AWS Proton

Administrator Guide

AWS Proton: Administrator Guide

Table of Contents

What is AWS Proton?

AWS Proton is:

- **Automated infrastructure provisioning and deployment of serverless and container-based applications**

  The AWS Proton service is a two-pronged automation framework. As an administrator, you can create *versioned service templates* that define standardized infrastructure and deployment tooling for serverless and container-based applications. Then, developers can select from the available *service templates* to automate their application or service deployments.

  AWS Proton identifies all existing *service instances* that are using an outdated template version for you. As an administrator, you can request AWS Proton to upgrade them with one click.
- **Standardized infrastructure**

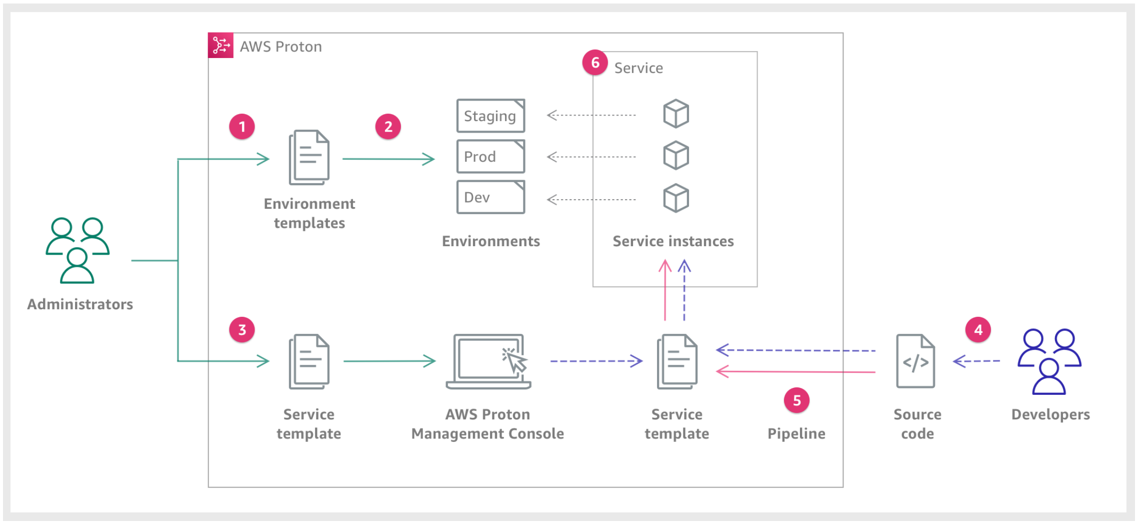
  Platform teams can use AWS Proton and versioned infrastructure as code templates to define and manage standard application stacks that contain the architecture, infrastructure resources, and the CI/CD software deployment pipeline.
- **Deployments integrated with CI/CD**

  When developers use the AWS Proton self-service interface to select a *service template*, they're selecting a standardized application stack definition for their code deployments. AWS Proton automatically provisions the resources, configures the CI/CD pipeline, and deploys the code into the defined infrastructure.

AWS Proton for platform teams

As an administrator, you or members of your platform team, create *environment templates* and *service templates*. The *environment template* defines shared infrastructure used by multiple applications or resources. The *service template* defines the type of infrastructure that's needed to deploy and maintain a single application or microservice in an *environment*. An AWS Proton *service* is an instantiation of a *service template*, which normally includes several *service instances* and a *pipeline*. An AWS Proton *service instance* is an instantiation of a *service template* in a specific *environment*. You or others in your team can specify which *environment templates* are compatible with a given *service template*. For more information about *templates*, see AWS Proton templates (p. 63).

The following diagram is a visualization of the main AWS Proton concepts discussed in the preceding paragraph. It also offers a high-level overview of what constitutes a simple AWS Proton workflow.



1 Identify input parameters (p. 15).

2 Create a schema file (p. 56) that defines your input parameters.

3 As an **Administrator**, you create and register a **Service Template** with AWS Proton, which defines the related infrastructure, monitoring, and CI/CD resources as well as compatible **Environment Templates**.

4 As a **Developer**, you select a registered **Service Template** and provide a link to your **Source code** repository.

5 AWS Proton provisions the **Service** with a **CI/CD Pipeline** for your **Service instances**.

6 AWS Proton provisions and manages the **Service** and the **Service Instances** that are running the **Source code** as was defined in the selected **Service Template**. A **Service Instance** is an instantiation of the selected **Service Template** in an **Environment** for a single stage of a **Pipeline** (for example Prod).

Setting up

Complete the tasks in this section so that you can create and register service and environment templates. You need these to deploy environments and services with AWS Proton.

Note
We're offering the use of AWS Proton at no additional expense. You can create, register, and maintain service and environment templates at no charge. You can also count on AWS Proton to self-manage its own operations, such as storage, security and deployment, at no cost to you. The only expenses that you incur while using AWS Proton are the following.

- Costs of deploying and using AWS cloud resources that you instructed AWS Proton to deploy and maintain for you.
- Costs of maintaining an AWS connection to your code repository.
- Costs of maintaining an Amazon S3 bucket that you need to provide inputs to AWS Proton.

Topics

Setting up with IAM

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including AWS Proton. You're charged only for the services and resources that you use.

Note
You and your team, including administrators and developers, must all be under the same account.

Sign Up for AWS

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Create an IAM User

To create an administrator user for yourself and add the user to an administrators group (console)

1. Sign in to the IAM console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

> **Note**
> We strongly recommend that you adhere to the best practice of using the **`Administrator`** IAM user that follows and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **`Administrator`**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **`Administrators`**.
10. Choose **Filter policies**, and then select **AWS managed - job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

    > **Note**
    > You must activate IAM user and role access to Billing before you can use the `AdministratorAccess` permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in step 1 of the tutorial about delegating access to the billing console.

12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM entities in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see Access management and Example policies.

Setting up AWS Proton service roles

Create an AWS Proton service role to allow AWS Proton to make API calls to other AWS services, like CloudFormation, on your behalf. To learn more about the AWS Proton service role, see AWS Proton service role (p. 135).

Create an AWS Proton pipeline service role to allow AWS Proton to make API calls to other services, like CloudFormation, on your behalf. To learn more about the AWS Proton service pipeline role, see AWS Proton pipeline service role (p. 137).

You can create the IAM service roles that AWS Proton needs to make API calls on your behalf by using the console as shown in the following steps.

> **Note**
> Because we don't know which resources you will define in your AWS Proton templates, the role that you create using the console has broad permissions and can be used to act as both the AWS Proton pipeline service role and the AWS Proton service role. For production deployments, we recommend that you scope down the permissions to the specific resources that will be deployed

by creating customized policies for both the AWS Proton pipeline service role and the AWS Proton service role. You can create and customize these roles by using the AWS CLI or IAM. For more information, see Service roles for AWS Proton (p. 131) and Create a service (p. 97).

To set up IAM AWS Proton service roles using the console.

1. Open the AWS Proton console.
2. In the navigation pane, select **Settings** and then **Account roles**.
3. In the **Account roles** page, select **Configure**.
4. In the **Configure CI/CD pipeline role** page, select **New service role**.
5. Enter the name of the role as **`myProtonPipelineServiceRole`**.
6. Check the check box to agree to create an AWS Proton role with administrative privileges in your account.
7. Choose **Save changes**.

   Your new pipeline service role is displayed on the **Account roles** page.

Setting up with AWS Proton

You need to set up a repository connection and Amazon S3 bucket. Verify that you have installed the AWS CLI if you plan to run AWS Proton commands using it.

Setting up an Amazon S3 bucket

Follow the instructions at https://docs.aws.amazon.com/AmazonS3/latest/gsg/CreatingABucket.html to set up an S3 bucket. You'll place your inputs to AWS Proton in the bucket where AWS Proton can retrieve them. These inputs are known as template bundles. You can learn more about them in other sections of this guide.

Setting up a repository connection

Connect AWS Proton to your code repository with AWS CodeStar connections to prompt the AWS Proton service pipeline when a new push is made to your repository. You can connect to Bitbucket, GitHub, GitHub Enterprise and GitHub Enterprise Server repositories with AWS CodeStar connections. For more information, see AWS CodeStar connections in the *AWS CodePipeline User Guide*.

To set up a repository connection.

1. In the AWS Proton console.
2. In the navigation pane, select **Settings** and then **Source connections** to take you to the **Connections** page in **Developer Tools Settings**. The page displays a list of repository connections.
3. Choose **Create connection** and follow the instructions.

Setting up with the AWS CLI

To use the AWS CLI to make AWS Proton API calls, verify that you have installed the latest version of the AWS CLI. For more information, see https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

Getting started with AWS Proton

Before getting started, get set up (p. 3) to use AWS Proton and verify you have met the Getting started preqrequisites (p. 6).

Get started with AWS Proton by choosing one or more of the following paths:

- Follow a guided example console or CLI workflow (p. 6) through documention links.
- Run through a guided example console workflow (p. 8).
- Run through a variety of AWS Proton CLI examples (p. 11) that are posted on GitHub.

Topics

Prerequisites

Before you start step 1 in Getting started with the AWS Management Console (p. 8), make sure that the following prerequisites are met. For more information, see Setting up (p. 3).

- You have an IAM account with administrator permissions. For more information, see Setting up with IAM (p. 3).
- You have the AWS Proton service role and the AWS Proton pipeline service role are attached to your account. For more information, see Setting up AWS Proton service roles (p. 4) and Service roles for AWS Proton (p. 131).
- You have a version control repository connection. For more information, see Setting up a repository connection (p. 5).
- You're familiar with creating AWS CloudFormation templates and Jinja parametrization. For more information, see What is AWS CloudFormation? *in the AWS CloudFormation User Guide* and Jinja website.
- You have working knowledge of AWS infrastructure services.
- You're logged into your AWS account.

Getting started workflow

Learn to create template bundles, create and register templates, and create environments and services by following the example steps and links.

Before starting, verify that you created an AWS Proton service role (p. 4).

If your service template includes an AWS Proton service pipeline, verify that you created a repository connection (p. 5) and a AWS Proton pipeline service role (p. 4).

For more information, see The AWS Proton service API Reference.

Example: Getting started workflow

1. Refer to the diagram in How AWS Proton works (p. 12) for a high-level view of AWS Proton inputs and outputs.
2. Create an environment bundle and a service template bundle (p. 14).

   a. Identify input parameters (p. 15).
   b. Create a schema file (p. 56).
   c. Create infrastructure as code (IaC) files (p. 21).
   d. To wrap up your template bundle (p. 61), create a manifest file and organize your IaC files, manifest files, and schema file in directories.
   e. Make your template bundle (p. 61) accessible to AWS Proton.
3. Create and register an environment template version (p. 64) with AWS Proton.

   When you use the console to create and register a template, a template version is automatically created.

   When you use the AWS CLI to create and register a template:

   a. Create an environment template.
   b. Create an environment template version.

   For more information, see CreateEnvironmentTemplate and CreateEnvironmentTemplateVersion in the *AWS Proton API reference*.
4. Publish your environment template (p. 75) to make it available for use.

   For more information, see UpdateEnvironmentTemplateVersion in the *AWS Proton API reference*.
5. To create an environment (p. 80), select a published environment template version and provide values for required inputs.

   For more information, see CreateEnvironment in the *AWS Proton API reference*.
6. Create and register a service template version (p. 64) with AWS Proton.

   When you use the console to create and register a template, a template version is automatically created.

   When you use the AWS CLI to create and register a template:

   a. Create a service template.
   b. Create a service template version.

   For more information, see CreateServiceTemplate and CreateServiceTemplateVersion in the *AWS Proton API reference*.
7. Publish your service template (p. 75) to make it available for use.

   For more information, see UpdateServiceTemplateVersion in the *AWS Proton API reference*.
8. To create a service (p. 97), select a published service template version and provide values for required inputs.

   For more information, see CreateService in the *AWS Proton API reference*.

Getting started with the AWS Management Console

Get started with AWS Proton

- Create and view an environment template.
- Create, view, and publish a service template that uses the environment template that you just created.
- Create an environment and service (optional).
- Delete the service template, environment template, environment and service, if created.

Step 1: Open the AWS Proton console

- Open the AWS Proton console.

Step 2: Create an environment template

In the navigation pane, choose **Environment templates**.

1. In the **Environment templates** page, choose **Create Environment template**.
2. In the **Create environment template** page, in the **Template options** section, choose **Create a template for provisioning new environments**.
3. In the **Template bundle source** section, choose **Use one of our sample template bundles**.
4. In the **Sample template bundle** section, select **fargate-environment**.
5. In the **Template details** section.

   a. Enter the template name as **`my-env-template`**.

   b. Enter the environment template display name as **`My Fargate Environment`**.

   c. (Optional) Enter a description for the environment template.
6. (Optional) In the **Tags** section, choose **Add new tag** and enter a key and value to create a customer managed tag.
7. Choose **Create Environment template**.

   You're now on a new page that displays the status and details for your new environment template. These details include a list of AWS and customer managed tags. AWS Proton automatically generates AWS managed tags for you when you create AWS Proton resources. For more information, see AWS Proton resources and tagging (p. 150).
8. Refresh the **Template versions** section and the status of the new version is *Draft*. The status of a new environment template status starts in the **Draft** state. You and others with `proton:CreateEnvironment` permissions can view and access it. Follow the next step to make the template available to others.
9. In the **Template versions** section, choose the radio button to the left of the minor version of the template you just created (1.0). As an alternative, you can choose **Publish** in the info alert banner and skip the next step.
10. In the **Template versions** section, choose **Publish**.
11. The template status changes to **Published**. Because it's the latest version of the template, it's the **Recommended** version.
12. In the navigation pane, select **Environment templates**.

A new page displays a list of your environment templates along with template details.

Step 3: Create a service template

Create a service template.

1. In the navigation pane, choose **Service templates**.
2. In the **Service templates** page, choose **Create Service template**.
3. In the **Create service template** page, in the **Template bundle source** section, choose **Use one of our sample template bundles**.
4. In the **Sample template bundle** section, select **fargate-service**.
5. In the **Template details** section.

   a. Enter the service template name as `my-svc-template`.
   b. Enter the service template display name as `My Fargate Service`.
   c. (Optional) Enter a description for the service template.
6. In the **Compatible environment templates** section.

   - Check the check-box to the left of the environment template **My Fargate Environment** to select the compatible environment template for the new service template.
7. For **Encryption settings**, keep the defaults.
8. In the **Pipeline - optional** section.

