinte função do IAM e a política de confiança para sua função AWS Proton de serviço. Quando você usa o AWS Proton console para criar um ambiente e opta por criar uma nova função, essa é a política que é AWS Proton adicionada à função de serviço que ela cria para você. Ao definir o escopo da permissão nesta política, lembre-se de que ela AWS Proton falha nos `Access Denied` erros.

> **Important**
>
> Lembre-se de que as políticas mostradas nos exemplos a seguir concedem privilégios de administrador a qualquer pessoa que possa registrar um modelo em sua conta. Como não sabemos quais recursos você definirá em seus AWS Proton modelos, essas políticas têm amplas permissões. Recomendamos que você defina o escopo das permissões para os recursos específicos que serão implantados em seus ambientes.

AWS Proton exemplo de política de função de serviço para CloudFormation

*123456789012*Substitua pelo seu Conta da AWS ID.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
```

```
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "NotAction": [
        "organizations:*",
        "account:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeOrganization",
        "account:ListRegions"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
```

```
    }
  ]
}
```

AWS Proton política de confiança do serviço

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Política de função de serviço AWS de provisionamento gerenciado com escopo reduzido

Veja a seguir um exemplo de uma política de função de AWS Proton serviço com escopo reduzido que você pode usar se precisar apenas de AWS Proton serviços para provisionar recursos do S3.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton função de serviço para CodeBuild provisionamento

Como membro da equipe da plataforma, você pode, como administrador, criar uma função de AWS Proton serviço e fornecê-la AWS Proton ao criar um ambiente como função de CodeBuild serviço

do ambiente (o `codebuildRoleArn` parâmetro da ação da CreateEnvironmentAPI). Essa função permite AWS Proton fazer chamadas de API para outros serviços em seu nome quando o ambiente ou qualquer uma das instâncias de serviço em execução nele usa o provisionamento para CodeBuild provisionar a infraestrutura.

Quando você usa o AWS Proton console para criar um ambiente e opta por criar uma nova função, AWS Proton adiciona uma política com privilégios de administrador à função de serviço que ela cria para você. Ao criar sua própria função e reduzir o escopo das permissões, lembre-se de que isso AWS Proton falha nos `Access Denied` erros.

> **Important**
>
> Esteja ciente de que as políticas AWS Proton associadas às funções que ela cria para você concedem privilégios de administrador a qualquer pessoa que possa registrar um modelo em sua conta. Como não sabemos quais recursos você definirá em seus AWS Proton modelos, essas políticas têm amplas permissões. Recomendamos que você defina o escopo das permissões para os recursos específicos que serão implantados em seus ambientes.

AWS Proton exemplo de política de função de serviço para CodeBuild

O exemplo a seguir fornece permissões CodeBuild para provisionar recursos usando AWS Cloud Development Kit (AWS CDK) o.

*123456789012*Substitua pelo seu Conta da AWS ID.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
   {
     "Action": [
       "logs:CreateLogStream",
       "logs:CreateLogGroup",
       "logs:PutLogEvents"
     ],
     "Resource": [
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
 Shell-*",
```

```
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
  Shell-*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": "proton:NotifyResourceDeploymentStatusChange",
      "Resource": "arn:aws:proton:us-east-1:123456789012:*",
      "Effect": "Allow"
    },
    {
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::123456789012:role/cdk-*-deploy-role-*",
        "arn:aws:iam::123456789012:role/cdk-*-file-publishing-role-*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS Proton CodeBuild política de confiança

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "CodeBuildTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
```

```
  }
}
```

AWS Proton funções do serviço de pipeline

Para provisionar pipelines de serviços, AWS Proton precisa de permissões para fazer chamadas de API para outros serviços. Os perfis de serviço necessários são semelhantes aos perfis de serviço que você fornece ao criar ambientes. No entanto, as funções para criar pipelines são compartilhadas entre todos os serviços da sua AWS conta, e você fornece essas funções como configurações da conta no console ou por meio da ação da UpdateAccountSettingsAPI.

Quando você usa o AWS Proton console para atualizar as configurações da conta e opta por criar uma nova função para as funções de CodeBuild serviço CloudFormation ou para as funções de serviço, as políticas AWS Proton adicionadas às funções de serviço criadas para você são as mesmas descritas nas seções anteriores Perfil de provisionamento gerenciado pela AWS CodeBuild função de provisionamento e. Ao definir o escopo da permissão nesta política, lembre-se de que ela AWS Proton falha nos `Access Denied` erros.

> **Important**
>
> Lembre-se de que as políticas exemplificadas nas seções anteriores concedem privilégios de administrador a qualquer pessoa que possa registrar um modelo em sua conta. Como não sabemos quais recursos você definirá em seus AWS Proton modelos, essas políticas têm amplas permissões. Recomendamos que você defina o escopo das permissões para os recursos específicos que serão implantados em seus pipelines.

AWS Proton função do componente

Como membro da equipe da plataforma, você pode, como administrador, criar uma função de AWS Proton serviço e fornecê-la AWS Proton ao criar um ambiente como função de CloudFormation componente do ambiente (o `componentRoleArn` parâmetro da ação da CreateEnvironmentAPI). Esse perfil abrange a infraestrutura que componentes diretamente definidos podem provisionar. Para obter mais informações sobre componentes, consulte Componentes.

O exemplo de política a seguir oferece suporte à criação de um componente diretamente definido que provisiona um bucket do Amazon Simple Storage Service (Amazon S3) e uma política de acesso relacionada.

AWS Proton exemplo de política de função de componente

*123456789012*Substitua pelo seu Conta da AWS ID.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
```

```
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS Proton política de confiança de componentes

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Exemplos de políticas baseadas em chaves de condição para AWS Proton

O exemplo a seguir da política do IAM nega acesso a AWS Proton ações que correspondam aos modelos especificados no `Condition` bloco. Observe que essas chaves de condição são suportadas somente pelas ações listadas em Ações, recursos e chaves de condição

para AWS Proton. Para gerenciar permissões em outras ações, como, por exemplo, `DeleteEnvironmentTemplate`, você deve usar o controle de acesso em nível de recurso.

Exemplo de política que nega ações AWS Proton de modelo em modelos específicos:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:EnvironmentTemplate":
 ["arn:aws:proton:region_id:123456789012:environment-template/my-environment-
template"]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 ["arn:aws:proton:region_id:123456789012:service-template/my-service-template"]
                }
            }
        }
    ]
}
```

No próximo exemplo de política, a primeira instrução em nível de recurso nega acesso às ações do AWS Proton modelo, exceto aquelas que `ListServiceTemplates` correspondam ao modelo de serviço listado no bloco. `Resource` A segunda declaração nega o acesso às AWS Proton ações que correspondem ao modelo listado no `Condition` bloco.

Exemplo de política que nega AWS Proton ações que correspondam a um modelo específico:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
        },
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate": [
                        "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
                    ]
                }
            }
        }
    ]
}
```

O exemplo final da política permite AWS Proton ações do desenvolvedor que correspondam ao modelo de serviço específico listado no `Condition` bloco.

Exemplo de política para permitir ações AWS Proton do desenvolvedor que correspondam a um modelo específico:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListServices",
                "proton:ListServiceInstances",
                "proton:ListEnvironments",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetEnvironment",
                "proton:CreateService",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "proton:DeleteService",
                "codestar-connections:ListConnections"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 "arn:aws:proton:region_id:123456789012:service-template/my-service-template"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService":
 "proton.amazonaws.com"
```