   - Keep the checkbox checked to include a service pipeline in your service template.
9. Choose **Create service template**.

   You're now on a new page that displays the status and details for your new service template, including a list of AWS and customer managed tags.
10. Refresh the **Template versions** section and the status of the new version is *Draft*. The status of a new service template status starts in the **Draft** state. Only administrators can view and access it. To make the service template available for use by developers, follow the next step.
11. In the **Template versions** section, choose the radio button to the left of the minor version of the template you just created (1.0). As an alternative, you can choose **Publish** in the info alert banner and skip the next step.
12. In the **Template versions** section, choose **Publish**.
13. The template status changes to **Published**.

    The first minor version of your service template is published and available for use by developers. Because it's the latest version of the template, it's the **Recommended** version.
14. In the navigation pane, choose **Service templates**.

    A new page displays a list of your service templates and details.

Step 4: Create an environment

In the navigation pane, choose **Environments**.

1. Choose **Create environment**.
2. In the **Choose an environment template** page, select the template that you just created. It's named **My Fargate Environment**. Then, choose **Configure**.

3. In the **Configure environment** page, in the **Deployment account** section, select **This AWS account**.
4. Enter the environment name as **`my-fargate-environment`**.
5. In the **Environment roles** section, select **New service role** or, if you have already created an AWS Proton service role, select **Existing service role**.
    a. Select **New service role** to create a new role.
        i. Enter the **Environment role name** as **`MyProtonServiceRole`**.
        ii. Check the checkbox to agree to create an AWS Proton service role with administrative privileges for your account.
    b. Select **Existing service role** to use an existing role.
        - Select your role in the **Environment role name** drop down field.
6. Choose **Next**.
7. On the **Configure custom settings** page, use the defaults.
8. Choose **Next** and review your inputs.
9. Choose **Create**.

    View the environment details and status, as well as the AWS managed tags and customer managed tags for your environment.
10. In the navigation pane, choose **Environments**.

    A new page displays a list of your environments along with the status and other environment details.

Step 5: Optional - Create a service and deploy an application

- After the environment is created and **Deployment status** is *Succeeded*, launch a service based on your service template and environment. To do this, follow the instructions at Prerequisites and Create a service in the *AWS Proton User Guide*.

Step 6: Clean up.

1. **Delete a service (if you created one)**

    To delete the service, follow the instructions at Delete a service in the *AWS Proton User Guide*.
2. Open the AWS Proton console.
3. **Delete an environment**
    a. In the navigation pane, choose **Environments**.
    b. In the **Environments** page, select the radio button the left of the environment that you just created.
    c. Choose **Actions**, then **Delete**.
    d. A modal prompts you to confirm the delete action.
    e. Follow the instructions and choose **Yes, delete**.
4. **Delete a service template**
    a. In the navigation pane, choose **Service templates**.

   b. In the **Service templates** page, select the radio button to the left of service template **my-svc-template**.
   c. Choose **Actions**, then **Delete**.
   d. A modal prompts you to confirm the delete action.
   e. Follow the instructions and choose **Yes, delete**. This deletes the service template and all of its versions.

5. **Delete an environment template**

   a. In the navigation pane, choose **Environment templates**.
   b. In the **Environment templates** page, select the radio button to the left of **my-env-template**.
   c. Choose **Actions**, then **Delete**.
   d. A modal prompts you to confirm the delete action.
   e. Follow the instructions and choose **Yes, delete**. This deletes the environment template and all of its versions.

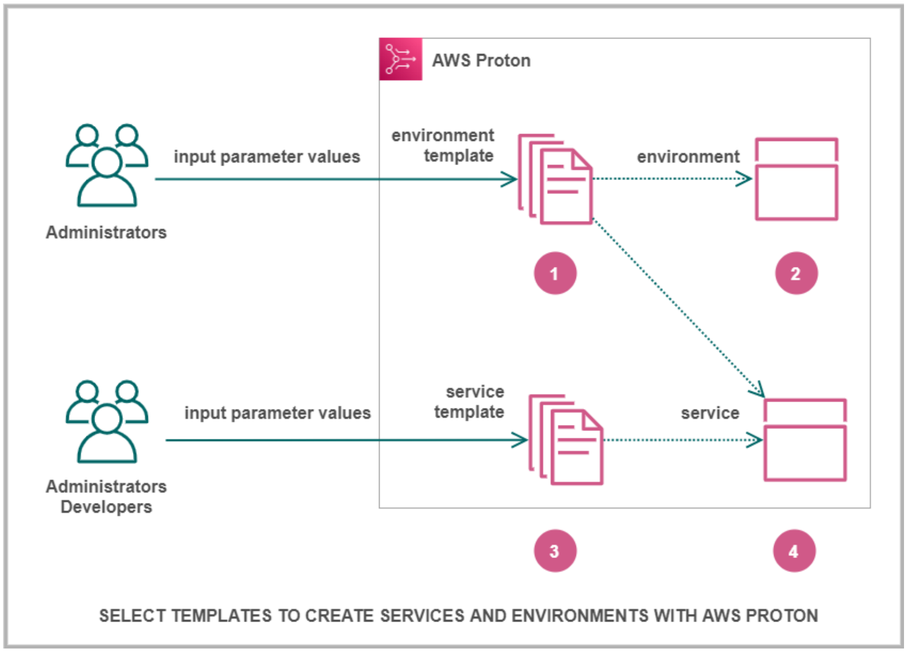
Getting started with the AWS Proton AWS CLI

To get started with the AWS CLI, walk thru the following AWS CLI examples on GitHub.

- Multiple services share an AWS Proton environment
- Load-balanced web service using Amazon ECS and AWS Fargate
- Microservices using Amazon ECS and AWS Fargate

How AWS Proton works

AWS Proton is configured to provision *environments* or *services* when you select *environment or service templates* in your AWS Proton versioned template library.



1 When you, as an administrator, select an environment template with AWS Proton, you provide values for required *input parameters*.

2 AWS Proton uses the environment template and parameter values to provision your environment.

3 When you, as a developer or administrator, select a service template with AWS Proton, you provide values for required input parameters. You also select an environment to deploy your application or service to.

4 AWS Proton uses the service template, and both your service and selected environment parameter values, to provision your service.

You provide values for the input parameters to customize your template for re-use and multiple use cases, applications, or services.

To make this work, you create environment or service template bundles and upload them to registered environment or service templates, respectively.

Template bundles (p. 14) contain everything AWS Proton needs to provision environments or services.

When you create an environment or service template, you upload a template bundle that contains the parametrized infrastructure as code (IaC) files that AWS Proton uses to provision environments or services.

When you select an environment or service template to create or update an environment or service, you provide values for the template bundle IaC file parameters.

AWS Proton terminology

Environment template

Defines shared infrastructure, such as a VPC or cluster, that is used by multiple applications or resources.

Environment template bundle

1. A schema file that defines infrastructure as code input parameters.
2. An infrastructure as code (IaC) file that defines shared infrastructure, such as a VPC or cluster, that is used by multiple applications or resources.
3. A manifest file that lists the IaC file.

Environment

Provisioned shared infrastructure, such as a VPC or cluster, that is used by multiple applications or resources.

Service template

Defines the type of infrastructure that's needed to deploy and maintain an application or microservice in an environment.

Service template bundle

1. A schema file that defines infrastructure as code (IaC) input parameters.
2. An IaC file that defines the infrastructure that's needed to deploy and maintain an application or microservice in an environment.
3. A manifest file that lists the IaC file.
4. Optional
   a. An IaC file that defines the service pipeline infrastructure.
   b. A manifest file that lists the IaC file.

Service

Provisioned infrastructure that's needed to deploy and maintain an application or microservice in an environment.

Service instance

Provisioned infrastructure that supports an application or microservice in an environment.

Service pipeline

Provisioned infrastructure that supports a pipeline.

Template version

A major or minor version of a template. For more information, see Versioned templates (p. 64).

Schema file

Defines infrastructure as code file input parameters.

Manifest file

Lists an infrastructure as code file.

Input parameters

Defined in a schema file and used in an infrastructure as code (IaC) file so that the IaC file can be used repeatably and for a variety of use cases.

Template bundles

As an administrator, you create and register templates (p. 64) with AWS Proton. You use these templates to create environments and services. When you create a service, AWS Proton provisions and deploys service instances to selected environments. For more information, see AWS Proton for platform teams (p. 1).

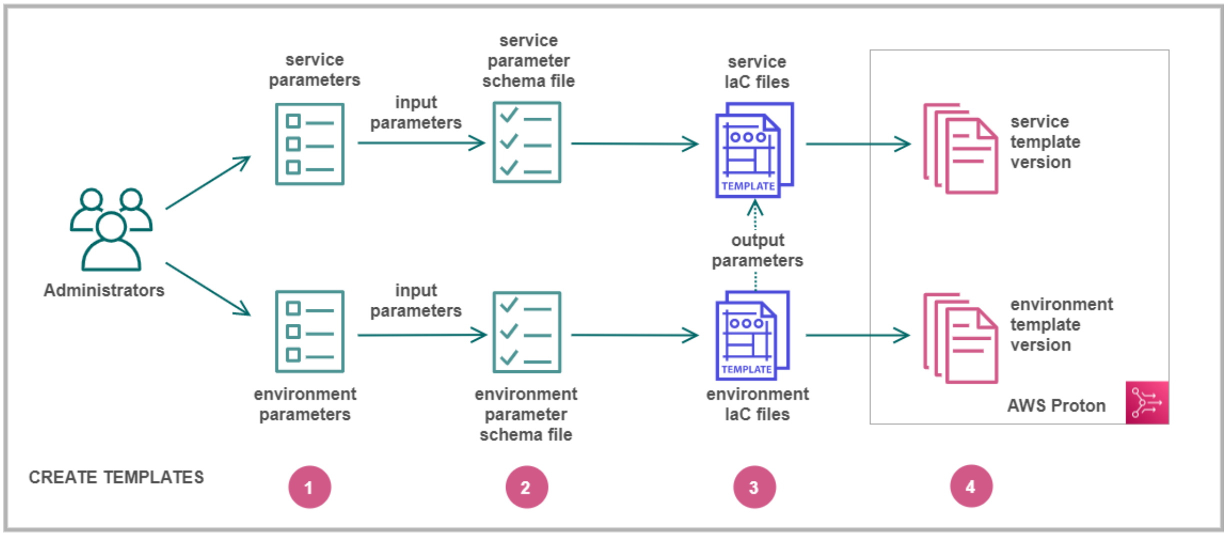
To create and register a template in AWS Proton, you upload a template bundle that contains the infrastructure as code (IaC) files that AWS Proton needs to provision and environment or service.

A *template bundle* contains the following:

- Infrastructure as code (IaC) YAML files (p. 21) with a manifest YAML file (p. 61) that lists the *IaC files*.
- A schema YAML file (p. 56) for your IaC file input parameter definitions.

AWS Proton requires an input parameter schema file. When you use AWS CloudFormation to create your IaC files, you use Jinja syntax to reference your input parameters. AWS Proton provides parameter namespaces that you can use to reference parameters (p. 15) in your IaC files.

The following diagram shows an example of steps that you can take to create a *template* for AWS Proton.



1 Identify input parameters (p. 15).

2 Create a schema file (p. 56) to define your input parameters.

3 Create IaC files (p. 21) that reference your input parameters. You can reference environment IaC file *outputs* as *inputs* for your service IaC files.

4 Register a template version (p. 64) with AWS Proton and upload your template bundle.

Topics

- Schema file (p. 56)
- Wrap up template bundles for AWS Proton (p. 61)
- Template bundle considerations (p. 63)
- AWS Proton templates (p. 63)

Parameters

To work with different types of infrastructure as code (IaC) files, AWS Proton uses parameter *namespaces* and a schema file (p. 56) for handling parameters. AWS Proton checks input parameter values against your schema file. It matches the input parameter values with the parameters that are referenced in your IaC files to inject the input values. AWS Proton uses Jinja syntax with AWS CloudFormation IaC files.

AWS Proton automatically creates AWS Proton resource parameters. You can reference these resource parameters in all of your template bundle IaC files. An example of a resource parameter is `environment.name`.

You can use parameters in your environment and service IaC files with the following requirements:

- The length of each parameter name can't exceed 100 characters.
- The length of the parameter namespace and *resource name* combined can't exceed the character limit for the resource name.

AWS Proton provisioning fails if these quotas are exceeded.

To reference an input parameter in an IaC file, you can attach an AWS Proton parameter namespace. For AWS CloudFormation IaC files, you can use *Jinja* syntax and surround the parameter with pairs of moustache brackets and quotation marks.

For an input parameter defined as `"VPC"` in your schema, you can reference your input parameter as `"{{ environment.input.VPC }}"` in your CloudFormation environment IaC file.

The following table lists AWS Proton resource parameters and the namespaces that you can use in your IaC files.

AWS Proton parameters and namespaces

IaC file	Resource parameter	Parameter namespace	Jinja syntax for CloudFormation	Description
Environment `IaC.yaml`		environment.	{{ environment.*name* }}	Environment name
		environment.inputs.	{{ environment.inputs.`env-input` }}	Schema defined input
Instance `IaC.yaml`		environment.	{{ environment.*name* }}	Environment name
		environment.outputs.	{{ environment.outputs.`env-outputs` }}	Input from from an environment IaC file output

IaC file	Resource parameter	Parameter namespace	Jinja syntax for CloudFormation	Description

		service.	{{ service.*branch_name* }}
			
			{{ service.*name* }}
			
			{{ service.*repository_connection_arn* }}
			
			{{ service.*repository_id* }}
				Service name and code repository data

		service_instance.inputs.	{{ service_instance.inputs.`svc-instance-input` }}	Schema defined service instance input
		service_instance.	{{ service_instance.*name* }}	Service instance name
Pipeline `IaC.yaml`		service_instance.environment.outputs.	{{ service_instance.environment.outputs.`env-outputs` }}	Input from from an environment IaC file output
		pipeline.inputs.	{{ pipeline.inputs.`pipeline-input` }}	Schema defined pipeline input

		service.	{{ service.*branch_name* }}
			
			{{ service.*name* }}
			
			{{ service.*repository_connection_arn* }}
			
			{{ service.*repository_id* }}
				Service name and code repository data

	service_instances		{% for service_instance in service_instances %}...{% endfor %}	Array of service instances that you can loop through

For more information and examples, see the following parameter sections for environment and service infrastructure template files.

Environment infrastructure as code file parameter details and examples

You can use the following types of parameters in your environment infrastructure as code (IaC) files.

- **Input parameters**

These are the parameters that you add to your IaC files to make them flexible and re-usable. You can define them in your schema file (p. 56). In your environment IaC file, you can attach a namespace to an input parameter to associate it with an AWS Proton resource. Input parameter values can only be provided when you create the environment. The following list includes examples of input parameters for typical use cases.

- VPC CIDR values
- Load balancer settings
- Database settings
- A health check timeout

As an administrator, you can provide values for input parameters when you create an environment (p. 80):

- Use the console to fill out a schema-based form that AWS Proton provides.
- Use the CLI to provide spec that includes the values.

To reference an input parameter in an IaC file, you attach a namespace to it.

In the following CloudFormation environment IaC file example, the `environment.inputs.` namespace identifies input parameters.

```
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 parameters
Outputs:
  MyEnvParameterValue:                                        # output
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                         # output
    Value: "{{ environment.inputs.my_sample_input }}"         # input parameter
  MyOtherSampleInputValue:                                    # output
    Value: "{{ environment.inputs.my_other_sample_input }}"   # input parameter
  AnotherOptionalInputValue:                                  # output
    Value: "{{ environment.inputs.another_optional_input }}"  # input parameter
```

- **AWS Proton resource parameters**

  AWS Proton automatically creates resource parameters, such as an environment `name`. You can reference these resource parameters in any of the IaC files that are used for provisioning an environment and the services deployed to it.

  To refer to an environment `name`, you can use the `environment.` namespace, for example, `environment.name`.

- **Output parameters**

  These are parameters that reference outputs from an environment IaC file.

  To reference an output parameter in a IaC file, you can attach a namespace to it.

The following example is a snippet from an environment CloudFormation IaC file. It lists CloudFormation output parameters and their corresponding values. These are available to service IaC files with the `environment.outputs.` namespace. For example, the namespace `environment.outputs.ClusterName` is used to designate the value for the output for the parameter `ClusterName`.

```
# 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:
  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'
```

The following example is a snippet from a service CloudFormation IaC file. The `environment.outputs.` namespace identifies environment outputs from an environment IaC file.

```
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 from 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 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}}' # output from 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'
[...]
```

Service infrastructure as code file parameter details and examples

You can use the following types of parameters in your service and pipeline infrastructure as code (IaC) files.

- **Input parameters**

These are the parameters that you add to your IaC files to make them flexible and re-usable. You can define them in your schema file (p. 56). In your service IaC file, you can attach a namespace to an input parameter to associate it with an AWS Proton resource. Input parameter values can only be provided when you create the service. The following list includes examples of input parameters for typical use cases.

- Port
- Task size
- Image
- Desired count
- Docker file
- Unit test command

As an administrator, you can provide values for input parameters when you create a service (p. 97):

- Use the console to fill out a schema-based form that AWS Proton provides.
- Use the CLI to provide spec that includes the values.

To reference an input parameter in an IaC file, you can attach a namespace to it.

- **AWS Proton resource parameters**

  AWS Proton automatically creates resource parameters, such as a service `name` that you can reference in all IaC files used for provisioning a service.

  To refer to a service or service instance `name`, you can attach the `service.` or `service_instance.` namespace, for example, `service.name` or `service_instance.name`.

- **Output parameters**

  These are parameters that reference outputs from an environment IaC file.

  To reference an output in a service IaC file, you can attach a namespace to it, for example, `environment.outputs.MySampleInputValue`.

The following example is a snippet from a service CloudFormation IaC file. The `environment.outputs.` namespace identifies outputs from the environment IaC file. The `service_instance.inputs.` namespace identifies input parameters. The `service_instance.name` namespace and property identifies an AWS Proton resource parameter.

```
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 parameters                          # input parameters

                        # outputs from an environment infrastructure as code file

Outputs:
  MyServiceInstanceParameter:                                                    #
 output
    Value: !Ref StoreServiceInstanceInputValue
  MyServiceInstanceRequiredInputValue:                                           #
 output
```

```
   Value: "{{ service_instance.inputs.my_sample_service_instance_required_input }}"  #
input parameter
 MyServiceInstanceOptionalInputValue:                                                 #
output
   Value: "{{ service_instance.inputs.my_sample_service_instance_optional_input }}"  #
input parameter
 MyServiceInstancesEnvironmentSampleOutputValue:                                      #
output
   Value: "{{ environment.outputs.MySampleInputValue }}"                              #
output from an environment infrastructure as code file
 MyServiceInstancesEnvironmentOtherSampleOutputValue:                                 #
output
   Value: "{{ environment.outputs.MyOtherSampleInputValue }}"                         #
output from an environment infrastructure as code file
```

Infrastructure as code files

The primary components of the template bundle are *infrastructure as code (IaC) YAML files* that define the infrastructure resources and properties that you want to provision. AWS CloudFormation and other infrastructure as code engines use these types of files to provision infrastructure resources.

AWS Proton currently supports CloudFormation IaC files. To learn more about CloudFormation, see What is AWS CloudFormation in *the AWS CloudFormation User Guide*.

Start with your own existing infrastructure as code files

You can build template bundles by adapting *your own existing* infrastructure as code (IaC) files for use with AWS Proton.

The following AWS CloudFormation examples, Example 1 (p. 21), and Example 2 (p. 24), represent *your own existing* CloudFormation IaC files. CloudFormation can use these files to create two different CloudFormation stacks.

In Example 1 (p. 21), the CloudFormation IaC file is configured to provision infrastructure to be shared among container applications. In this example, input parameters are added so that you can use the same IaC file to create multiple sets of provisioned infrastructure. Each set can have different names along with a different set of VPC and subnet CIDR values. As either administrator or a developer, you provide values for these parameters when you use an IaC file to provision infrastructure resources with CloudFormation. For your convenience, these input parameters are marked with comments and referenced multiple times in the example. The *outputs* are defined at the end of the template. They can be referenced in other CloudFormation IaC files.

In Example 2 (p. 24), the CloudFormation IaC file is configured to deploy an application to the infrastructure that's provisioned from *Example 1*. The parameters are commented for your convenience.

Example 1: CloudFormation IaC file

```
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"
```

Example 2: CloudFormation IaC file

```
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 'ContainerPortInput'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "Low CPU utilization for service"
            - !Ref 'ContainerPortInput'
      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}"
```

You can adapt these files for use with AWS Proton.

Bring your infrastructure as code to AWS Proton

With slight modifications, you can use Example 1 (p. 21) as an infrastructure as code (IaC) file for an environment template bundle that AWS Proton uses to deploy an environment as shown in Example 3 (p. 30).

Instead of using the CloudFormation parameters, you use Jinja syntax to reference parameters that you have defined in an Open API based schema file (p. 56). These input parameters are commented for your convenience and referenced multiple times in the IaC file. This way, AWS Proton can audit and check parameter values, as well as match and insert output parameter values in one IaC file to parameters in another IaC file.

As administrator, you can add the AWS Proton `environment.inputs.` namespace to the input parameters. When you reference environment IaC file outputs in a service IaC file, you can add the `environment.outputs.` namespace to the outputs, such as `environment.outputs.ClusterName` and `environment.outputs.ECSTaskExecutionRole`. Finally, you surround them with moustache brackets and quotation marks.

With these modifications, your CloudFormation IaC files can be used by AWS Proton.

Example 3: AWS Proton environment infrastructure as code file

```
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'

<<<<<<< HEAD
# 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.
=======
# These output values are available to service infrastructure files as resource parameters,
 when given the
# the 'environment.outputs' namespace, for example, environment.outputs.ClusterName.
# In a service infrastructure file, resource parameters refer to resources in a related
 infrastructure template file.
>>>>>>> help
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'
```

The IaC files shown in Example 1 (p. 21) and Example 3 (p. 30) produce slightly different CloudFormation stacks in the way that parameters are displayed when using the CloudFormation console to view stack templates. The *Example 1* CloudFormation stack template file displays the parameter labels (keys) in the stack template view while the *Example 3* AWS Proton CloudFormation infrastructure stack template file displays the parameter values. AWS Proton input parameters *don't* appear in the CloudFormation stack parameters view.

In Example 4 (p. 32), the AWS Proton service IaC file corresponds with Example 2 (p. 24).

Example 4: AWS Proton service instance IaC file

```
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}"
```

In Example 5 (p. 37), the AWS Proton pipeline IaC file provisions the pipeline infrastructure to support the service instances provisioned by Example 4 (p. 32).

Example 5: AWS Proton service pipeline IaC file

```
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
            Effect: Allow
            Resource:
              - "arn:aws:s3:::aws-proton-preview-public-files/*"
          - 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
          Effect: Allow
          Resource:
            - "arn:aws:s3:::aws-proton-preview-public-files/*"
        - 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}"
```

Schema file

As an administrator, when you use the Open API Data Models (schemas) section to define a parameter schema YAML file for your template bundle, AWS Proton is able to validate parameter value inputs against the requirements that you defined in your schema.

For more information about formats and available keywords, see the Schema object section of the OpenAPI.

Schema requirements for environment template bundles

Your schema must follow the Data Models (schemas) section of the OpenAPI in the YAML format. It must also be a part of your environment template bundle.

For your environment schema, you must include the formatted headers to establish that you're using the Data Models (schemas) section of the Open API. In the following environment schema examples, these headers appear in the first three lines.

An `environment_input_type` must be included and defined with a name that you provide. In the following examples, this is defined on line 5. By defining this parameter, you associate it with an AWS Proton environment resource.

To follow the Open API schema model, you must include `types`. In the following example, this is line 6.

Following `types`, you must define an `environment_input_type` type. You define the input parameters for your environment as properties of the `environment_input_type`. You must include at least one property with a name that matches at least one parameter that's listed in the environment infrastructure as code (IaC) file that's associated with schema.

When you create an environment and provide customized parameter values, AWS Proton uses the schema file to match, validate, and inject them into the moustache bracketed parameters in the associated CloudFormation IaC file. For each property (parameter), provide a `name` and `type`. Optionally, also provide a `description`, `default`,and `pattern`.

The defined parameters for the following example *standard* environment template schema include `vpc_cidr`, `subnet_one_cidr` and `subnet_two_cidr` with the `default` keyword and default values. When you create an environment with this environment template bundle schema, you can accept the default values or provide your own. If a parameter *doesn't* have a default value and is listed as a `required` property (parameter), you must provide values for it when you create an environment.

The second example *standard* environment template schema lists the `required` parameter `my_other_sample_input`.

You can create a schema for two types of environment templates. For more information, see Register and publish templates (p. 64).

- ***Standard* environment templates**

  In the following example, an environment input type is defined with a description and input properties. This schema example can be used with the AWS Proton CloudFormation IaC file shown in Example 3 (p. 30).

  Example schema for a *standard* environment template:

```
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}($|/(16|24))
        subnet_one_cidr:            # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.0.0/2
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(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))
```

Example schema for a *standard* environment template that includes a `required` parameter:

```
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
```

- ***Customer managed* environment templates**

  In the following example, the schema only includes a list of outputs that replicate the outputs from the IaC that you used to provision your *customer managed* infrastructure. You need to define output value types as *strings only* (*not* lists, arrays or other types). For example, the next code snippet shows the outputs section of an external AWS CloudFormation template. This is from the template shown in Example 1 (p. 21). It can be used to create external *customer managed* infrastructure for an AWS Proton Fargate service created from Example 4 (p. 32).

  > **Important**
  > As an administrator, you must ensure that your provisioned and managed infrastructure and all output parameters are compatible with the associated *customer managed* environment templates. AWS Proton can't account for changes on your behalf because these changes aren't visible to AWS Proton. Inconsistencies result in failures.

  Example CloudFormation IaC file outputs for a *customer managed* environment template:

```
// 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'
[...]
```

The schema for the corresponding AWS Proton *customer managed* environment template bundle is shown in the following example. Each output value is defined as a string.

Example schema for a *customer managed* environment template:

```
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"
[...]
```

Schema requirements for service template bundles

Your schema must follow the Data Models (schemas) section of the OpenAPI in YAML format as shown in the following examples. You must provide a schema file in your service template bundle.

In the following service schema examples, you must include the formatted headers. In the following example, this is in the first three lines. This is to establish that you're using the Data Models (schemas) section of the Open API.

A `service_input_type` must be included and defined with a name that you provide. In the following example, this is in line 5. This associates the parameters with an AWS Proton service resource.

An AWS Proton service pipeline is included by default when you use the console or the CLI to create a service. When you include a service pipeline for your service, you must include `pipeline_input_type` with a name that you provide. In the following example, this is in line 7. *Don't* include this parameter if you *aren't* including an AWS Proton service pipeline. For more information, see Register and publish templates (p. 64).

To follow the Open API schema model, you must include `types` In the following example, this is in line 9.

Following `types`, you must define a `service_input_type` type. You define the input parameters for your service as properties of the `service_input_type`. You must include at least one property with a name that matches at least one parameter listed in the service infrastructure as code (IaC) file that is associated with schema.

To define a service pipeline, below your `service_input_type` definition, you must define a `pipeline_input_type`. As above, you must include at least one property with a name that matches at least one parameter listed in a pipeline IaC file that is associated with schema. *Don't* include this definition if you *aren't* including an AWS Proton service pipeline.

When you, as an administrator or developer, create a service and provide customized parameter values, AWS Proton uses the schema file to match, validate and inject them into the associated CloudFormation IaC file's moustache bracketed parameters. For each property (parameter), provide a `name` and a `type`. Optionally, also provide a `description`, `default`, and `pattern`.

The defined parameters for the example schema include `port`, `desired_count`, `task_size` and `image` with the `default` keyword and default values. When you create a service with this service template bundle schema, you can accept the default values or provide your own. The parameter `unique_name` is also included in the example and *doesn't* have a default value. It is listed as a `required` property (parameter). You, as administrator or developer, must provide values for `required` parameters when you create services.

If you want to create a service template with a service pipeline, include the `pipeline_input_type` in your schema.

Example service schema file for a service that includes an AWS Proton service pipeline.

This schema example can be used with the AWS Proton IaC files shown in Example 4 (p. 32) and Example 5 (p. 37). A service pipeline is included.

```
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
```

If you want to create a service template without a service pipeline, *don't* include the `pipeline_input_type` in your schema, as shown in the following example.