```
                }
            }
        }

    ]
}
```

AWS políticas gerenciadas para AWS Proton

Para adicionar permissões a usuários, grupos e funções, é mais fácil usar políticas AWS gerenciadas do que escrever políticas você mesmo. É necessário tempo e experiência para criar políticas gerenciadas pelo cliente do IAM que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, você pode usar nossas políticas AWS gerenciadas. Essas políticas abrangem casos de uso comuns e estão disponíveis na sua Conta da AWS. Para obter mais informações sobre políticas AWS gerenciadas, consulte políticas AWS gerenciadas no Guia do usuário do IAM.

Serviços da AWS manter e atualizar políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Os serviços ocasionalmente acrescentam permissões adicionais a uma política gerenciada pela AWS para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política gerenciada pela AWS quando um novo recurso for iniciado ou novas operações se tornarem disponíveis. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações de políticas não violarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política ReadOnlyAccess AWS gerenciada fornece acesso somente de leitura a todos Serviços da AWS os recursos. Quando um serviço inicia um novo atributo, a AWS adiciona permissões somente leitura para novas operações e atributos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.

AWS Proton fornece políticas gerenciadas de IAM e relações de confiança que você pode associar a usuários, grupos ou funções que permitem diferentes níveis de controle sobre recursos e operações de API. É possível aplicar essas políticas diretamente ou usá-las como ponto de partida para criar suas próprias políticas.

A relação de confiança a seguir é usada para cada uma das políticas AWS Proton gerenciadas.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ExampleTrustRelationshipWithProtonConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS política gerenciada: AWSProton FullAccess

Você pode anexar `AWSProtonFullAccess` às suas entidades do IAM. AWS Proton também anexa essa política a uma função de serviço que permite AWS Proton realizar ações em seu nome.

Essa política concede permissões administrativas que permitem acesso total às AWS Proton ações e acesso limitado a outras ações AWS de serviço que AWS Proton dependem de.

A política inclui os seguintes namespaces de ações-chave:

- `proton`— Permite que os administradores tenham acesso AWS Proton APIs total a.
- `iam`: Permite que os administradores passem perfis para o AWS Proton. Isso é necessário para que ele AWS Proton possa fazer chamadas de API para outros serviços em nome do administrador.
- `kms`: Permite que os administradores adicionem uma concessão a uma chave gerenciada pelo cliente.

- `codeconnections`— Permite que os administradores listem e passem conexões de código para que possam ser usadas por. AWS Proton

Para obter mais informações, consulte AWSProtonFullAccess.

AWS política gerenciada: AWSProton DeveloperAccess

Você pode anexar `AWSProtonDeveloperAccess` às suas entidades do IAM. AWS Proton também anexa essa política a uma função de serviço que permite AWS Proton realizar ações em seu nome.

Essa política concede permissões que permitem acesso limitado às AWS Proton ações e a outras AWS ações que AWS Proton dependem de. O escopo dessas permissões foi projetado para apoiar a função de um desenvolvedor que cria e implanta AWS Proton serviços.

Essa política não fornece acesso à criação, exclusão e atualização de AWS Proton modelos e ambientes APIs. Se os desenvolvedores precisarem de permissões ainda mais limitadas do que as fornecidas por essa política, recomendamos criar uma política personalizada com escopo reduzido para conceder o privilégio mínimo.

A política inclui os seguintes namespaces de ações-chave:

- `proton`— Permite que os colaboradores acessem um conjunto limitado de AWS Proton APIs.
- `codeconnections`— Permite que os colaboradores listem e passem conexões de código para que possam ser usadas por. AWS Proton

Para obter mais informações, consulte AWSProtonDeveloperAccess.

AWS política gerenciada: AWSProton ReadOnlyAccess

Você pode anexar `AWSProtonReadOnlyAccess` às suas entidades do IAM. AWS Proton também anexa essa política a uma função de serviço que permite AWS Proton realizar ações em seu nome.

Essa política concede permissões que permitem acesso somente leitura às AWS Proton ações e acesso limitado somente leitura a outras ações de AWS serviço que dependem de. AWS Proton

A política inclui os seguintes namespaces de ações-chave:

- `proton`— Permite que os colaboradores tenham acesso somente para leitura a. AWS Proton APIs

Para obter mais informações, consulte AWSProtonReadOnlyAccess.

AWS política gerenciada: AWSProton SyncServiceRolePolicy

AWS Proton anexa essa política à função AWSServiceRoleForProtonSyncvinculada ao serviço que permite realizar AWS Proton a sincronização de modelos.

Essa política concede permissões que permitem acesso limitado às AWS Proton ações e a outras ações AWS de serviço que AWS Proton dependem de.

A política inclui os seguintes namespaces de ações-chave:

- `proton`— Permite acesso limitado à AWS Proton sincronização AWS Proton APIs a.
- `codeconnections`— Permite acesso limitado à AWS Proton sincronização CodeConnections APIs a.

Para obter mais informações, consulte AWSProtonSyncServiceRolePolicy.

AWS política gerenciada: AWSProton CodeBuildProvisioningBasicAccess

As permissões CodeBuild precisam executar uma compilação para AWS Proton CodeBuild provisionamento. Você pode se associar `AWSProtonCodeBuildProvisioningBasicAccess` à sua função de CodeBuild aprovisionamento.

Essa política concede as permissões mínimas para o AWS Proton CodeBuild funcionamento do provisionamento. Ele concede permissões que permitem CodeBuild gerar registros de construção. Também concede permissão para o Proton disponibilizar saídas de Infraestrutura como Código (IaC) para os usuários. AWS Proton Não fornece as permissões necessárias às ferramentas do IaC para gerenciar a infraestrutura.

A política inclui os seguintes namespaces de ações-chave:

- `logs`- Permite CodeBuild gerar registros de construção. Sem essa permissão, não CodeBuild conseguirá iniciar.
- `proton`- Permite que um comando CodeBuild de provisionamento solicite a atualização das saídas do IAAC `aws proton notify-resource-deployment-status-change` para um determinado recurso. AWS Proton

Para obter mais informações, consulte AWSProtonCodeBuildProvisioningBasicAccess.

AWS política gerenciada: AWSProton CodeBuildProvisioningServiceRolePolicy

AWS Proton anexa essa política à função AWSServiceRoleForProtonCodeBuildProvisioningvinculada ao serviço que permite realizar CodeBuild o provisionamento AWS Proton baseado.

Essa política concede permissões que permitem acesso limitado às ações AWS de serviço que AWS Proton dependem de.

A política inclui os seguintes namespaces de ações-chave:

- `cloudformation`— Permite acesso limitado ao provisionamento AWS Proton CodeBuild baseado a. CloudFormation APIs
- `codebuild`— Permite acesso limitado ao provisionamento AWS Proton CodeBuild baseado a. CodeBuild APIs
- `iam`: Permite que os administradores passem perfis para o AWS Proton. Isso é necessário para que ele AWS Proton possa fazer chamadas de API para outros serviços em nome do administrador.
- `servicequotas`— Permite AWS Proton verificar o limite de compilação CodeBuild simultânea, o que garante o enfileiramento adequado da construção.

Para obter mais informações, consulte AWSProtonCodeBuildProvisioningServiceRolePolicy.

AWS política gerenciada: AWSProton ServiceGitSyncServiceRolePolicy

AWS Proton anexa essa política à função AWSServiceRoleForProtonServiceSyncvinculada ao serviço que permite realizar AWS Proton a sincronização do serviço.

Essa política concede permissões que permitem acesso limitado às AWS Proton ações e a outras ações AWS de serviço que AWS Proton dependem de.

A política inclui os seguintes namespaces de ações-chave:

- `proton`— Permite acesso limitado à AWS Proton sincronização AWS Proton APIs a.

Para obter mais informações, consulte AWSProtonServiceGitSyncServiceRolePolicy.

AWS Proton atualizações nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas AWS Proton desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nessa página, assine o feed RSS na página Histórico do AWS Proton documento.

Alteração	Descrição	Data
AWSProtonCodeBuildProvisioningServiceRolePolicy: atualizar para uma política existente	A política gerenciada para a função vinculada ao serviço, que permite AWS Proton realizar o provisionamento CodeBuild baseado, agora concede permissões para chamar as ações e a CloudFormation `TagResource` API. `UntagResource` Essas permissões são necessárias para realizar operações de marcação em recursos.	15 de junho de 2024
AWSProtonFullAccess – atualização para uma política existente	A política gerenciada para a função vinculada ao serviço de usar o Git sync com repositórios Git foi atualizada para recursos com os dois prefixos de serviço. Para obter mais informações, consulte Uso de funções vinculadas a serviços para AWS CodeConnections e políticas gerenciadas.	25 de abril de 2024
AWSProtonDeveloperAccess – atualização para uma política existente	A política gerenciada para a função vinculada ao serviço de usar o Git sync com repositórios Git foi atualizad	25 de abril de 2024

Alteração	Descrição	Data
	a para recursos com os dois prefixos de serviço. Para obter mais informações, consulte Uso de funções vinculadas a serviços para AWS CodeConnections e políticas gerenciadas.	
AWSProtonSyncServiceRolePolicy – atualização para uma política existente	A política gerenciada para a função vinculada ao serviço de usar o Git sync com repositórios Git foi atualizad a para recursos com os dois prefixos de serviço. Para obter mais informações, consulte Uso de funções vinculadas a serviços para AWS CodeConnections e políticas gerenciadas.	25 de abril de 2024
AWSProtonCodeBuildProvisioningServiceRolePolicy – atualização para uma política existente	AWS Proton atualizou essa política para adicionar permissões para garantir que as contas tenham o limite de criação CodeBuild simultâne a necessário para usar o CodeBuild provisionamento.	12 de maio de 2023

Alteração	Descrição	Data
AWSProtonServiceGitSyncServiceRolePolicy – Nova política	AWS Proton adicionou uma nova política para permitir AWS Proton a sincronização de serviços. A política é usada na função AWSServiceRoleForProtonServiceSyncvinculada ao serviço.	31 de março de 2023
AWSProtonDeveloperAccess – atualização para uma política existente	AWS Proton adicionou uma nova `GetResourcesSummary` ação que permite que você visualize um resumo de seus modelos, recursos de modelo implantados e recursos desatualizados.	18 de novembro de 2022
AWSProtonReadOnlyAccess – atualização para uma política existente	AWS Proton adicionou uma nova `GetResourcesSummary` ação que permite que você visualize um resumo de seus modelos, recursos de modelo implantados e recursos desatualizados.	18 de novembro de 2022
AWSProtonCodeBuildProvisioningBasicAccess – Nova política	AWS Proton adicionou uma nova política que fornece CodeBuild as permissões necessárias para executar uma compilação para AWS Proton CodeBuild Provisioning.	16 de novembro de 2022

Alteração	Descrição	Data
AWSProtonSyncServiceRolePolicy – Nova política	AWS Proton adicionou uma nova política para permitir AWS Proton a execução de operações relacionadas ao provisionamento CodeBuild baseado. A política é usada na função AWSServiceRoleForProtonCodeBuildProvisioningvinculada ao serviço.	2 de setembro de 2022
AWSProtonFullAccess – atualização para uma política existente	AWS Proton atualizou esta política para fornecer acesso a novas operações de AWS Proton API e corrigir problemas de permissão para algumas operações AWS Proton do console.	30 de março de 2022
AWSProtonDeveloperAccess – atualização para uma política existente	AWS Proton atualize esta política para fornecer acesso a novas operações de AWS Proton API e corrigir problemas de permissão para algumas operações AWS Proton do console.	30 de março de 2022
AWSProtonReadOnlyAccess – atualização para uma política existente	AWS Proton atualize esta política para fornecer acesso a novas operações de AWS Proton API e corrigir problemas de permissão para algumas operações AWS Proton do console.	30 de março de 2022

Alteração	Descrição	Data
AWSProtonSyncServiceRolePolicy – Nova política	AWS Proton adicionou uma nova política para permitir AWS Proton a execução de operações relacionadas à sincronização de modelos. A política é usada na função AWSServiceRoleForProtonSyncvinculada ao serviço.	23 de novembro de 2021
AWSProtonFullAccess – Nova política	AWS Proton adicionou uma nova política para fornecer acesso de funções administrativas às operações da AWS Proton API e ao AWS Proton console.	9 de junho de 2021
AWSProtonDeveloperAccess – Nova política	AWS Proton adicionou uma nova política para fornecer acesso à função de desenvolvedor às operações da AWS Proton API e ao AWS Proton console.	9 de junho de 2021
AWSProtonReadOnlyAccess – Nova política	AWS Proton adicionou uma nova política para fornecer acesso somente de leitura às operações AWS Proton da API e ao AWS Proton console.	9 de junho de 2021
AWS Proton começou a rastrear as alterações.	AWS Proton começou a rastrear as mudanças em suas políticas AWS gerenciadas.	9 de junho de 2021

Usando funções vinculadas a serviços para AWS Proton

AWS Proton usa funções vinculadas ao serviço AWS Identity and Access Management (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente a. AWS Proton As funções vinculadas ao serviço são predefinidas AWS Proton e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Tópicos

Usando funções para AWS Proton sincronização

AWS Proton usa funções vinculadas ao serviço AWS Identity and Access Management (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente a. AWS Proton As funções vinculadas ao serviço são predefinidas AWS Proton e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração AWS Proton porque você não precisa adicionar manualmente as permissões necessárias. AWS Proton define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, só AWS Proton pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus AWS Proton recursos porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas a serviços, consulte AWS Serviços que funcionam com IAM e procure os serviços que têm Sim na coluna Funções vinculadas ao serviço. Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculadas ao serviço para AWS Proton

AWS Proton usa duas funções vinculadas ao serviço chamadas e. AWSServiceRoleForProtonSyncAWSServiceRoleForProtonServiceSync

O perfil vinculado ao serviço AWSServiceRoleForProtonSync confia nos seguintes serviços para aceitar o perfil:

- `sync.proton.amazonaws.com`

A política de permissões de função nomeada `AWSProtonSyncServiceRolePolicy` AWS Proton permite concluir as seguintes ações nos recursos especificados:

- Action: create, manage, and read on AWS Proton templates and template versions
- Action: use connection on CodeConnections

Para obter mais informações sobre essa política, consulte AWS política gerenciada: AWSProton SyncServiceRolePolicy.

O perfil vinculado ao serviço AWSServiceRoleForProtonServiceSync confia nos seguintes serviços para aceitar o perfil:

- `service-sync.proton.amazonaws.com`

A política de permissões de função nomeada `AWSProtonServiceGitSyncServiceRolePolicy` AWS Proton permite concluir as seguintes ações nos recursos especificados:

- Ação: criar, gerenciar e ler sobre AWS Proton serviços e instâncias de serviço

Para obter mais informações sobre essa política, consulte AWS política gerenciada: AWSProton ServiceGitSyncServiceRolePolicy.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço para AWS Proton

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você configura um repositório ou serviço para sincronização AWS Proton na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, AWS Proton cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você configura um repositório ou

serviço para sincronização AWS Proton, AWS Proton cria a função vinculada ao serviço para você novamente.

Para recriar a função AWSServiceRoleForProtonSyncvinculada ao serviço, você gostaria de configurar um repositório para sincronização e, para recriar AWSServiceRoleForProtonServiceSync, você gostaria de configurar um serviço para sincronização.

Editando uma função vinculada ao serviço para AWS Proton

AWS Proton não permite que você edite a função AWSServiceRoleForProtonSyncvinculada ao serviço. Depois que você criar um perfil vinculado ao serviço, não poderá alterar o nome do perfil, pois várias entidades podem fazer referência ao perfil. No entanto, você poderá editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluindo uma função vinculada ao serviço para AWS Proton

Você não precisa excluir manualmente a função AWSServiceRoleForProtonSync. Quando você exclui todos os repositórios AWS Proton vinculados para sincronização de repositórios na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, AWS Proton limpa os recursos e exclui a função vinculada ao serviço para você.

Regiões suportadas para funções vinculadas a AWS Proton serviços

AWS Proton suporta o uso de funções vinculadas ao serviço em todos os lugares em Regiões da AWS que o serviço está disponível. Para obter mais informações, consulte endpoints e cotas do AWS Proton na Referência geral da AWS.

Usando funções para provisionamento CodeBuild baseado

AWS Proton usa funções vinculadas ao serviço AWS Identity and Access Management (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente a. AWS Proton As funções vinculadas ao serviço são predefinidas AWS Proton e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração AWS Proton porque você não precisa adicionar manualmente as permissões necessárias. AWS Proton define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, só AWS Proton pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus AWS Proton recursos porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas a serviços, consulte AWS Serviços que funcionam com IAM e procure os serviços que têm Sim na coluna Funções vinculadas ao serviço. Escolha um Sim com um link para visualizar a documentação do perfil vinculado a esse serviço.