Example service schema file for a service that *doesn't* include an AWS Proton service pipeline

```
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"
```

Wrap up template bundles for AWS Proton

After preparing your environment and service infrastructure as code (IaC) files and their respective schema files, you must organize them in directories. You must also create a manifest YAML file.

The manifest file lists the IaC files and needs to adhere to the format and content shown in the following example.

Manifest file format:

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

After you set up the directories and manifest files for your environment or service template bundle, you gzip the directories into a tar ball upload them to an Amazon Simple Storage Service (Amazon S3) bucket where AWS Proton can retrieve them.

When you create a minor version of an environment or a service template that you registered with AWS Proton, you provide the path to your environment or service template bundle tar ball that's located in your S3 bucket. AWS Proton saves it with the new template minor version. You can select the new template minor version to create or update environments or services with AWS Proton.

Environment template bundle wrap up

There are two types of environment template bundles that you create for AWS Proton.

- To create an environment template bundle for a *standard* environment template, organize the schema, infrastructure as code (IaC) files and manifest file in directories as shown in the following environment template bundle directory structure.
- To create an environment template bundle for a *customer managed* environment template, provide only the schema file and directory. *Don't* include the infrastructure directory and files. AWS Proton throws an error if the infrastructure directory and files are included.

For more information, see Register and publish templates (p. 64).

Environment template bundle directory structure:

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

Service template bundle wrap up

To create a service template bundle, you must organize the schema, infrastructure as code (IaC) files, and manifest files into directories as shown in the service template bundle directory structure example.

If you *don't* include a service pipeline in your template bundle, *don't* include the pipeline directory and files and set `"pipelineProvisioning": "CUSTOMER_MANAGED"` when you create the service template that is to be associated with this template bundle.

> **Note**
> You can't modify `pipelineProvisioning` after the service template is created.

For more information, see Register and publish templates (p. 64).

Service template bundle directory structure:

```
/schema
  schema.yaml
/instance_infrastructure
```

```
  manifest.yaml
  cloudformation.yaml
/pipeline_infrastructure
  manifest.yaml
  cloudformation.yaml
```

Template bundle considerations

- **Infrastructure as code (IaC) files**

  AWS Proton audits templates for the correct file format. However, AWS Proton doesn't check for template development, dependency and logic errors. For example, assume that you specified the creation of an Amazon S3 bucket in your AWS CloudFormation IaC file as part of your service or environment template. A service is created based on those templates. Now, suppose at some point you want to delete the service. If the specified S3 bucket is not empty and the CloudFormation IaC file *doesn't* mark it as `Retain` in the `DeletionPolicy`, AWS Proton fails on the service delete operation.
- **Bundle file size limits and format**
  - Bundle file size, count and name size limits can be found at AWS Proton quotas (p. 154).
  - The template bundle directories of files are gzipped into a tar ball and located in an Amazon Simple Storage Service (Amazon S3) bucket.
  - Each file in the bundle must be a valid formatted YAML file.
- **S3 bucket template bundle encryption**

  If you want to encrypt sensitive data in your template bundles at rest in your S3 bucket, you must use SSE-S3 or SSE-KMS keys to allow AWS Proton to retrieve them.

AWS Proton templates

To add your template bundle to your AWS Proton template library, create a template minor version and register it with AWS Proton. When creating the template, provide the name of the Amazon S3 bucket and path for your template bundle. After templates are published, they can be selected by platform team members and developers. After they're selected, AWS Proton uses the template to create and provision infrastructure and applications.

As an administrator, you can create and register an environment template with AWS Proton. This environment template can then be used to deploy multiple environments. For example, it can be used to deploy "dev," "staging," and "prod" environments. The "dev" environment might include a VPC with private subnets and a restrictive access policy to all resources. Environment outputs can be used as inputs for services.

You can create and register environment templates to create two different types of environments. Both you and developers can use AWS Proton to deploy services to both types.

- Register and publish a *standard* environment template that AWS Proton uses to create a *standard* environment that provisions and manages the environment infrastructure.
- Register and publish a *customer managed* environment template that AWS Proton uses to create a *customer managed* environment that connects to your existing provisioned infrastructure. AWS Proton *doesn't* manage your existing provisioned infrastructure.

You can create and register service templates with AWS Proton to deploy services to environments. An AWS Proton environment must be created before a service can be deployed to it.

The following list describes how you create and manage templates with AWS Proton.

- (Optional) Prepare an IAM role to control developer access to AWS Proton API calls and AWS Proton IAM service roles For more information, see IAM Roles (p. 80).
- Compose a template bundle. For more information, see Template bundles (p. 14).
- Create and register a template with AWS Proton after the template bundle is composed, compressed, and saved in an Amazon S3 bucket. You can do this either in the console or by using the AWS CLI.
- Test and use the template to create and manage AWS Proton provisioned resources after it's registered with AWS Proton.
- Create and manage major and minor versions of the template through out the life of the template.

Topics

Versioned templates

As an administrator or a member of a platform team, you define, create and manage a library of versioned templates that are used to provision infrastructure resources.

You register and propagate minor versions in AWS Proton. You can do this without involving developers. Minor versions must be backward compatible. AWS Proton fails a new minor version registration if it's not backward compatible.

Major versions require new inputs from the developer due to template schema changes.

AWS Proton makes a best-effort attempt to determine whether the schema of the new version is backward compatible with the previous minor versions of the template. However, it *doesn't* check if the template bundle infrastructure as code (IaC) file is backward compatible with the previous minor versions. For example, AWS Proton *doesn't* check if the new IaC file will cause breaking changes for the applications that are running on the infrastructure provisioned by a previous minor version of the template.

When a new version's schema is backward compatible, you're responsible for determining how changes to the template bundle infrastructure template files impact applications and resources that are running on previous versions of the template. Based on the determination, you, as an administrator, decide if the new version is designated as a major or minor version.

When you publish a new version of a template, it becomes the **Recommended** version if it's the highest major and minor version. New AWS Proton resources are created using the new recommended version, and AWS Proton prompts administrators to use the new version and to update existing AWS Proton resources that are using an outdated version.

Register and publish templates

You can register and publish environment and service templates with AWS Proton, as described in the following sections.

Register and publish environment templates

You can register and publish the following types of environment templates.

- Register and publish a *standard* environment template that AWS Proton uses to deploy and manage environment infrastructure.
- Register and publish a *customer managed* environment template that AWS Proton uses to connect to your existing provisioned infrastructure that you manage. AWS Proton *doesn't* manage your existing provisioned infrastructure.

Important
As an administrator, ensure that your provisioned and managed infrastructure and all output parameters are compatible with associated *customer managed* environment templates. AWS Proton can't account for changes on your behalf because these changes aren't visible to AWS Proton. Inconsistencies result in failures.

Use the console to register and publish templates.

Register an environment template by using the console and following the steps in Step 2: Create an environment template (p. 8) or by following the next steps.

1. In the AWS Proton console, choose **Environment templates**.
2. Choose **Create environment template**.
3. In the **Create environment template** page, in the **Template options** section, choose one of the two available template options.
   - **Create a template for provisioning new environments**.
   - **Create a template to use provisioned infrastructure that you manage**.
4. In the **Template bundle source** section, choose one of the two available template bundle source options.
   - **Use one of our sample template bundles**.
   - **Use your own template bundle**.
5. Provide a path to the selected template bundle.
   - If you chose **Use one of our sample template bundles**, in the **Sample template bundle** section, select a sample template bundle.
   - If you chose **Use your own template bundle**, in the **S3 bundle location** section, provide a path to your template bundle.
6. In the **Template details** section.
   a. Enter a **Template name**.
   b. (Optional) Enter a **Template display name**.
   c. (Optional) Enter a **Template description** for the environment template.
7. (Optional) Check the checkbox for **Customize encryption settings (advanced)** in the **Encryption settings** section to provide your own encryption key.
8. (Optional) In the **Tags** section, choose **Add new tag** and enter a key and value to create a customer managed tag.
9. Choose **Create Environment template**.

   You're now on a new page that displays the status and details for your new environment template. These details include a list of AWS and customer managed tags. AWS Proton automatically generates AWS managed tags for you when you create AWS Proton resources. For more information, see AWS Proton resources and tagging (p. 150).
10. Refresh the **Template versions** section and the status of the new version is *Draft*. The status of a new environment template status starts in the **Draft** state. You and others with `proton:CreateEnvironment` permissions can view and access it. Follow the next step to make the template available to others.

11. In the **Template versions** section, choose the radio button to the left of the minor version of the template you just created (1.0). As an alternative, you can choose **Publish** in the info alert and skip the next step.
12. In the **Template versions** section, choose **Publish**.
13. The template status changes to **Published**. Because it's the latest version of the template, it's the **Recommended** version.
14. In the navigation pane, select **Environment templates** to view a list of your environment templates and details.

Use the console to register new major and minor versions of an environment template.

1. In the AWS Proton console, choose **Environment Templates**.
2. In the list of environment templates, choose the name of the environment template that you want to create a major or minor version for.
3. In the environment template detail view, choose **Create new version** in the **Template versions** section.
4. In the **Create new environment template version** page, in the **Template details** section, choose one of the following options.
   - To create a minor version, keep the check box **Check to create a new major version** empty.
   - To create a major version, check the check box **Check to create a new major version**.
5. Continue through the console steps to create the new minor or major version.

Use the AWS CLI to register, publish and manage templates as shown in the following steps.

1. Create a *standard* OR *customer managed* environment template by specifying the region, name, display name (optional) and description (optional).

   a. Create a *standard* environment template as shown in the following example command and response.

      Command:

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

      Response:

      ```
      {
          "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. Create a *customer managed* environment template by adding the `provisioning` parameter with value `CUSTOMER_MANAGED` as shown in the following example command and response.

      Command:

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

Response:

```
{
    "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. **This and the remaining steps are the same for both the *standard* and *customer managed* environment templates.**

   Create a minor version 0 of major version 1 of the environment template by including the template name, major version and the S3 bucket *name* and *key* for the bucket that contains your environment template bundle as shown in the following command and response.

   Command:

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

   Response:

```
{
    "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 the *get* command to check the registrations status.

   Command:

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

   Response:

```
{
    "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. Publish of minor version 0 of major version 1 of the environment template by providing the template name and the major and minor version as shown in the following command and response. This version is the `Recommended` version.

   Command:

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

   Response:

```
{
    "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"
    }
}
```

After creating a new template using the AWS CLI, you can view a list of AWS and customer managed tags, as shown in the following example command. AWS Proton automatically generates AWS managed tags for you. You can also modify and create customer managed tags using the AWS CLI. For more information, see AWS Proton resources and tagging (p. 150).

Command:

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

Register and publish service templates

When you create a service template version, you specify a list of compatible environment templates. That way, when developers select a service template, they have options for which environment to deploy their service to. Before creating a service from a service template or before publishing a service template, confirm that environments have been deployed from the listed compatible environment templates. To add or remove compatible environment templates for a service template version, you create a new major version of it. You *can't* update a service to the new major version if it is deployed to an environment that was built from a removed compatible environment template.

You can register and publish a service template by using the console and following the steps in Step 3: Create a service template (p. 9). If you aren't including a service pipeline definition in your service template, uncheck the **Pipeline - optional** checkbox at the bottom of the page. For more information, see Template bundles (p. 14).

You can also use the AWS CLI for AWS Proton. You can create a service template with or without defining a service pipeline. To create service template that deploys a service without a service pipeline, add the parameter and value `--pipeline-provisioning "CUSTOMER_MANAGED"` to the `create-service-template` command. Configure your template bundles as described in Template bundles (p. 14) creation and Schema requirements for service template bundles (p. 59).

> **Note**
> You can't modifty `pipelineProvisioning` after the service template is created.

You can create and publish a service template with or without a service pipeline as shown in the following steps.

1. Create a service template with a service pipeline by specifying the name, display name (optional), and description (optional) as shown in the following example command and response.

   Command:

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

   Response:

   ```
   {
       "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"
       }
   }
   ```

   Create a service template without a service pipeline by adding `--pipeline-provisioning` as shown in the following example command and response.

   Command:

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

Response:

```
{
    "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. Create a minor version 0 of major version 1 of the service template by including the template name, compatible environment templates, major version, and the S3 bucket *name* and *key* for the bucket that contains your service template bundle as shown in the following command and response.

   Command:

```
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"}]'
```

   Response:

```
{
    "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 the *get* command to check the registrations status.

   Command:

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

   Response:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-west-2:123456789012: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. Publish the service template by using the update command to change the status to `"PUBLISHED"` as shown in the following example command and response.

   Command:

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

   Response:

   ```
   {
       "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. Check that AWS Proton has published version 1.0 by using the *get* command to retrieve service template detail data as shown in the following example command and response.

   Command:

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

   Response:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-west-2:123456789012: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"
    }
}
```

View template data

You can view lists of templates with details and view individual templates with detail data by using the AWS Proton console.

Customer managed environment template data includes the `provisioned` parameter with the value `CUSTOMER_MANAGED`.

If a service template *doesn't* include a service pipeline, the service template data includes the `pipelineProvisioning` parameter with the value `CUSTOMER_MANAGED`.

For more information, see Register and publish templates (p. 64).

View templates

1. To view a list of templates, choose **(Environment or Service) templates**.
2. To view detail data choose the name of a template.

   View the detail data of the template, a list of the major and minor versions of the template, a list of the AWS Proton resources that were deployed using template versions and template tags.

   The recommended major version and minor version is labelled as **Recommended**.

You can also use the AWS CLI for AWS Proton by using the get or list operations as shown in the following example commands and responses. You can get or list templates and major or minor versions of a template.

Command:

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

Response:

```
{
    "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"
    }
}
```

Command:

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

Response:

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

View a minor version of a service template as shown in the first example command and response.

Command:

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

Response:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-west-2:123456789012: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"
    }
}
```

View a service template without a service pipeline as shown in the next example command and response.

Command:

```
aws proton get-service-template --name "simple-svc-template-cli"
```

Response:

```
{
    "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"
    }
}
```

Update a template

You can update a template as described in the following list.

- Edit the `description` or `display name` of a template when you use either the console or AWS CLI. You *can't* edit the `name` of a template.
- Update the status of a template minor version when you use either the console or AWS CLI. You can only change the status from `DRAFT` to `PUBLISHED`.
- Edit the display name and description of a minor or major version of a template when you use the AWS CLI.

Edit a template description and display name using the console as described in the following steps.

In the list of templates.