Permissões de função vinculadas ao serviço para AWS Proton

AWS Proton usa a função vinculada ao serviço chamada AWSServiceRoleForProtonCodeBuildProvisioning— Uma função vinculada ao serviço para AWS Proton CodeBuild provisionamento.

O perfil vinculado ao serviço AWSServiceRoleForProtonCodeBuildProvisioning confia nos seguintes serviços para aceitar o perfil:

- `codebuild.proton.amazonaws.com`

A política de permissões de função nomeada `AWSProtonCodeBuildProvisioningServiceRolePolicy` AWS Proton permite concluir as seguintes ações nos recursos especificados:

- Action: create, manage, and read on CloudFormation stacks and transforms
- Ação: criar, gerenciar e ler sobre CodeBuild projetos e construções

Para obter mais informações sobre essa política, consulte AWS política gerenciada: AWSProton CodeBuildProvisioningServiceRolePolicy.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço para AWS Proton

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um ambiente que usa provisionamento CodeBuild baseado na Console de gerenciamento da AWS, AWS Proton na ou na AWS API AWS CLI, AWS Proton cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um ambiente que usa provisionamento CodeBuild baseado em AWS Proton, AWS Proton cria a função vinculada ao serviço para você novamente.

Editando uma função vinculada ao serviço para AWS Proton

AWS Proton não permite que você edite a função AWSServiceRoleForProtonCodeBuildProvisioningvinculada ao serviço. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluindo uma função vinculada ao serviço para AWS Proton

Se você não precisar mais usar um recurso ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve excluir todos os ambientes e serviços (instâncias e pipelines) que usam o provisionamento CodeBuild baseado AWS Proton antes de poder excluí-lo manualmente.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForProtonCodeBuildProvisioningvinculada ao serviço. Para saber mais, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões suportadas para funções vinculadas a AWS Proton serviços

AWS Proton suporta o uso de funções vinculadas ao serviço em todos os lugares em Regiões da AWS que o serviço está disponível. Para obter mais informações, consulte endpoints e cotas do AWS Proton na Referência geral da AWS.

Solução de problemas AWS Proton de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS Proton um IAM.

Tópicos

- Não estou autorizado a realizar uma ação em AWS Proton

- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS Proton recursos

Não estou autorizado a realizar uma ação em AWS Proton

Se isso Console de gerenciamento da AWS indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `proton:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `proton:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS Proton.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no AWS Proton. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS Proton recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é AWS Proton compatível com esses recursos, consulteComo AWS Proton funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Análise de configuração e vulnerabilidade em AWS Proton

AWS Proton não fornece patches ou atualizações para o código fornecido pelo cliente. Os clientes são responsáveis por atualizar e aplicar patches em seu próprio código, incluindo o código-fonte de seus serviços e aplicativos em execução AWS Proton e o código fornecido em seus pacotes de modelos de serviços e ambientes.

Os clientes são responsáveis por atualizar e corrigir os recursos de infraestrutura em seus ambientes e serviços. AWS Proton não atualizará nem corrigirá automaticamente nenhum recurso. Os clientes devem consultar a documentação dos recursos em sua arquitetura para entender suas respectivas políticas de patches.

Além de fornecer atualizações de ambiente e serviço solicitadas pelo cliente para versões secundárias de serviços e modelos de ambiente, AWS Proton não fornece patches ou atualizações para os recursos que os clientes definem em seus modelos de serviços e ambientes e pacotes de modelos.

Para obter mais detalhes, consulte os seguintes recursos da :

- Modelo de responsabilidade compartilhada
- Amazon Web Services: visão geral dos processos de segurança

Proteção de dados em AWS Proton

AWS Proton está em conformidade com o modelo de responsabilidade AWS compartilhada modelo que inclui regulamentos e diretrizes para proteção de dados. AWS é responsável por proteger a infraestrutura global que executa todos os Serviços da AWS. AWS mantém o controle sobre os dados hospedados nessa infraestrutura, incluindo os controles de configuração de segurança para lidar com o conteúdo do cliente e os dados pessoais. AWS clientes e parceiros da APN, atuando como controladores ou processadores de dados, são responsáveis por quaisquer dados pessoais que coloquem no Nuvem AWS

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas de usuário individuais com AWS Identity and Access Management (IAM), para que cada usuário receba somente as permissões necessárias para cumprir suas tarefas. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Recomendamos usar o TLS 1.2 ou posterior.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de texto de formato livre, como um campo Nome. Isso inclui quando você trabalha com AWS Proton ou Serviços da AWS usa o console, a API ou AWS SDKs. AWS CLI Todos os dados inseridos em campos de texto de formato livre para

identificadores de recursos ou itens similares relacionados ao gerenciamento de recursos da AWS poderão ser selecionados para inclusão em logs de diagnóstico. Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar a solicitação a esse servidor.

Para mais informações sobre proteção de dados, consulte a publicação Modelo de responsabilidade compartilhada da AWS e do GDPR no Blog de segurança da AWS .

Criptografia do lado do servidor em repouso

Se você optar por criptografar dados confidenciais em seus pacotes de modelos em repouso no bucket do S3 em que você armazena seus pacotes de modelos, deverá usar uma chave SSE-S3 ou SSE-KMS para permitir a recuperação dos pacotes de modelos AWS Proton para que possam ser anexados a um modelo registrado. AWS Proton

Criptografia em trânsito

Todas as comunicações de serviço para serviço são criptografadas em trânsito usando SSL/TLS.

AWS Proton gerenciamento de chaves de criptografia

Dentro AWS Proton, todos os dados do cliente são criptografados por padrão usando uma AWS Proton chave própria. Se você fornecer uma AWS KMS chave gerenciada e de propriedade do cliente, todos os dados do cliente serão criptografados usando a chave fornecida pelo cliente, conforme descrito nos parágrafos a seguir.

Ao criar um AWS Proton modelo, você especifica sua chave e AWS Proton usa suas credenciais para criar uma concessão que permite AWS Proton usar sua chave.

Se você retirar manualmente a concessão ou desativar ou excluir a chave especificada, o AWS Proton não conseguirá ler os dados que foram criptografados pela chave especificada e cuja saída é `ValidationException`.

AWS Proton contexto de criptografia

AWS Proton suporta cabeçalhos de contexto de criptografia. Um contexto de criptografia é um conjunto opcional de pares chave-valor que pode conter informações contextuais adicionais sobre os dados. Para obter informações gerais sobre o contexto de criptografia, consulte Conceitos do AWS Key Management Service : contexto de criptografia no Guia do desenvolvedor do AWS Key Management Service .

Um contexto de criptografia é um conjunto de pares de chave-valor que contêm dados arbitrários não secretos. Quando inclui um contexto de criptografia em uma solicitação para criptografar dados, o AWS KMS vincula de forma criptográfica o contexto de criptografia aos dados criptografados. Para descriptografar os dados, você deve passar o mesmo contexto de criptografia.

Os clientes podem usar o contexto de criptografia para identificar o uso da chave gerenciada pelo cliente em registros de auditoria e logs. Ele também aparece em texto simples em registros, como AWS CloudTrail Amazon CloudWatch Logs.

AWS Proton não aceita nenhum contexto de criptografia especificado pelo cliente ou externamente.

AWS Proton adiciona o seguinte contexto de criptografia.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

O primeiro contexto de criptografia identifica o AWS Proton modelo ao qual o recurso está associado e também serve como uma restrição para permissões e concessões de chaves gerenciadas pelo cliente.

O segundo contexto de criptografia identifica o AWS Proton recurso que está criptografado.

Os exemplos a seguir mostram o uso do contexto de AWS Proton criptografia.

Desenvolvedor criando uma instância de serviço.

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service/my-service/
service-instance/my-service-instance"
}
```

Um administrador criando um modelo.

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
```

```
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service-template/my-
template"
}
```

Segurança da infraestrutura em AWS Proton

Como serviço gerenciado, AWS Proton é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar AWS Proton pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Para melhorar o isolamento da rede, você pode usar AWS PrivateLink conforme descrito na seção a seguir.

AWS Proton e endpoints VPC de interface ()AWS PrivateLink

Você pode estabelecer uma conexão privada entre sua VPC e criar uma AWS Proton interface VPC endpoint. Os endpoints de interface são alimentados por AWS PrivateLinkuma tecnologia que permite que você acesse de forma privada AWS Proton APIs sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão. AWS Direct Connect As instâncias em sua VPC não precisam de endereços IP públicos para se comunicar. AWS Proton APIs O tráfego entre sua VPC e AWS Proton o tráfego não sai da rede Amazon.

Cada endpoint de interface é representado por uma ou mais Interfaces de Rede Elástica nas sub-redes.

Para obter mais informações, consulte Endpoints da VPC da interface (AWS PrivateLink) no Manual do Usuário do Amazon VPC.

Considerações sobre AWS Proton VPC endpoints

Antes de configurar uma interface para o VPC endpoint AWS Proton, certifique-se de revisar as propriedades e limitações do endpoint da interface no Guia do usuário do Amazon VPC.

AWS Proton suporta fazer chamadas para todas as suas ações de API a partir de sua VPC.

Há suporte para políticas de endpoint de VPC. AWS Proton Por padrão, o acesso total ao AWS Proton é permitido por meio do endpoint. Para mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Criação de uma interface VPC endpoint para AWS Proton

Você pode criar um VPC endpoint para o AWS Proton serviço usando o console Amazon VPC ou o (). AWS Command Line Interface AWS CLI Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC.

Crie um VPC endpoint para AWS Proton usar o seguinte nome de serviço:

- com.amazonaws. *region*.próton

Se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API AWS Proton usando seu nome DNS padrão para a região, por exemplo,. `proton.region.amazonaws.com`

Para mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

Criação de uma política de VPC endpoint para AWS Proton

É possível anexar uma política de endpoint ao endpoint da VPC que controla o acesso ao AWS Proton. Essa política especifica as seguintes informações:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Exemplo: política de VPC endpoint para ações AWS Proton

Veja a seguir um exemplo de uma política de endpoint para AWS Proton. Quando anexada a um endpoint, essa política concede acesso às AWS Proton ações listadas para todos os diretores em todos os recursos.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServices",
        "proton:ListServiceInstances",
        "proton:ListEnvironments",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetEnvironment",
        "proton:CreateService",
        "proton:UpdateService",
        "proton:UpdateServiceInstance",
        "proton:UpdateServicePipeline",
        "proton:DeleteService"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Registro e monitoramento em AWS Proton

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS Proton suas outras AWS soluções. AWS fornece as seguintes ferramentas de

monitoramento para observar suas instâncias em execução AWS Proton, relatar quando algo está errado e realizar ações automáticas quando apropriado.

No momento, AWS Proton ele não está integrado ao Amazon CloudWatch Logs ou AWS Trusted Advisor. Os administradores podem configurar e usar CloudWatch para monitorar outros, Serviços da AWS conforme definido em seus modelos de serviço e ambiente. AWS Proton está integrado com AWS CloudTrail.

- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. Você pode coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas EC2 instâncias da Amazon e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.
- O Amazon CloudWatch Logs permite que você monitore, armazene e acesse seus arquivos de log de EC2 instâncias da Amazon e de outras fontes. CloudTrail CloudWatch Os registros podem monitorar as informações nos arquivos de log e notificá-lo quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por você ou em seu nome Conta da AWS e entrega os arquivos de log em um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaram AWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o Guia do usuário do AWS CloudTrail.
- EventBridgeA Amazon é um serviço de ônibus de eventos sem servidor que facilita a conexão de seus aplicativos com dados de várias fontes. EventBridge fornece um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos Software-as-a-Service (SaaS) Serviços da AWS e encaminha esses dados para destinos como o Lambda. Isso permite monitorar eventos que ocorram em serviços e criem arquiteturas orientadas a eventos. Para obter mais informações, consulte Automatize AWS Proton com EventBridge e o Manual do usuário do EventBridge .

Resiliência em AWS Proton

A infraestrutura AWS global é construída em torno Região da AWS de zonas de disponibilidade. Região da AWS s fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e altamente redundantes. Com

as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre Região da AWS s e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, AWS Proton oferece recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

AWS Proton cópias de segurança

AWS Proton mantém um backup de todos os dados do cliente. No caso de uma paralisação total, esse backup pode ser usado para restaurar AWS Proton os dados do cliente de um estado válido anterior.

Práticas recomendadas de segurança para AWS Proton

AWS Proton fornece recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

Tópicos

- Usar o IAM para controlar o acesso
- Não incorpore credenciais em seus modelos e pacotes de modelos.
- Use criptografia para proteger dados confidenciais
- Use AWS CloudTrail para visualizar e registrar chamadas de API

Usar o IAM para controlar o acesso

O IAM é um AWS service (Serviço da AWS) que você pode usar para gerenciar usuários e suas permissões no AWS. Você pode usar o IAM com AWS Proton para especificar quais AWS Proton ações os administradores e desenvolvedores podem realizar, como gerenciar modelos, ambientes

ou serviços. Você pode usar as funções de serviço do IAM AWS Proton para permitir fazer chamadas para outros serviços em seu nome.

Para obter mais informações sobre AWS Proton as funções do IAM, consulteIdentity and Access Management para AWS Proton.

Implemente o acesso de privilégio mínimo. Para ter mais informações, consulte Políticas e permissões no IAM, no Guia do usuário do AWS Identity and Access Management .

Não incorpore credenciais em seus modelos e pacotes de modelos.

Em vez de incorporar informações confidenciais em seus CloudFormation modelos e pacotes de modelos, recomendamos que você use referências dinâmicas em seu modelo de pilha.

As referências dinâmicas fornecem uma maneira compacta e poderosa de referenciar valores externos que são armazenados e gerenciados em outros serviços, como o AWS Systems Manager Parameter Store ou AWS Secrets Manager. Quando você usa uma referência dinâmica, CloudFormation recupera o valor da referência especificada quando necessário durante as operações de pilha e conjunto de alterações e passa o valor para o recurso apropriado. No entanto, CloudFormation nunca armazena o valor de referência real. Para obter mais informações, consulte Usar referências dinâmicas para especificar valores de modelo no Guia do Usuário do CloudFormation .

O AWS Secrets Manager ajuda você a criptografar, armazenar e recuperar credenciais com segurança para bancos de dados e outros serviços. A Loja de parâmetros do AWS Systems Manager fornece armazenamento hierárquico seguro para o gerenciamento de dados de configuração.

Para obter mais informações sobre definir parâmetros de modelos, consulte https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html no Guia do usuário do CloudFormation .

Use criptografia para proteger dados confidenciais

Dentro AWS Proton, todos os dados do cliente são criptografados por padrão usando uma AWS Proton chave própria.

Como membro da equipe da plataforma, você pode fornecer uma chave gerenciada pelo cliente AWS Proton para criptografar e proteger seus dados confidenciais. Criptografe dados em repouso

confidenciais em seu bucket do S3. Para obter mais informações, consulte Proteção de dados em AWS Proton.

Use AWS CloudTrail para visualizar e registrar chamadas de API

AWS CloudTrail rastreia qualquer pessoa que faça chamadas de API no seu Conta da AWS. As chamadas de API são registradas sempre que alguém usa a AWS Proton API, o AWS Proton console ou AWS Proton AWS CLI os comandos. Ative o registro em log e especifique um bucket do Amazon S3 para armazenar os logs. Dessa forma, se precisar, você pode auditar quem fez qual AWS Proton chamada em sua conta. Para obter mais informações, consulte Registro e monitoramento em AWS Proton.

Prevenção contra o ataque do “substituto confuso” em todos os serviços

“Confused deputy” é um problema de segurança no qual uma entidade sem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Em AWS, a falsificação de identidade entre serviços pode resultar em um problema confuso de delegado. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta.

Recomendamos usar as chaves de contexto de condição global `aws:SourceAccount` e `aws:SourceArn` nas políticas de recursos para limitar as permissões que AWS Proton concedem outro serviço ao recurso. Se o valor de `aws:SourceArn` não contém ID da conta, como um ARN do bucket do Amazon S3, você deve usar ambas as chaves de contexto de condição global para limitar as permissões. Se você usa ambas as chaves de contexto de condição global, e o valor `aws:SourceArn` contém o ID da conta, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão utilizar a mesma ID de conta quando na mesma declaração de política. Use `aws:SourceArn` se quiser apenas um recurso associado a acessibilidade de serviço. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

O valor de `aws:SourceArn` deve ser um recurso que AWS Proton armazena.

A maneira mais eficaz de se proteger do problema ‘confused deputy’ é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou se especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com curingas (*) para as partes desconhecidas do ARN. Por exemplo, .`arn:aws::proton:*:`*`123456789012`*`:environment/*`

O exemplo a seguir mostra como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto AWS Proton para evitar o confuso problema substituto.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ExampleProtonConfusedDeputyPreventionPolicy",
        "Effect": "Allow",
        "Principal": {"Service": "proton.amazonaws.com"},
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
            }
        }
    }
}
```

CodeBuild provisionamento de suporte personalizado à Amazon VPC

AWS Proton CodeBuild O provisionamento executa comandos CLI arbitrários fornecidos pelo cliente em um projeto localizado na CodeBuild conta Environment. AWS Proton Esses comandos normalmente gerenciam recursos usando uma ferramenta de Infraestrutura como Código (IaC), como CDK. Se você tiver recursos em uma Amazon VPC, CodeBuild talvez não consiga acessá-

los. Para permitir isso, CodeBuild oferece suporte à capacidade de execução em uma Amazon VPC específica. Alguns exemplos de casos de uso incluem:

- Recuperar dependências de repositórios de artefatos auto-hospedados, internos, como `PyPI` para Python, `Maven` para Java e `npm` para Node.js.
- CodeBuild precisa acessar um servidor Jenkins em uma Amazon VPC específica para registrar um pipeline.
- Acessar objetos em um bucket do Amazon S3 configurado para permitir acesso apenas por meio de um endpoint da VPC da Amazon.
- Executar testes de integração na compilação em relação a dados em um banco de dados do Amazon RDS isolado em uma sub-rede privada.

Para obter mais informações, consulte a CodeBuild documentação da VPC.

Se você quiser que o CodeBuild provisionamento seja executado em uma VPC personalizada, AWS Proton fornece uma solução simples. Primeiro, você deve adicionar a ID da VPC, as sub-redes e os grupos de segurança ao modelo de ambiente. Em seguida, você insere esses valores na especificação do ambiente. Isso resultará na criação de um CodeBuild projeto para você que tem como alvo uma determinada VPC.

Atualizando o modelo de ambiente

Schema

O ID da VPC, as sub-redes e os grupos de segurança precisam ser adicionados ao esquema do modelo para que possam existir na especificação do ambiente.