1. In the AWS Proton console, choose **(Environment or Service) Templates**.
2. In the list of templates, choose the radio button to the left of the template that you want to update the description or display name for.
3. Choose **Actions** and then **Edit**.
4. In the **Edit (environment or service) template** page, in the **Template details** section, enter your edits in the form and choose **Save changes**.

Change the status of a minor version of a template using the console to publish a template as described in the following. You can only change the status from `DRAFT` to `PUBLISHED`.

In the (environment or service) template detail page.

1. In the AWS Proton console, choose **(Environment or Service) templates**.
2. In the list of templates, choose the name of the template that you want to update the status of a minor version from **Draft** to **Published**.
3. In the (environment or service) template detail page, in the **Template versions** section, select the radio button to the left of the minor version that you want to publish.
4. Choose **Publish** in the **Template versions** section. The status changes from **Draft** to **Published**.

You can also use the AWS CLI for AWS Proton to update templates. The following example command and response shows how you can edit the description of an environment template.

Command:

```
aws proton update-environment-template --name "simple-env" --description "A single VPC with
 public access"
```

Response:

```
{
    "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"
    }
}
```

You can also use the AWS CLI to update service templates. See Register and publish service templates (p. 69), step 5, for an example of updating the status of a minor version of a service template.

Delete templates

Templates can be deleted using the console and AWS CLI.

You can delete a minor version of an environment template if there are no environments deployed to that version.

You can delete a minor version of a service template if there are no service instances or pipelines deployed to that version. Your pipeline can be deployed to a different template version than your service instance. For example, if your service instance has been updated to version 1.1 from 1.0 and your pipeline is still deployed to version 1.0, you can't delete service template 1.0.

You can use the console to delete the entire template or individual minor and major versions of a template.

Use the console to delete templates as follows.

> **Note**
>
> **When using the console to delete templates.**
>
> - When you delete the entire template, you also delete the major and minor versions of the template.

In the list of (environment or service) templates.

1. In the AWS Proton console, choose **(Environment or Service) Templates**.
2. In the list of templates, select the radio button to the left of the template you want to delete.

   You can only delete an entire template if there are no AWS Proton resources deployed to its versions.
3. Choose **Actions** and then **Delete** to delete the entire template.
4. A modal prompts you to confirm the delete action.
5. Follow the instructions and choose **Yes, delete**.

In the (environment or service) template detail page.

1. In the AWS Proton console, choose **(Environment or Service) Templates**.
2. In the list of templates, choose the name of the template that you want to entirely delete or delete individual major or minor versions of it.
3. **To delete the entire template.**

   You can only delete an entire template if there are no AWS Proton resources deployed to its versions.

   a. Choose **Delete**, top right corner of page.
   b. A modal prompts you to confirm the delete action.
   c. Follow the instructions and choose **Yes, delete**.
4. **To delete major or minor versions of a template.**

   You can only delete a minor version of a template if there are no AWS Proton resources deployed to that version.

   a. In the **Template versions** section, select the radio button to the left of the version that you want to delete.
   b. Choose **Delete** in the **Template versions** section.
   c. A modal prompts you to confirm the delete action.
   d. Follow the instructions and choose **Yes, delete**.

AWS CLI template delete operations *don't* include the deletion of other versions of a template. When using the AWS CLI, delete templates with the following conditions.

- Delete an entire template if no minor or major versions of the template exist.

- Delete a major version when you delete the last remaining minor version.
- Delete a minor version of a template if there are no AWS Proton resources deployed to that version.
- Delete the recommended minor version of a template if no other minor versions of the template exist and there are no AWS Proton resources deployed to that version.

The following example commands and responses show how to use the AWS CLI to delete templates.

Command:

```
aws proton delete-environment-template-version --template-name "simple-env" --major-
version "1" --minor-version "0"
```

Response:

```
{
    "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"
    }
}
```

Command:

```
aws proton delete-environment-template --name "simple-env"
```

Response:

```
{
    "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"
    }
}
```

Command:

```
aws proton delete-service-template-version --template-name "fargate-service" --major-
version "1" --minor-version "0"
```

Response:

```
{
    "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"
    }
}
```

AWS Proton environments

For AWS Proton, an environment represents the set of shared resources and policies that AWS Proton services (p. 97) are deployed into. They can contain any resources that are expected to be shared across AWS Proton service instances. These resources can include VPCs, clusters, and shared load balancers or API Gateways. An AWS Proton environment must be created before a service can be deployed to it.

This section describes how to manage environments using create, view, update and delete operations. For additional information, see the *The AWS Proton Service API Reference*.

Topics

IAM Roles

With AWS Proton, you supply the IAM roles and AWS KMS keys for the AWS resources that you own and manage. These are later applied to and used by resources owned and managed by developers. You create an IAM role to control your developer team's access to the AWS Proton API.

AWS Proton service role

When you create a new environment, you must apply an IAM service role that AWS Proton needs to assume to deploy to the environment. You define the role. The role must contain all permissions that are necessary for AWS Proton to update all provisioned infrastructure defined both in the environment templates and the service templates. For more information, see AWS Proton service role (p. 135). If you use environment account connections and environment accounts, you create the role in a selected environment account. For more information, see Create an environment in one account and provision in another account (p. 82) and Environment account connections (p. 90).

Create an environment

You can create two types of environments:

- Create, manage and provision a *standard* environment by using *standard* environment template to create an environment.
- Connect AWS Proton to *customer managed* infrastructure by using a *customer managed* environment template to create an environment.

You can also use two different environment provisioning methods when you create environments.

- Create, manage and provision an environment in a single account.
- In a single management account create and manage an environment that is provisioned in another account with environment account connections. For more information, see Create an environment in one account and provision in another account (p. 82) and Environment account connections (p. 90).

See Template bundles (p. 14) and Schema requirements for environment template bundles (p. 57) for additional requirements.

Create and provision an environment in the same account

Use the console or AWS CLI to create environments in a single account.

Use the console to create an environment by following the steps in Step 4: Create an environment (p. 9) or by following these steps.

1. In the AWS Proton console, choose **Environments**.
2. Choose **Create environment**.
3. In the **Choose an environment template** page, select a template and choose **Configure**.
4. In the **Configure environment** page, in the **Environment** settings section, enter an **Environment name**.
5. Select the ARN of the AWS Proton service role that you created as part of Setting up AWS Proton service roles (p. 4) in **Environment roles**.
6. Choose **Next**.
7. In the **Configure environment custom settings** page, you must enter values for the `required` parameters. You can enter values for the `optional` parameters or use the defaults when given.
8. Choose **Next** and review your inputs.
9. Choose **Create**.

   View the environment details and status, as well as the AWS managed tags and customer managed tags for your environment.
10. In the navigation pane, choose **Environments**.

    A new page displays a list of your environments along with the status and other environment details.

Use the AWS CLI to create an environment as defined by a selected environment template.

You specify the AWS Proton service role (p. 135) ARN, path to your spec file, environment name, environment template ARN, the major and minor versions, and description (optional).

The next examples shows a YAML formatted spec file that defines values for two inputs, based on the environment template schema file. You can use the `get-environment-template-minor-version` command to view the environment template schema.

Spec:

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

Create an environment as shown in the following example CLI command and response.

Command:

```
aws proton create-environment --name "MySimpleEnv" --template-
name simple-env --template-major-version 1 --proton-service-role-arn
 "arn:aws:iam::123456789012:role/AWSProtonServiceRole" --spec "file://env-spec.yaml"
```

Response:

```
{
    "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"
    }
}
```

After you create a new environment, you can view a list of AWS and customer managed tags, as shown in the following example command. AWS Proton automatically generates AWS managed tags for you. You can also modify and create customer managed tags using the AWS CLI. For more information, see AWS Proton resources and tagging (p. 150).

Command:

```
aws proton list-tags-for-resource --resource-arn "arn:aws:proton:region-
id:123456789012:environment/MySimpleEnv"
```

Create an environment in one account and provision in another account

Use the console or AWS CLI to create an environment in a management account that provisions environment infrastructure in another account.

Before using the console or CLI, complete the following steps.

1. Identify the AWS account IDs for the management and environment account, and copy them for later use.
2. In the environment account, create an AWS Proton service role with minimum permissions for the environment to create. For more information, see AWS Proton service role (p. 135).

Use the console as shown in the following steps.

1. In the environment account, create an environment account connection, and use it to send a request to connect to the management account.

   a. In AWS Proton console, choose **Environment account connections** in the navigation pane.
   b. In the **Environment account connections** page, choose **Request to connect**.

      Note
      Verify that the account ID listed in the **Environment account connection** page heading matches your pre-identified environment account ID.

   c. In the **Request to connect** page, in the **Environment role** section, select **Existing service role** and the name of the service role that you created for the environment.
   d. In the **Connect to management account** section, enter the **Management account ID** and an **Environment name** for your AWS Proton environment. Copy the name for later use.
   e. Choose **Request to connect** at the lower right corner of the page.
   f. Your request shows as pending in the **Environment connections sent to a management account** table and a modal shows how to accept the request from the management account.

2. **In the management account, accept a request to connect from the environment account.**

   a. Log in to your management account and choose **Environment account connections** in the AWS Proton console.
   b. In the **Environment account connections** page, in the **Environment account connection requests** table, select the environment account connection with the environment account ID that matches your pre-identified environment account ID.

      Note
      Verify that the account ID listed in the **Environment account connection** page heading matches your pre-identified management account ID.
   c. Choose **Accept**. The status changes from pending to connected.

3. **In the management account, create an environment.**

   a. Choose **Environment templates** in the navigation pane.
   b. In the **Environment templates** page, choose **Create environment template**.
   c. In the **Choose an environment template** page, choose an environment template.
   d. In the **Configure environment** page, in the **Deployment account** section, select **Another AWS account**.
   e. In the **Environment details** section, select your **Environment account connection** and **Environment name**.
   f. Choose **Next**.
   g. Fill out the forms and choose **Next** until you reach the **Review and Create** page.
   h. Review and choose **Create environment**.

Use the AWS CLI to create an environment in one account and deploy to another account, as shown in the following example.

In the environment account, create an environment account connection and request as shown in the following example command and response.

Command:

```
aws proton create-environment-account-connection --environment-name "simple-env-connected"
 --role-arn "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-role"
 --management-account-id "123456789111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "PENDING"
    }
}
```

In the management account, accept the environment account connection request as shown in the next example command and response.

Command:

```
aws proton accept-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "CONNECTED"
    }
}
```

View your environment account connection by using the *get* as shown in the following command and response.

Command:

```
aws proton get-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "CONNECTED"
    }
}
```

In the management account, create an environment as shown in the following example command and response.

Command:

```
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"
```

Response:

```
{
    "environment": {
        "arn": "arn:aws:proton:us-east-1:123456789111:environment/simple-env-connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
        "templateName": "simple-env-template"
    }
}
```

View environment data

You can view environment detail data using either the console or the AWS CLI.

You can view lists of environments with details and individual environments with detail data by using the AWS Proton console.

1. To view a list of your environments, choose **Environments** in the navigation pane.
2. To view detail data, choose the name of an environment.

   View your environment detail data.

You can also use the AWS CLI for AWS Proton by using the get or list operations as shown in the following example command and response. You can get or list environments.

Command:

```
aws proton get-environment --name "MySimpleEnv"
```

Response:

```
{
    "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"
    }
}
```

Update an environment

If the environment is associated with an environment account connection, *don't* update or include the `protonServiceRoleArn` parameter to update or connect to an environment account connection.

You can only update to a new environment account connection if it was created in the same environment account that the current environment account connection was created in and it is associated with the current environment.

If the environment *isn't* associated with an environment account connection, *don't* update or include the `environmentAccountConnectionId` parameter.

You can update either the `environmentAccountConnectionId` or `protonServiceRoleArn` parameter and value. You can't update both.

There are four modes for updating an environment as described in the following list. When using the AWS CLI, the `deployment-type` field defines the mode. When using the console, these modes map to the **Edit**, **Update**, **Update minor**, and **Update major** actions that drop down from **Actions**.

`NONE`

In this mode, a deployment *doesn't* occur. Only the requested metadata parameters are updated.

`CURRENT_VERSION`

In this mode, the environment is deployed and updated with the new spec that you provide. Only requested parameters are updated. *Don't* include minor or major version parameters when you use this `deployment-type`.

`MINOR_VERSION`

In this mode, the environment is deployed and updated with the published, recommended (latest) minor version of the current major version in use by default. You can also specify a different minor version of the current major version in use.

`MAJOR_VERSION`

In this mode, the environment is deployed and updated with the published, recommended (latest) major and minor version of the current template by default. You can also specify a different major version that is higher than the major version in use and a minor version (optional).

You can attempt to cancel an environment update deployment if the `deploymentStatus` is in `IN_PROGRESS`. AWS Proton attempts to cancel the deployment. Successful cancellation *isn't* guaranteed.

When you cancel an update deployment, AWS Proton attempts to cancel the deployment as listed in the following steps.

- Sets the deployment state to `CANCELLING`.

- Stops the deployment in progress and deletes any new resources that were created by the deployment when `IN_PROGRESS`.
- Sets the deployment state to `CANCELLED`.
- Reverts the state of the resource to what it was before the deployment was started.

For more information on cancelling an environment deployment, see CancelEnvironmentDeployment in the *AWS Proton API Reference*.

Use the console and AWS CLI to make updates or cancel update deployments.

Update an environment using the console as shown in the following steps.

1. **Choose 1 of the following 2 steps.**

   a. **In the list of environments.**

      i. In the AWS Proton console, choose **Environments**.
      ii. In the list of environments, choose the radio button to the left of the environment template that you want to update.

   b. **In the console environment detail page.**

      i. In the AWS Proton console, choose **Environments**.
      ii. In the list of environments, choose the name of the environment that you want to update.

2. **Choose 1 of the next 4 steps to update your environment.**

   a. **To make an edit that doesn't require environment deployment.**

      i. For example, to change a description.

         Choose **Edit**.
      ii. Fill out the form and choose **Next**.
      iii. Review your edit and choose **Update**.

   b. **To make updates to metadata inputs only.**

      i. Choose **Actions** and then **Update**.
      ii. Fill out the form and choose **Edit**.
      iii. Fill out the forms and choose **Next** until you reach the **Review** page.
      iv. Review your updates and choose **Update**.

   c. **To make an update to a new minor version of its environment template.**

      i. Choose **Actions** and then **Update minor**.
      ii. Fill out the form and choose **Next**.
      iii. Fill out the forms and choose **Next** until you reach the **Review** page.
      iv. Review your updates and choose **Update**.

   d. **To make an update to a new major version of its environment template.**

      i. Choose **Actions** and then **Update major**.
      ii. Fill out the form and choose **Next**.
      iii. Fill out the forms and choose **Next** until you reach the **Review** page.
      iv. Review your updates and choose **Update**.

Use the AWS Proton AWS CLI to update an environment to update to a new minor version as shown in the following example commands and responses.

Command: to update

```
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"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-environment --name "MySimpleEnv"
```

Response:

```
{
    "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"
    }
}
```

Use the console to cancel an environment update deployment as shown in the following steps.

1. In the AWS Proton console, choose **Environments** in the navigation pane.
2. In the list of environments, choose the name of the environment with the deployment update that you want to cancel.
3. If your update deployment status is **In progress**, in the environment detail page, choose **Actions** and then **Cancel deployment**.
4. A modal prompts you to confirm the cancellation. Choose **Cancel deployment**.
5. Your update deployment status is set to **Cancelling** and then **Cancelled** to complete the cancellation.

Use the AWS Proton AWS CLI to cancel an IN_PROGRESS environment update deployment to a new minor version 2 as shown in the following commands and responses.

A wait condition is included in the template used for this example so that the cancellation starts before the update deployment succeeds.

Command: to cancel

```
aws proton cancel-environment-deployment --environment-name "MySimpleEnv"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-environment --name "MySimpleEnv"
```

Response:

```
{
    "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"
    }
}
```

Delete an environment

You can delete an environment by using the console or the AWS CLI.

Delete an environment using the console as described in the following two options.

In the list of environments.

1. In the AWS Proton console, choose **Environments**.
2. In the list of environments, select the radio button to the left of the environment you want to delete.
3. Choose **Actions** and then **Delete**.
4. A modal prompts you to confirm the delete action.
5. Follow the instructions and choose **Yes, delete**.

In the environment detail page.

1. In the AWS Proton console, choose **Environments**.
2. In the list of environments, choose the name of the environment you want to delete.
3. In the environment detail page, choose **Actions** and then **Delete**.
4. A modal prompts you to confirm the delete action.
5. Follow the instructions and choose **Yes, delete**.

Use the AWS CLI to delete an environment as shown in following example command and response. *Don't* delete an environment if services or service instances are deployed to the environment.

Command:

```
aws proton delete-environment --name "MySimpleEnv"
```

Response:

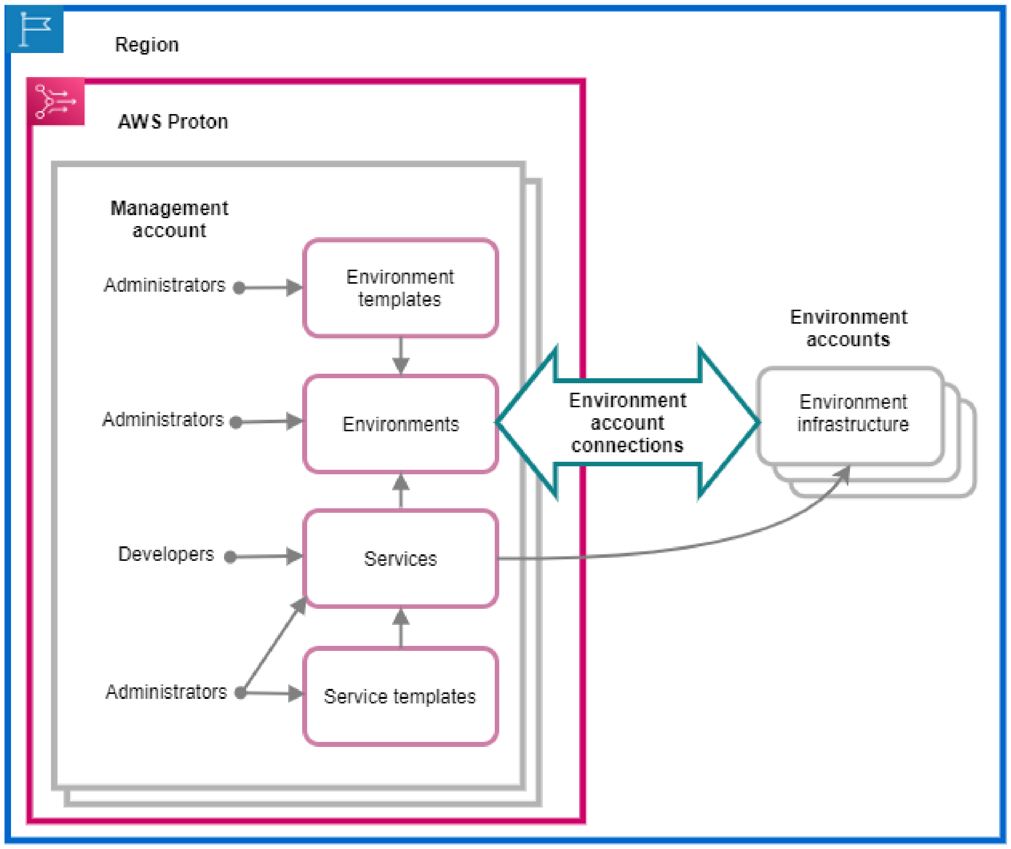
```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "DELETE_IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Environment account connections

Overview

Learn how to create and manage an environment in one account and provision its infrastructure resources in another account to improve visibility and efficiency at scale.

Terminology



With AWS Proton *environment account connections*, you can create an AWS Proton environment from one account and provision it's infrastructure in another account.

Management account

The single account where you, as an administrator, create an AWS Proton environment that provisions infrastructure resources in another *environment account*.

Environment account

An account that environment infrastructure is provisioned in, when you create an AWS Proton environment in another account.

Environment account connection

A secure bi-directional connection between a *management account* and an *environment account*. It maintains authorization and permissions as described further in the following sections.

When you create an environment account connection in an environment account, in a specific Region, only the management accounts in the same Region can see and use the environment account connection. This means that the AWS Proton environment created in the management account and the environment infrastructure provisioned in the environment account must be in the same Region.

Environment account connection considerations

- You need an environment account connection for each environment that you want to provision in an environment account.
- For information about environment account connection quotas, see AWS Proton quotas (p. 154).

Create an environment in one account and provision its infrastructure in another account

To create and provision an environment from a single management account, set up an environment account for an environment that you plan to create.

Start in the environment account and create connection.

In the environment account, create an AWS Proton service role that's scoped down to only the permissions that are needed for provisioning your environment infrastructure resources. For more information, see AWS Proton service role (p. 135).

Then create and send an environment account connection request to your management account. When the request is accepted, AWS Proton can use the associated IAM role that permits environment resource provisioning in the associated environment account.

In the management account, accept or reject the environment account connection.

In the management account, accept or reject the environment account connection request. You *can't* delete an environment account connection from your management account.

If you accept the request, the AWS Proton can use the associated IAM role that permits resource provisioning in the associated environment account.

The environment infrastructure resources are provisioned in the associated environment account. You can only use AWS Proton APIs to access and manage your environment and it's infrastructure resources, from your management account. For more information, see Create an environment in one account and provision in another account (p. 82) and Update an environment (p. 86).

After you reject a request, you *can't* accept or use the rejected environment account connection.

> **Note**
> You *can't* reject an environment account connection that's connected to an environment. To reject the environment account connection, you must first delete the associated environment.

In the environment account, access the provisioned infrastructure resources.

In the environment account, you can view and access the provisioned infrastructure resources. For example, you can use CloudFormation API actions to monitor and clean up stacks if needed. You can't use the AWS Proton API actions to access or manage the AWS Proton environment that was used to provision the infrastructure resources.

In the environment account, you can delete environment account connections that you have created in the environment account. You *can't* accept or reject them. If you delete an environment account connection that's in use by an AWS Proton environment, AWS Proton *won't* be able to manage the environment infrastructure resources until a new environment connection has been accepted for the environment account and named environment. You are responsible for cleaning up provisioned resources that remain without an environment connection.

Use the console to manage environment account connections

Use the console to create an environment account connection and send a request to the management account as shown in the next steps.

1. Decide on a name for the environment that you plan to create in your management account or choose the name of an existing environment that requires an environment account connection.

2. In an environment account, in the AWS Proton console, choose **Environment account connections** in the navigation pane.
3. In the **Environment account connections** page, choose **Request to connect**.

   Note
   Verify that the account ID listed in the **Environment account connection** page heading matches the account ID of the environment account that you want your named environment to provision in.
4. In the **Request to connect** page, in the **Environment role** section, choose **New service role** and AWS Proton automatically creates a new role for you. Or, select **Existing service role** and the name of the service role that you created previously.

   Note
   The role that AWS Proton automatically creates for you has broad permissions. We recommend that you scope down the role to the permissions required to provision your environment infrastructure resources. For more information, see AWS Proton service role (p. 135).
5. In the **Connect to management account** section, enter the **Management account ID** and the **Environment name** that you entered in step 1.
6. Choose **Request to connect** at the lower right corner of the page.
7. Your request shows as pending in the **Environment connections sent to a management account** table and a modal lets you know how to accept the request from the management account.

Accept or reject an environment account connection request.

1. In a management account, in the AWS Proton console, choose **Environment account connections** in the navigation pane.
2. In the **Environment account connections** page, in the **Environment account connection requests** table, choose the environment connection request to accept or reject.

   Note
   Verify that the account ID listed in the **Environment account connection** page heading matches the account ID of the management account that's associated with the environment account connection to reject. After you reject this environment account connection, you *can't* accept or use the rejected environment account connection.
3. Choose **Reject** or **Accept**.
   - If you selected **Reject**, the status changes from *pending* to *rejected*.
   - If you selected **Accept**, the status changes from *pending* to *connected*.

Delete an environment account connection.

1. In an environment account, in the AWS Proton console, choose **Environment account connections** in the navigation pane.

   Note
   Verify that the account ID that's listed in the **Environment account connection** page heading matches the account ID of the management account that's associated with the environment account connection to reject. After you delete this environment account connection, AWS Proton *can't* manage the environment infrastructure resources in the environment account until a new environment account connection for the environment account and named environment is accepted by the management account.
2. In the **Environment account connections** page, in the **Sent requests to connect to management account** section, choose **Delete**.
3. A modal prompts you to confirm the deletion. Choose **Delete**.

Use the AWS CLI to manage environment account connections

Decide on a name for the environment that you plan to create in your management account or choose the name of an existing environment that requires an environment account connection.

Create an environment account connection in an environment account as shown in the following example command and response.

Command:

```
aws proton create-environment-account-connection --environment-name "simple-env-connected"
 --role-arn "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-role"
 --management-account-id "123456789111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "PENDING"
    }
}
```

Accept or reject an environment account connection in a management account as shown in the following command and response.

> **Note**
> If you reject this environment account connection, you *won't* be able to accept or use the rejected environment account connection.

If you specify **Reject**, the status changes from *pending* to *rejected*.

If you specify **Accept**, the status changes from *pending* to *connected*.

Command to accept:

```
aws proton accept-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "CONNECTED"
    }
}
```

Command to reject:

```
aws proton reject-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-reject",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role"
    }
}
```

View an environment account connections as shown in the following *get* example command and response. You can *get* or *list* environment account connections

Command:

```
aws proton get-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "CONNECTED"
    }
}
```

Delete an environment account connection in an environment account as shown in the following command and response.

Note
If you delete this environment account connection, AWS Proton *won't* be able to manage the environment infrastructure resources in the environment account until a new environment connection has been accepted for the environment account and named environment. You are responsible for cleaning up provisioned resources that remain without an environment connection.

Command:

```
aws proton delete-environment-account-connection --id "a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
```

Response:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:123456789222:environment-account-connection/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "123456789222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "123456789111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::123456789222:role/service-role/env-account-proton-service-
role",
        "status": "CONNECTED"
    }
}
```

AWS Proton services

An AWS Proton service is an instantiation of a service template, normally including several service instances and a pipeline. An AWS Proton service instance is an instantiation of a service template in a specific environment (p. 80). A service template is a complete definition of the infrastructure and optional service pipeline for a AWS Proton service.

This section shows how to manage services by using create, view, update and delete operations. For additional information, see the *The AWS Proton Service API Reference*.

Topics

Create a service

When you create a service, you can choose from two different types of service templates as shown in the following list.

- A service template that includes a service pipeline (default).
- A service template that *doesn't* include a service pipeline.

Create a service using the console.

Use the console to deploy a service by following the steps shown in Step 5: Optional - Create a service and deploy an application (p. 10). When you *don't* want to use an enabled pipeline, choose a template marked with *Excludes pipeline* for your service.

Use the AWS CLI for AWS Proton to deploy a service.

When you use the AWS CLI, you specify service inputs in a YAML formatted file, `.aws-proton/service.yaml`, located in your source code directory. If you want to use a service template that has `pipelineProvisioning: "CUSTOMER_MANAGED"`, *don't* include the `pipeline:` section in your spec and *don't* include `--repository-connection-arn`, `--repository-id`, and `--branch-name` parameters in your `create-service` command.

You can use the `get-service-template-minor-version` command to view the schema required and optional parameters that you provide values for in your spec file.

Create a service with a service pipeline as shown in the following steps.

1. Set up the service role (p. 137) for the pipeline as shown in the following example command.

   Command:

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

2. The following shows an example spec, based on the service template schema, that includes the service pipeline and instance inputs.

   Spec:

```
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"
```

   Create a service as defined by a service template by specifying the name, repository connection ARN, repository ID, repository branch, service template ARN, spec, environment name, environment template ARN, the major and minor versions, and description (optional) as shown in the following command and response.

   Command:

```
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"
```

   Response:

```
{
    "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"
    }
}
```

Create a service without a service pipeline as shown in the following example command and response.

The following shows an example spec that *doesn't* include service pipeline inputs.

Spec:

```
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"
```

To create a service *without* a provisioned service pipeline, you provide the path to a `spec.yaml` and you *don't* include repository parameters as shown in the following example command and response.

Command:

```
aws proton create-service --name "MySimpleServiceNoPipeline" --template-major-version "1"
 --template-name "fargate-service" --spec "file://spec-no-pipeline.yaml"
```

Response:

```
{
    "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",
        "templateName": "fargate-service-no-pipeline",
        "status": "CREATE_IN_PROGRESS"
    }
}
```

View service data

You can view service detail data using either the console or the AWS CLI.