Um exemplo de `schema.yaml`:

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentInputType"
  types:
    EnvironmentInputType:
      type: object
      properties:
        codebuild_vpc_id:
          type: string
        codebuild_subnets:
```

```
          type: array
          items:
            type: string
        codebuild_security_groups:
          type: array
          items:
            type: string
```

Isso adiciona três novas propriedades que serão usadas pelo manifesto:

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifesto

Para definir as configurações da Amazon VPC em CodeBuild, uma propriedade opcional chamada `project_properties` está disponível no modelo de manifesto. O conteúdo de `project_properties` é adicionado à CloudFormation pilha que cria o CodeBuild projeto. Isso possibilita adicionar não apenas CloudFormation propriedades da Amazon VPC, mas também qualquer CodeBuild CloudFormation propriedade compatível, como tempo limite de compilação. Os mesmos dados fornecidos ao `proton-inputs.json` são disponibilizados para os valores de `project_properties`.

Adicione esta seção ao seu `manifest.yaml`:

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Veja a seguir a aparência do `manifest.yaml` resultante:

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
```

```
      nodejs: 16
    provision:
      - npm install
      - npm run build
      - npm run cdk bootstrap
      - npm run cdk deploy -- --require-approval never
    deprovision:
      - npm install
      - npm run build
      - npm run cdk destroy -- --force
    project_properties:
      VpcConfig:
        VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
        Subnets: "{{ environment.inputs.codebuild_subnets }}"
        SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Criar o ambiente

Ao criar um ambiente com seu modelo habilitado para CodeBuild Provisioning VPC, você deve fornecer o ID, as sub-redes e os grupos de segurança da Amazon VPC.

Para obter uma lista de todas as Amazon VPC IDs em sua região, execute o seguinte comando:

```
aws ec2 describe-vpcs
```

Para obter uma lista de toda a sub-rede IDs, execute:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> Inclua somente sub-redes privadas. CodeBuild falhará se você fornecer sub-redes públicas. As sub-redes públicas têm uma rota padrão para um Gateway da Internet, enquanto as sub-redes privadas não.

Execute o comando a seguir para obter o grupo de segurança IDs. Eles também IDs podem ser obtidos por meio do Console de gerenciamento da AWS:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Os valores serão semelhantes a:

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

Garantindo CodeBuild permissões

O suporte ao Amazon VPC exige certas permissões, como a capacidade de criar uma interface de rede elástica.

Se o ambiente estiver sendo criado no console, insira esses valores durante o assistente de criação de ambiente. Se você quiser criar o ambiente de forma programática, a aparência do seu `spec.yaml` será a seguinte:

```

proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton recursos e marcação

AWS Proton os recursos atribuídos a um Amazon Resource Name (ARN) incluem modelos de ambiente e suas versões principais e secundárias, modelos de serviço e suas versões principais e secundárias, ambientes, serviços, instâncias de serviço, componentes e repositórios. Você pode colocar tags nesses recursos para ajudar a organizá-los e identificá-los. Você pode usar tags para categorizar recursos por finalidade, proprietário, ambiente ou outros critérios. Para obter mais informações, consulte Estratégias de marcação da . Para rastrear e gerenciar seus atributos do AWS Proton , você pode usar os atributos de marcação descritos nas seções a seguir.

AWS marcação

Você pode atribuir metadados aos seus AWS recursos na forma de tags. Cada tag consiste em uma chave definida pelo cliente e um valor opcional. As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos.

> **Important**
>
> Não adicione informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em tags. As tags são acessíveis a muitos Serviços da AWS, incluindo faturamento. As tags não devem ser usadas para dados privados ou confidenciais.

Cada tag da tem duas partes.

- Uma chave de tag (por exemplo ,`CostCenter`, `Environment` ou `Project`). As chaves de tag diferenciam maiúsculas de minúsculas
- Um valor de tag (opcional) (por exemplo, `111122223333` ou `Production`). Assim como as chaves de tag, os valores de tag diferenciam maiúsculas de minúsculas.

Os requisitos básicos de uso e de nomenclatura a seguir se aplicam às tags.

- Cada recurso pode ter no máximo 50 tags criadas pelo usuário.

> **Note**
>
> As tags criadas pelo sistema que começam com o `aws:` prefixo são reservadas para AWS uso e não contam para esse limite. Não é possível editar nem excluir uma tag que começa com o prefixo `aws:`.

- Em todos os recursos, cada chave de tag deve ser exclusiva e possuir apenas um valor.
- A chave da tag deve ter no mínimo 1 e no máximo 128 caracteres Unicode em UTF-8.
- O valor da tag deve ter no mínimo 1 e no máximo de 256 caracteres Unicode em UTF-8.
- Os caracteres permitidos nas tags são letras, números, espaços representáveis em UTF-8 e os seguintes caracteres:* _ . : / = + - @.

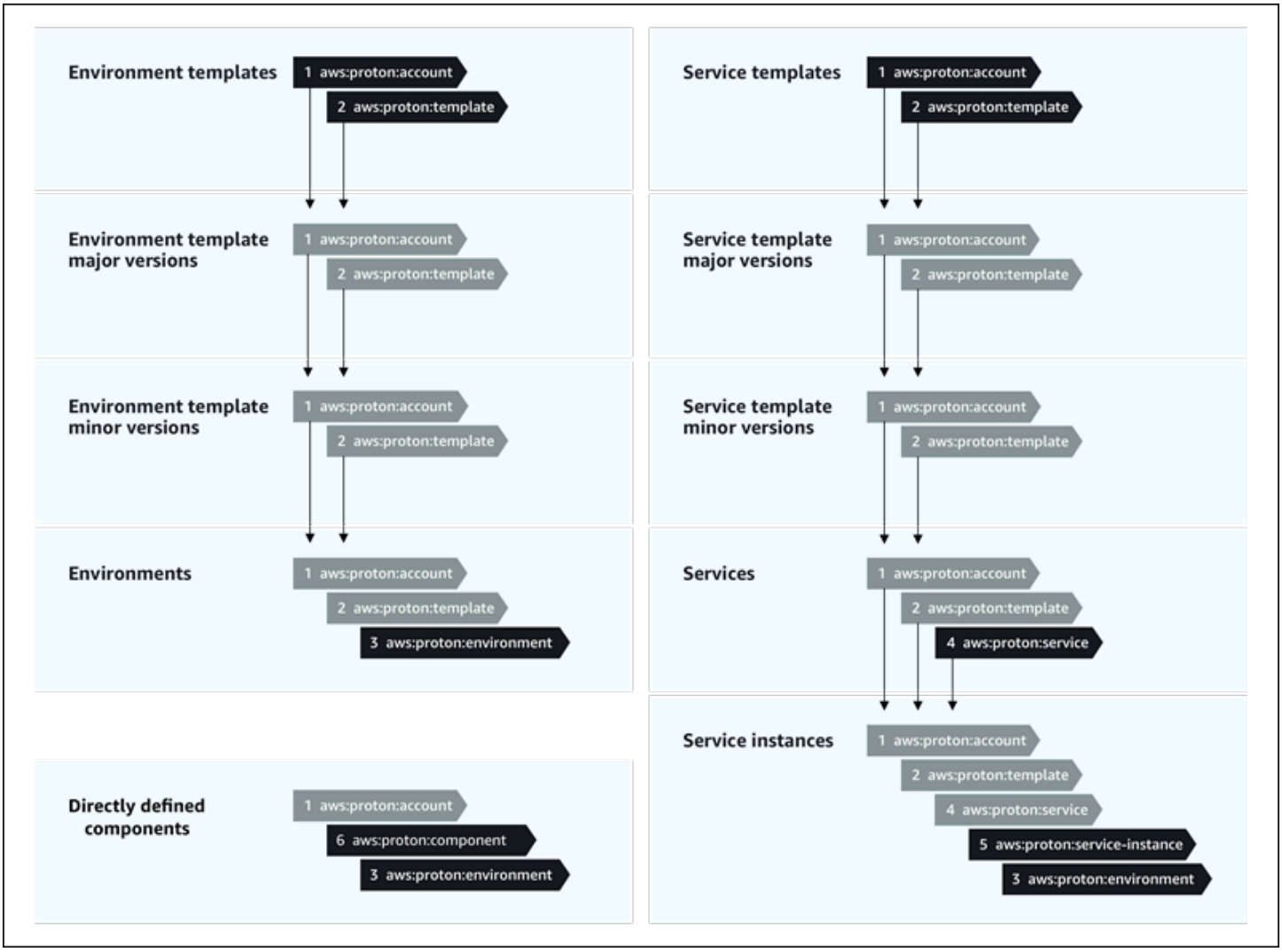
AWS Proton marcação

Com AWS Proton, você pode usar tanto as tags que você cria quanto as tags que AWS Proton são geradas automaticamente para você.