You can view lists of services with details and view individual services with detail data by using the AWS Proton console.

1. To view a list of your services, choose **Services** in the navigation pane.
2. To view detail data, choose the name of a service.

   View your service detail data.

You can also use the AWS CLI for AWS Proton by using the get or list operations as shown in the following examples. You can get or list services.

View the details of a service with a service pipeline as shown in the following example command and response.

Command:

```
aws proton get-service --name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

View the details of a service without a service pipeline as shown in the following example command and response.

Command:

```
aws proton get-service --name "simple-svc-no-pipeline"
```

Response:

```
{
    "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"
    }
}
```

Edit a service

You can make the following edits to a service.

- Edit the service description.
- Edit a service by adding and removing service instances.

Edit service description

Edit a service using the console as described in the following steps.

In the list of services.

1. In the AWS Proton console, choose **Services**.
2. In the list of services, choose the radio button to the left of the service that you want to update.
3. Choose **Edit**.
4. In the **Configure service** page, fill out the form and choose **Next**.
5. In the **Configure custom settings** page, choose **Next**.
6. Review your edits and choose **Save changes**.

In the service detail page.

1. In the AWS Proton console, choose **Services**.
2. In the list of services, choose the name of the service that you want to edit.
3. In the service detail page, choose **Edit**.
4. In the **Configure service** page, fill out the form and choose **Next**.
5. In the **Configure custom settings** page, fill out the form and choose **Next**.
6. Review your edits and choose **Save changes**.

You can also use the AWS CLI to edit a description as shown in the next example command and response.

Command:

```
aws proton update-service --name "MySimpleService" --description "Edit by updating
 description"
```

Response:

```
{
    "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"
    }
}
```

Edit by adding or removing service instances

For an AWS Proton service, you can add or delete service instances by submitting an edited `spec` (p. 102), if your request *isn't* made under the following conditions. AWS Proton will fail your request under these conditions.

- Your service and pipeline is already being edited or deleted when you submit the edit request.
- Your edited `spec` includes edits that modify the service pipeline or existing service instances that *aren't* to be deleted.

Deletion-failed instances are service instances in the `DELETE_FAILED` state. When you request a service edit, AWS Proton attempts to remove the deletion-failed instances for you, as part of the edit process. If any of your service instances failed to delete, there might still be resources that are associated with the instances, even though they aren't visible from the console or AWS CLI. Check your deletion-failed instance infrastructure resources and clean them up so that AWS Proton can remove them for you.

Add or remove service instances

For the quota of service instances for a service, see AWS Proton quotas (p. 154). You also must maintain at least 1 service instance for your service after it's created. During the update process, AWS Proton makes a count of the existing service instances and the instances to be added or removed. Deletion-failed instances are included in this count and you must account for them when you edit your `spec`.

After you submit a service edit to delete and add service instances, AWS Proton takes the following actions.

- Sets the service to `UPDATE_IN_PROGRESS`.
- If the service has a pipeline, sets its status to `IN_PROGRESS` and blocks pipeline actions.
- Sets any service instances that are to be deleted to `DELETE_IN_PROGRESS`.
- Blocks service actions.
- Blocks actions on service instances that are marked for deletion.
- Creates new service instances.
- Deletes instances that you listed for deletion.
- Attempts to remove deletion-failed instances.
- After additions and deletions are complete, re-provisions the service pipeline (if there is one), sets your service to `ACTIVE` and enables service and pipeline actions.

AWS Proton attempts to remediate failure modes as follows.

- If one or more service instances *failed to be created*, AWS Proton tries to de-provision all of the newly created service instances and reverts the `spec` to the previous state. It *doesn't* delete any service instances and it *doesn't* modify the pipeline in any way.
- If one or more service instances *failed to be deleted*, AWS Proton re-provisions the pipeline without the deleted instances. The `spec` is updated to include the added instances and to exclude the instances that were marked for deletion.
- If the *pipeline fails provisioning*, a rollback *isn't* attempted and both the service and pipeline reflect a failed update state.

Tagging and service edits

When you add service instances as part of your service edit, AWS managed tags propagate to and are automatically created for the new instances and provisioned resources. If you create new tags, those tags are only applied to the new instances. Existing service customer managed tags also propagate to the new instances. For more information, see AWS Proton resources and tagging (p. 150).

Use the console or AWS CLI to edit a service

You can use the AWS Proton console and AWS CLI to edit a service by adding and removing instances.

Edit your service by adding or removing an instance using the console.

In the AWS Proton console

1. In the navigation pane, choose **Services**.
2. choose the name of the service you want to edit.
3. Choose **Edit**.
4. (Optional) In the **Configure service** page, edit the service name or description and choose **Next** at the lower right corner of the page.
5. In the **Configure custom settings** page, choose **Delete** to delete a service instance and choose **Add new instance** to add a service instance and fill out the form.
6. Choose **Next** in the lower right hand corner of the page.
7. Review your update and choose **Save changes**.
8. A modal asks you to verify deletion of service instances. Follow the instructions and choose **Yes, delete**.
9. In the service detail page, view the status details for your service.

The following is an example of using the AWS CLI with an edited `spec` to add and delete service instances.

When you use the CLI, your `spec` must *exclude* the service instances to delete and *include* both the service instances to add and the existing service instances that you *haven't* marked for deletion.

The following listing shows the example `spec` before the edit and a list of the service instances deployed by the spec. This spec was used in the previous example for editing a service description.

Spec:

```
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"
```

The following example `list-service-instances` command and response shows the active instances prior to adding or deleting a service instance.

Command:

```
aws proton list-service-instances --service-name "MySimpleService"
```

Response:

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

The following listing shows the example edited `spec` used to delete and add an instance. The existing instance named `my-instance` is removed and a new instance named `yet-another-instance` is added.

Spec:

```
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"
```

The next listing shows the CLI command and response to edit the service.

Command:

```
aws proton update-service --name "MySimpleService" --description "Edit by adding and
 deleting a service instance" --spec "file://spec.yaml"
```

Response:

```
{
    "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"
    }
}
```

The following `list-service-instances` command and response confirms that the existing instance named `my-instance` is removed and a new instance named `yet-another-instance` is added.

Command:

```
aws proton list-service-instances --service-name "MySimpleService"
```

Response:

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

Delete a service

You can delete a service by using the console or the AWS CLI.

Delete a service using the console.

In the service detail page.

1. In the AWS Proton console, choose **Services**.
2. In the list of services, choose the name of the service that you want to delete.
3. In the service detail page, choose **Actions** and then **Delete**.
4. A modal prompts you to confirm the delete action.
5. Follow the instructions and choose **Yes, delete**.

You can also use the AWS CLI for AWS Proton to delete a service as shown in the following steps.

Command:

```
aws proton delete-service --name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

View service instance data

You can view service instance detail data using either the console or the AWS CLI.

View a list of service instances with details and view individual service instances with detail data by using the AWS Proton console.

1. To view a list of your service instances, choose **Services instances** in the navigation pane.
2. To view detail data, choose the name of a service instance.

   View your service instance detail data.

You can also use the AWS CLI for AWS Proton by using *get* or *list* operations as shown in the following example commands and responses. You can get or list service instances.

Command:

```
aws proton list-service-instances
```

Response:

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

Command:

```
aws proton get-service-instance --name "instance-one" --service-name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Update a service instance

There are four modes for updating a service instance as described in the following list. When using the AWS CLI, the `deployment-type` field defines the mode. When using the console, these modes map to the **Edit** and the **Update to latest minor version** and **Update to latest major version** actions that drop down from **Actions** in the service instance detail page.

`NONE`

In this mode, a deployment *doesn't* occur. Only the requested metadata parameters are updated.

`CURRENT_VERSION`

In this mode, the service instance is deployed and updated with the new spec that you provide. Only requested parameters are updated. *Don't* include minor or major version parameters when you use this `deployment-type`.

`MINOR_VERSION`

In this mode, the service instance is deployed and updated with the published, recommended (latest) minor version of the current major version in use by default. You can also specify a different minor version of the current major version in use.

`MAJOR_VERSION`

In this mode, the service instance is deployed and updated with the published, recommended (latest) major and minor version of the current template by default. You can also specify a different major version that is higher than the major version in use and a minor version (optional).

You can attempt to cancel a service instance update deployment if the `deploymentStatus` is `IN_PROGRESS`. AWS Proton attempts to cancel the deployment. Successful cancellation *isn't* guaranteed.

When you cancel an update deployment, AWS Proton attempts to cancel the deployment as listed in the following steps.

- Sets the deployment state to `CANCELLING`.
- Stops the deployment in process and deletes any new resources that were created by the deployment when `IN_PROGRESS`.
- Sets the deployment state to `CANCELLED`.
- Reverts the state of the resource to what it was before the deployment was started.

For more information on cancelling a service instance deployment, see CancelServiceInstanceDeployment in the *AWS Proton API Reference*.

Update a service instance using the console as described in the following steps.

1. In the AWS Proton console, choose **Service instances** in the navigation pane.
2. In the list of service instances, choose the name of the service instance that you want to update.
3. Choose **Actions** and then choose one of the update options, **Edit** to update spec or **Actions** and then **Update to latest minor version**, or **Update to latest major version**.
4. Fill out each form and choose **Next** until you reach the **Review** page.
5. Review your edits and choose **Update**.

The following example commands and responses show how to use the AWS CLI to update a service instance to a new minor version.

Command: to update

```
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"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-service-instance --name "instance-one" --service-name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Use the console to cancel a service instance deployment as shown in the following steps.

1. In the AWS Proton console, choose **Service instances** in the navigation pane.
2. In the list of service instances, choose the name of the service instance with the deployment update that you want to cancel.
3. If your update deployment status is **In progress**, in the service instance detail page, choose **Actions** and then **Cancel deployment**.
4. A modal asks you to confirm the cancellation. Choose **Cancel deployment**.
5. Your update deployment status is set to **Cancelling** and then **Cancelled** to complete the cancellation.

Use the AWS Proton AWS CLI to cancel an IN_PROGRESS service instance update deployment to a new minor version 2 as shown in the following commands and responses.

A wait condition is included in the template used for this example so that the cancellation starts before the update deployment succeeds.

Command: to cancel

```
aws proton cancel-service-instance-deployment --service-instance-name "instance-one" --
service-name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-service-instance --name "instance-one" --service-name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Update a service pipeline

There are four modes for updating a service pipeline as described in the following list. When using the AWS CLI, the `deployment-type` field defines the mode. When you use the console, these modes map to the **Edit pipeline** and **Update to recommended version**.

`NONE`

In this mode, a deployment *doesn't* occur. Only the requested metadata parameters are updated.

`CURRENT_VERSION`

In this mode, the service pipeline is deployed and updated with the new spec that you provide. Only requested parameters are updated. *Don't* include minor or major version parameters when you use this `deployment-type`.

`MINOR_VERSION`

In this mode, the service pipeline is deployed and updated with the published, recommended (latest) minor version of the current major version in use by default. You can also specify a different minor version of the current major version in use.

`MAJOR_VERSION`

In this mode, the service pipeline is deployed and updated with the published, recommended (latest) major and minor version of the current template by default. You can also specify a different major version that is higher than the major version in use and a minor version (optional).

You can attempt to cancel a service pipeline update deployment if the `deploymentStatus` is `IN_PROGRESS`. AWS Proton attempts to cancel the deployment. Successful cancellation isn't guaranteed.

When you cancel an update deployment, AWS Proton attempts to cancel the deployment as listed in the following steps.

- Sets the deployment state to `CANCELLING`.
- Stops the deployment in process and deletes any new resources that were created by the deployment when `IN_PROGRESS`.
- Sets the deployment state to `CANCELLED`.
- Reverts the state of the resource to what it was before the deployment was started.

For more information on cancelling a service pipeline deployment, see CancelServicePipelineDeployment in the *AWS Proton API Reference*.

Update a service pipeline using the console as described in the following steps.

1. In the AWS Proton console, choose **Services**.
2. In the list of services, choose the name of the service that you want to update the pipeline for.

3. There are two tabs on the service detail page, **Overview** and **Pipeline**. Choose **Pipeline**.
4. If you want to update specs, choose **Edit Pipeline** and fill out each form and choose **Next** until you complete the final form and then choose **Update pipeline**.

   If you want to update to a new version and there's an **information icon** that indicates a new version is available at **Pipeline template**, choose the name of the new template version.

   a. Choose **Update to recommended version**.
   b. Fill out each form and choose **Next** until you complete the final form and choose **Update**.

Use the AWS CLI to update a service pipeline to a new minor version as shown in the following example commands and responses.

Command: to update

```
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"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-service --name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Use the console to cancel a service pipeline deployment as shown in the following steps.

1. In the AWS Proton console, choose **Services** in the navigation pane.
2. In the list of services, choose the name of the service that has the pipeline with the deployment update that you want to cancel.
3. In the service detail page, choose the **Pipeline** tab.
4. If your update deployment status is **In progress**, in the service pipeline detail page, choose **Cancel deployment**.
5. A modal asks you to confirm the cancellation. Choose **Cancel deployment**.
6. Your update deployment status is set to **Cancelling** and then **Cancelled** to complete the cancellation.

Use the AWS Proton AWS CLI to cancel an IN_PROGRESS service pipeline update deployment to a new minor version 2 as shown in the following example commands and responses.

A wait condition is included in the template used for this example so that the cancellation starts before the update deployment succeeds.