AWS Proton AWS tags gerenciadas

Quando você cria um AWS Proton recurso, gera AWS Proton automaticamente tags AWS gerenciadas para seu novo recurso, conforme mostrado no diagrama a seguir. AWS posteriormente, as tags gerenciadas se propagam para outros AWS Proton recursos baseados em seu novo recurso. Por exemplo, as tags gerenciadas de um modelo de ambiente se propagam para suas versões e as tags gerenciadas de um serviço se propagam para suas instâncias de serviço.

> **Note**
>
> AWS as tags gerenciadas não são geradas para conexões de contas de ambiente. Para obter mais informações, consulte the section called “Conexões de conta”.



Propagação de tags para recursos provisionados

Se os recursos provisionados, como aqueles definidos nos modelos de serviço e ambiente, oferecerem suporte à AWS marcação, as tags gerenciadas se propagarão como tags AWS gerenciadas pelo cliente para os recursos provisionados. Essas tags não se propagarão para um recurso provisionado que não ofereça suporte à marcação. AWS

AWS Proton aplica tags aos seus recursos por AWS Proton contas, modelos registrados e ambientes implantados, bem como serviços e instâncias de serviço, conforme descrito na tabela a seguir. Você pode usar tags AWS gerenciadas para visualizar e gerenciar seus AWS Proton recursos, mas não pode modificá-las.

AWS chave de tag gerenciada	Chave propagada gerenciada pelo cliente	Description
`aws:proton:account`	`proton:account`	A AWS conta que cria e implanta AWS Proton recursos.
`aws:proton:template`	`proton:template`	O ARN de um modelo selecionado.
`aws:proton:environment`	`proton:environment`	O ARN de um ambiente selecionado.
`aws:proton:service`	`proton:service`	O ARN de um serviço selecionado.
`aws:proton:service-instance`	`proton:service-instance`	O ARN de uma instância de serviço selecionada.
`aws:proton:component`	`proton:component`	O ARN de um componente selecionado.

Veja a seguir um exemplo de uma tag AWS gerenciada para um AWS Proton recurso.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

Veja a seguir um exemplo de uma tag gerenciada pelo cliente aplicada a um recurso provisionado que foi propagada a partir de uma AWS tag gerenciada.

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

Com o provisionamento AWS gerenciado, AWS Proton aplica tags propagadas diretamente aos recursos provisionados.

Com o provisionamento autogerenciado, AWS Proton disponibiliza tags propagadas junto com os arquivos IaC renderizados que ele envia no pull request (PR) de provisionamento. As tags são fornecidas na variável `proton_tags` do mapa de cadeias de caracteres. Recomendamos que você

faça uma referência a essa variável na configuração do Terraform para incluir AWS Proton tags. `default_tags` Isso propaga tags do AWS Proton para todos os recursos provisionados.

O exemplo a seguir mostra esse método de propagação de tags em um modelo de ambiente do Terraform.

Aqui está a definição da variável `proton_tags`:

proton.environment.variables.tf:

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

Veja como os valores das tags são atribuídos a essa variável:

proton.auto.tfvars.json:

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

E veja como você pode adicionar AWS Proton tags à sua configuração do Terraform para que elas sejam adicionadas aos recursos provisionados:

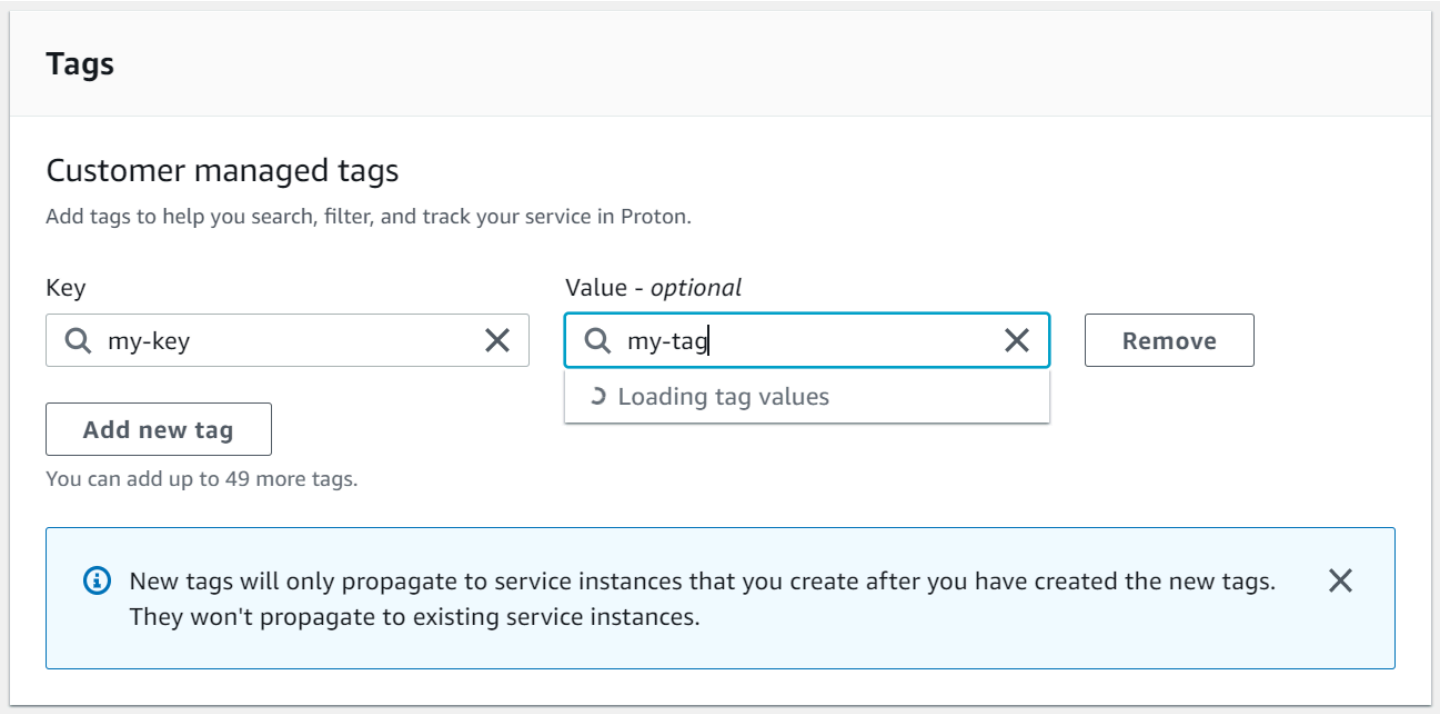
```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Tags gerenciadas pelo cliente

Cada AWS Proton recurso tem uma cota máxima de 50 tags gerenciadas pelo cliente. As tags gerenciadas pelo cliente se propagam para AWS Proton os recursos secundários da mesma forma que as tags AWS gerenciadas, exceto que elas não se propagam para AWS Proton os recursos existentes ou provisionados. Se você aplicar uma nova tag a um AWS Proton recurso com recursos secundários existentes e quiser que os recursos secundários existentes sejam marcados com a nova tag, você precisará marcar cada recurso secundário existente manualmente, usando o console ou AWS CLI.

Crie tags usando o console e a CLI

Ao criar um AWS Proton recurso usando o console, você tem a oportunidade de criar tags gerenciadas pelo cliente na primeira ou na segunda página do procedimento de criação, conforme mostrado no instantâneo do console a seguir. Escolha Adicionar nova tag, insira a chave e o valor e prossiga.



Depois de criar um novo recurso usando o AWS Proton console, você pode ver sua lista de tags AWS gerenciadas e gerenciadas pelo cliente na página de detalhes.

Criar e editar uma tag

1. No AWS Proton console, abra uma página de detalhes do AWS Proton recurso na qual você pode ver uma lista de tags.
2. Selecione Gerenciar tags.
3. Na página Gerenciar tags, você pode visualizar, criar, remover e editar tags. Você não pode modificar as tags AWS gerenciadas listadas na parte superior. No entanto, você pode adicionar e modificar as tags gerenciadas pelo cliente com campos de edição, listados após as tags AWS gerenciadas.

   Escolha Adicionar nova tag para criar uma nova tag.
4. Insira uma chave e um valor para a nova tag.
5. Para editar uma tag, insira um valor no campo de valor da tag para uma chave selecionada.
6. Para excluir uma tag, escolha Remover para uma tag selecionada.
7. Quando você tiver concluído as alterações, selecione Salvar alterações.

Crie tags usando o AWS Proton AWS CLI

Você pode visualizar, criar, remover e editar tags usando AWS Proton AWS CLI o.

É possível criar ou editar uma tag para um recurso, conforme mostrado no exemplo a seguir.

```
$ aws proton tag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tags '[{"key":"mykey1","value":"myval1"},{"key":"mykey2","value":"myval2"}]'
```

Você pode remover uma tag de um recurso, conforme mostrado no próximo exemplo.

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

Você pode listar as tags de um recurso, conforme mostrado no exemplo final.

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

Solução de problemas AWS Proton

Aprenda a solucionar problemas com o. AWS Proton

Tópicos

Erros de implantação que fazem referência a parâmetros CloudFormation dinâmicos

Se você ver erros de implantação que fazem referência às suas variáveis CloudFormation dinâmicas, verifique se elas são de escape do Jinja. Esses erros podem ser causados pela interpretação incorreta do Jinja de suas variáveis dinâmicas. A sintaxe do parâmetro CloudFormation dinâmico é muito semelhante à sintaxe do Jinja que você usa com seus parâmetros. AWS Proton

Exemplo de sintaxe de variável CloudFormation dinâmica:

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

Exemplo de sintaxe do AWS Proton parâmetro Jinja:

`'{{ service_instance.environment.outputs.env-outputs }}'`.

Para evitar esses erros de interpretação errônea, o Jinja escapa de seus parâmetros CloudFormation dinâmicos, conforme mostrado nos exemplos a seguir.

Este exemplo é do Guia CloudFormation do usuário. Os segmentos AWS Secrets Manager secret-name e json-key podem ser usados para recuperar as credenciais de login armazenadas no segredo.

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
```

```
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
```

Para escapar dos parâmetros CloudFormation dinâmicos, você pode usar dois métodos diferentes:

- Coloque um bloco entre `{% raw %}` and `{% endraw %}`:

```
'{% raw %}'
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
'{% endraw %}'
```

- Coloque um parâmetro entre `"{{ }}"`:

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}' }}"
      MasterUserPassword:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}' }}"
```

Para obter informações, consulte Escape do Jinja.

AWS Proton cotas

A tabela a seguir lista as AWS Proton cotas. Todos os valores são por AWS conta, por AWS região suportada.

Cota de recurso	Limite padrão	Ajustável?
Tamanho máximo do pacote de modelos	10 MB	✗ Não
Tamanho máximo do arquivo de manifesto do modelo	2 MB	✗ Não
Tamanho máximo do arquivo de esquema do modelo	2 MB	✗ Não
Tamanho máximo de cada arquivo do modelo	2 MB	✗ Não
Tamanho máximo de cada nome de modelo	100 caracteres	✗ Não
Número máximo de arquivos de CloudFormation modelo por pacote	1	✗ Não
Número máximo de modelos registrados por conta, serviço e modelo de ambiente combinados	1000	✓ Sim
Número máximo de versões do modelo registradas por modelo	1000	✓ Sim
Número máximo de arquivos por pacote de CodeBuild provisionamento	500	✗ Não
Número máximo de ambientes por conta	1000	✓ Sim
Número máximo de serviços por conta	1000	✓ Sim
Número máximo de instâncias de serviço por serviço	20	✓ Sim
Número máximo de componentes por conta	1000	✓ Sim
Número máximo de conexões de conta de ambiente por conta de ambiente	1000	✓ Sim

Histórico do documento

A tabela a seguir descreve as mudanças importantes na documentação relacionadas à versão mais recente do AWS Proton e aos comentários dos clientes. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

- Versão da API: 20-07-2020

Alteração	Descrição	Data
Aviso de fim do suporte	Em 7 de outubro de 2026, AWS encerrará o suporte para AWS Proton.	7 de outubro de 2025
Atualização da política gerenciada	A política de `AWSProtonCodeBuildProvisioningServiceRolePolicy` foi atualizada.	15 de junho de 2024
Atualização da política gerenciada	A política de `AWSProtonDeveloperAccess` foi atualizada.	25 de abril de 2024
Atualização da política gerenciada	A política de `AWSProtonFullAccess` foi atualizada.	25 de abril de 2024
Atualização da política gerenciada	A política de `AWSProtonSyncServiceRolePolicy` foi atualizada.	25 de abril de 2024
Atualização da política gerenciada	A política de `AWSProtonCodeBuildProvisioningServiceRolePolicy` foi atualizada.	12 de maio de 2023

Configurações de sincronização de serviços.	AWS Proton adiciona suporte para configurações de sincronização de serviços.	31 de março de 2023
CodeBuild	AWS Proton adiciona suporte para CodeBuildprovisionamento.	16 de novembro de 2022
Atualização da política gerenciada	Foi adicionada uma `AWSProtonCodeBuildProvisioningBasicAccess` política que fornece CodeBuild as permissões necessárias para executar uma compilação para AWS Proton CodeBuild provisionamento.	11 de novembro de 2022
Propagação de tags Terraform	Foi adicionada a propagação da tag Terraform ao capítulo Marcação.	16 de setembro de 2022
Guia de migração de API	O guia de migração da API pré-GA foi removido.	12 de agosto de 2022
AWS Proton objetos	Foi adicionado um tópico sobre AWS Proton objetos e sua relação com objetos de outros AWS e de terceiros . Veja os objetos do AWS Proton.	29 de julho de 2022
Esclarecimentos sobre o repositório vinculado	Esclareceu a finalidade dos repositórios vinculados (registrados) e seu uso em todo o guia.	18 de julho de 2022

Mesclar guias	Dois guias separados do administrador e do usuário mesclados em um único guia, o Guia AWS Proton do usuário.	30 de junho de 2022
Atualização da política gerenciada	Políticas gerenciadas atualizadas para fornecer acesso a novas operações de AWS Proton API e corrigir problemas de permissão para algumas operações AWS Proton do console. Consulte as políticas AWS gerenciadas para AWS Proton.	20 de junho de 2022
Conceitos básicos de CLI	Introdução ao AWS CLI atualizado com um novo tutorial que usa a nova biblioteca de modelos.	14 de junho de 2022
Componentes diretamente definidos	O capítulo Componentes foi adicionado e as modificações relacionadas foram feitas em todo o guia.	1º de junho de 2022
AWS Proton biblioteca de modelos	Adicionado o tópico da biblioteca de AWS Proton modelos.	6 de maio de 2022
Disponibilidade geral do Terraform	Provisionamento de pull request renomeado para provisionamento autogeren ciado. Tópico adicionado sobre métodos de provision amento.	23 de março de 2022

Marcação de repositório	Suporte adicionado para marcar recursos de repositór io. Consulte Criar um link para seu repositório.	23 de março de 2022
Atualização da documentação	Foi adicionada a marcação de conexão da conta de ambiente.	26 de novembro de 2021
Sincronizações de modelos e visualização prévia do Terraform	Foi adicionado o controle de versionamento automatizado do modelo com o atributo de sincronização de modelos para disponibilidade geral e provisionamento de pull request com o Terraform na prévia. Guia de migração de API de volta.	24 de novembro de 2021
Atualizações feitas na documentação	EventBridgeTutorial adicionad o, fluxo de trabalho de introdução, Como AWS Proton funciona e aprimoramentos na seção Pacote de modelos.	17 de setembro de 2021
AWS Proton lançamento de painéis de ajuda do console	Painéis de ajuda adicionad os ao console. A exclusão da versão do modelo do console não exclui mais as versões anteriores. O guia de migração da API foi removido.	8 de setembro de 2021

AWS Proton versão de disponibilidade geral (GA)	Foram adicionados ambientes de várias contas, EventBridge monitoramento, chaves de condição do IAM, suporte à idempotência e aumento de cotas.	9 de junho de 2021
Adicione e exclua instâncias de serviço para um serviço e use a infraestrutura externa existente para ambientes com AWS Proton	Essa versão prévia pública inclui atualizações que possibilitam que você adicione e exclua instâncias de serviço de um serviço, use sua infraestrutura externa existente em um AWS Proton ambiente e cancele implantações de ambiente, instância de serviço e pipeline. AWS Proton agora suporta PrivateLink. Uma validação de exclusão adicional foi adicionada para evitar que uma versão secundária seja excluída por engano enquanto um recurso a estiver usando.	27 de abril de 2021
Marcação com AWS Proton	A versão prévia pública 2 inclui AWS Proton marcação e a capacidade de iniciar serviços sem um pipeline de serviços.	5 de março de 2021
Lançamento inicial	A versão prévia pública agora está disponível em regiões selecionadas.	1º de dezembro de 2020

AWS Glossário

Para obter a AWS terminologia mais recente, consulte o AWS glossário na Glossário da AWS Referência.