Command: to cancel

```
aws proton cancel-service-pipeline-deployment --service-name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Command: to get and confirm status

```
aws proton get-service --name "simple-svc"
```

Response:

```
{
    "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"
    }
}
```

Monitoring AWS Proton

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Proton and your AWS solutions. The following section describes monitoring tools that you can use with AWS Proton.

Automate AWS Proton with EventBridge

You can monitor AWS Proton events in Amazon EventBridge. EventBridge delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services. You can configure events to respond to AWS resource state changes. EventBridge routes this data then to *target* services such as AWS Lambda and Amazon Simple Notification Service. These events are the same as those that appear in Amazon CloudWatch Events. CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. For more information, see What Is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

Use EventBridge to be notified of state changes in the AWS Proton provisioning workflows.

Event types

Events are composed of rules that include an event pattern and targets. You configure a rule by choosing event pattern and target objects:

Event pattern

Each rule is expressed as an event pattern with the source and type of events to monitor and the event targets. To monitor events, you create a rule with the service that you're monitoring as the event source. For example, you can create a rule with an event pattern that that uses AWS Proton as an event source to trigger a rule when there are changes in a deployment state.

Targets

The rule receives a selected service as the event target. You can set up a target service to send notifications, capture state information, take corrective action, initiate events, or take other actions.

Event objects contain standard fields of ID, account, AWS Region, detail-type, source, version, resource, time (optional). The detail field is a nested object containing custom fields for the event.

AWS Proton events are emitted on a best effort basis. Best effort delivery means that the service attempts to send all events to EventBridge, but in some rare cases an event might not be delivered.

The following table lists the detail-type values, status values, and detail fields for each AWS Proton resource that can emit events. When a resource is deleted, the `"status"` detail field value is `"DELETED"`.

Resource	**detail-type value**	**detail field: values**
Environment Template	"AWS Proton Environment Template Status Change"	"name": *`"myTemplate"`* "status": EnvironmentTemplate "previousStatus": EnvironmentTemplate

Resource	detail-type value	detail field: values
Environment Template Version	"AWS Proton Environment Template Version Status Change"	"name": *"myTemplate"* "majorVersion": *"1"* "minorVersion": *"0"* "status": EnvironmentTemplateVersion "previousStatus": EnvironmentTemplateVersion
Service Template	"AWS Proton Service Template Status Change"	"name": *"myTemplate"* "status": ServiceTemplate "previousStatus": ServiceTemplate
Service Template Version	"AWS Proton Service Template Version Status Change"	"name": *"myTemplate"* "majorVersion": *"1"* "minorVersion": *"0"* "status": ServiceTemplateVersion "previousStatus": ServiceTemplateVersion
Environment	"AWS Proton Environment Status Change"	"name": *"myEnvironment"* "status": Environment "previousStatus": Environment
Service	"AWS Proton Service Status Change"	"name": *"myService"* "status": Service "previousStatus": Service
Service Instance	"AWS Proton Service Instance Status Change"	"name": *"myServiceInstance"* "serviceName": *"myService"* "status": ServiceInstance "previousStatus": ServiceInstance
Service Pipeline	"AWS Proton Service Pipeline Status Change"	"serviceName": *"myService"* "status": ServicePipeline "previousStatus": ServicePipeline

Resource	detail-type value	detail field: values
Environment Account Connection	"AWS Proton Environment Account Connection Status Change"	"id": *"myEnvironmentAccountConnection"* "status": EnvironmentAccountConnection "previousStatus": EnvironmentAccountConnection

AWS Proton event examples

The following examples show the ways that AWS Proton can send events to EventBridge.

Service template

```
{
    "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"
    }
}
```

Service template version

```
{
    "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"
    }
}
```

Environment

```
{
    "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: Send Amazon Simple Notification Service alerts for AWS Proton service status changes

In this tutorial, you use an AWS Proton pre-configured *event rule* that captures status changes for your AWS Proton service. EventBridge sends the status changes to an Amazon SNS topic. You subscribe to the topic and Amazon SNS sends you status change emails for your AWS Proton service.

Prerequisites

You have an existing AWS Proton service with an `Active` status. As part of this tutorial, you can add service instances to this service, and then delete the instances.

If you need to create an AWS Proton service, see Getting started with AWS Proton. For more information, see Quotas and Edit a service.

Step 1: Create and subscribe to an Amazon SNS topic

Create an Amazon SNS topic to serve as an *event target* for the *event rule* that you create in Step 2.

Create an Amazon SNS topic

1. Log in and open the Amazon SNS console.
2. In the navigation pane, choose **Topics**, **Create topic**.
3. In **Create topic** page:
   a. Choose **Type Standard**.
   b. For **Name**, enter **`tutorial-service-status-change`** and choose **Create topic**.
4. In the **tutorial-service-status-change** detail page, choose **Create subscription**.
5. In the **Create subscription** page:
   a. For **Protocol**, choose **Email**.
   b. For **Endpoint**, enter an email address that you currently have access to and choose **Create subscription**.
6. Check your email account and wait to receive a subscription confirmation email message. When you receive it, open it and choose **Confirm subscription**.

Step 2: Register an event rule

Register an *event rule* that captures status changes for your AWS Proton service. For more information, see Prerequisites (p. 118).

Create an event rule.

1. Open the Amazon EventBridge console.

2. In the navigation pane, choose **Events**, **Rules**.
3. In the **Rules** page, in the **Rules** section, choose **Create rule**.
4. In the **Create rule** page:

   a. In the **Name and description** section, for **Name**, enter `tutorial-rule`.
   b. In the **Define pattern** section, choose **Event pattern**.

      i. For **Event matching pattern**, choose **Pre-defined by service**.
      ii. For **Service provider**, choose **AWS**.
      iii. For **Service name**, choose **Proton**.
      iv. For **Event type**, choose **AWS Proton Service Status Change**.

          The **Event pattern** appears in a text editor.
      v. Open the AWS Proton console.
      vi. In the navigation pane, choose **Services**.
      vii. In **Services** page, choose the name of your AWS Proton service.
      viii. In **Service details** page, copy the service Amazon Resource Name (ARN).
      ix. Navigate back to the *EventBridge console* and your tutorial rule and choose **Edit** at the text editor.
      x. In the text editor, for `"resources":`, enter the service ARN that you copied in step viii.

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

      xi. **Save** the event pattern.
   c. In the **Select targets** section:

      i. For **Target**, choose **SNS topic**.
      ii. For **Topic**, choose **tutorial-service-status-change**.
   d. Choose **Create**.

Step 3: Test your event rule

Verify that your *event rule* is working by adding an instance to your AWS Proton service.

1. Switch to the AWS Proton console.
2. In the navigation pane, choose **Services**.
3. In **Services** page, choose the name of your service.
4. In **Service details** page, choose **Edit**.
5. In **Configure service** page, choose **Next**.
6. In **Configure custom settings** page, in the **Service instances** section, choose **Add new instance**.
7. Complete the form for your **New instance**:

   a. Enter a **Name** for your new instance.
   b. Select the *same compatible environments* that you chose for your existing instances.
   c. Enter values for the required inputs.
   d. Choose **Next**.
8. Review your inputs and choose **Update**.

9. After the **Service status** is `Active`, check your email to verify you received AWS notifications that give status updates.

```
{
    "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"
    }
}
```

Step 4: Clean up

Delete your Amazon SNS topic and subscription and delete your EventBridge rule.

Delete your Amazon SNS topic and subscription.

1. Navigate to the Amazon SNS console.
2. In the navigation panel, choose **Subscriptions**.
3. In the **Subscriptions** page, choose the subscription that you made to the topic named `tutorial-service-status-change` and then choose **Delete**.
4. In the navigation panel, choose **Topics**.
5. In the **Topics** page, choose the topic named `tutorial-service-status-change` and then choose **Delete**.
6. A modal prompts you to verify the deletion. Follow the instructions and choose **Delete**.

Delete your EventBridge rule.

1. Navigate to the Amazon EventBridge console.
2. In the navigation pane, choose **Events**, **Rules**.
3. In the **Rules** page, choose the rule named `tutorial-rule` and then choose **Delete**.

4. A modal prompts you to verify the deletion. Choose **Delete**.

Delete the added service instance.

1. Navigate to the AWS Proton console.
2. In the navigation pane, choose **Services**.
3. In the **Services** page, choose the name of your service.
4. In the **Service** detail page, choose **Edit** and then **Next**.
5. In **Configure custom settings** page, in the **Service instances** section, choose **Delete** for the service instance that you created as part of this tutorial and then choose **Next**.
6. Review your inputs and choose **Update**.
7. A modal prompts you to verify the deletion. Follow the instructions and choose **Yes, delete**.

Security in AWS Proton

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Proton, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation is written to help you understand how to apply the shared responsibility model when using AWS Proton. The following topics show you how to configure AWS Proton to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Proton resources.

Topics

Identity and Access Management for AWS Proton

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Proton resources. IAM is an AWS service that you can use with no additional charge.

Topics

- AWS managed policies for AWS Proton (p. 138)
- Troubleshooting AWS Proton identity and access (p. 141)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Proton.

Service user – If you use the AWS Proton service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Proton features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS Proton, see Troubleshooting AWS Proton identity and access (p. 141).

Service administrator – If you're in charge of AWS Proton resources at your company, you probably have full access to AWS Proton. It's your job to determine which AWS Proton features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Proton, see How AWS Proton works with IAM (p. 126).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Proton. To view example AWS Proton identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Proton (p. 132).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Proton in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Proton works with IAM

Before you use IAM to manage access to AWS Proton, learn what IAM features are available to use with AWS Proton.

IAM features you can use with AWS Proton

IAM feature	AWS Proton support
Identity-based policies (p. 127)	Yes
Resource-based policies (p. 127)	No
Policy actions (p. 128)	Yes
Policy resources (p. 129)	Yes
Policy condition keys (p. 129)	Yes
ACLs (p. 130)	No
ABAC (tags in policies) (p. 130)	Yes
Temporary credentials (p. 130)	Yes
Principal permissions (p. 131)	Yes
Service roles (p. 131)	Yes
Service-linked roles (p. 131)	No

To get a high-level view of how AWS Proton and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for AWS Proton

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for AWS Proton

To view examples of AWS Proton identity-based policies, see Identity-based policy examples for AWS Proton (p. 132).

Resource-based policies within AWS Proton

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for AWS Proton

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of AWS Proton actions, see Actions defined by AWS Proton in the *Service Authorization Reference*.

Policy actions in AWS Proton use the following prefix before the action:

```
proton
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "proton:List*"
```

To view examples of AWS Proton identity-based policies, see Identity-based policy examples for AWS Proton (p. 132).

Policy resources for AWS Proton

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of AWS Proton resource types and their ARNs, see Resources defined by AWS Proton in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Proton.

To view examples of AWS Proton identity-based policies, see Identity-based policy examples for AWS Proton (p. 132).

Policy condition keys for AWS Proton

Supports policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of AWS Proton condition keys, see Condition keys for AWS Proton in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Proton.

To view an example condition-key-based policy for limiting access to a resource, see Condition-key based policy examples for AWS Proton (p. 132).

Access control lists (ACLs) in AWS Proton

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Access control lists (ACLs) are lists of grantees that you can attach to resources. They grant accounts permissions to access the resource to which they are attached.

Attribute-based access control (ABAC) with AWS Proton

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

For more information about tagging AWS Proton resources, see AWS Proton resources and tagging (p. 150).

Using Temporary credentials with AWS Proton

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for AWS Proton

Supports principal permissions	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Proton in the *Service Authorization Reference*.

Service roles for AWS Proton

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

For more information, see AWS Proton IAM service role policy examples (p. 134).

> **Warning**
> Changing the permissions for a service role might break AWS Proton functionality. Edit service roles only when AWS Proton provides guidance to do so.

Service-linked roles for AWS Proton

Supports service-linked roles	No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Policy examples for AWS Proton

Find AWS Proton IAM policy exampes in the following sections.

Topics

Identity-based policy examples for AWS Proton

By default, IAM users and roles don't have permission to create or modify AWS Proton resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Topics

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete AWS Proton resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using AWS Proton quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Links to Identity-based policy examples for AWS Proton

Links to example identity-based policy examples for AWS Proton

- AWS managed policies for AWS Proton (p. 138)
- AWS Proton IAM service role policy examples (p. 134)
- Condition-key based policy examples for AWS Proton (p. 132)

Condition-key based policy examples for AWS Proton

The following example IAM policy denies access to AWS Proton actions that match the templates specified in the `Condition` block. Note that these condition keys are only supported by the actions listed at Actions, resources, and condition keys for AWS Proton. To manage permissions on other actions, such as `DeleteEnvironmentTemplate`, you must use Resource-level access control.

Example policy that denies AWS Proton template actions on a specific templates:

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

In the next example policy, the first Resource-level statement denies access to AWS Proton template actions, other than `ListServiceTemplates`, that match the service template listed in the `Resource` block. The second statement denies access to AWS Proton actions that match the template listed in the `Condition` block.

Example policy that denies AWS Proton actions that match a specific template:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "arn:aws:region_id:123456789012:service-template/my-service-
template"
        },
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate": [
                        "arn:aws:proton:region_id:123456789012:service-template/my-service-
template"
                    ]
                }
            }
        }
    ]
}
```

The final policy example allows developer AWS Proton actions that match the specific service template listed in the `Condition` block.

Example policy to allow AWS Proton developer actions that match a specific template:

```
{
    "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 Proton IAM service role policy examples

Administrators own and manage the resources that AWS Proton creates as defined by the environment and service templates. They attach IAM service roles to their account that permit AWS Proton to create resources on their behalf. Administrators supply the IAM roles and AWS Key Management Service keys for resources that are later owned and managed by developers when AWS Proton deploys their application as an AWS Proton service in an AWS Proton environment. For more information about AWS KMS and data encryption, see Data protection in AWS Proton (p. 143).

A service role is an Amazon Web Services (IAM) role that allows AWS Proton to make calls to resources on your behalf. If you specify a service role, AWS Proton uses that role's credentials. Use a service role to explicitly specify the actions that AWS Proton can perform.

You create the service role and its permission policy with the IAM service. For more information about creating a service role, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

AWS Proton service role

As a member of the platform team, you can as an administrator create an AWS Proton service role to allow AWS Proton to make API calls to other services, like CloudFormation, on your behalf.

We recommend that you use the following IAM role and trust policy for your AWS Proton service role. When you use the AWS Proton console to create your roles, this is the AWS Proton service role policy that AWS Proton creates for you. When scoping down permission on this policy, keep in mind that AWS Proton fails on `Access Denied` errors.

> **Important**
> Be aware that the policies shown in the following examples grant administrator privileges to anyone that can register a template to your account. Because we don't know which resources you will define in your AWS Proton templates, these policies have broad permissions. We recommend that you scope down the permissions to the specific resources that will be deployed in your environments.

IAM AWS Proton service role policy

Replace *123456789012* with your AWS account ID.

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

IAM AWS Proton service trust policy

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

The following is an example of a scoped down AWS Proton service role policy that you can use if you only need AWS Proton services to provision S3 resources.

```
{
    "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 pipeline service role

As a member of the platform team, you, as an administrator, can create an AWS Proton pipeline service role to allow AWS Proton to make CloudFormation API calls to deploy a pipeline CloudFormation stack on your behalf.

We recommend that you use the following IAM role and trust policy for your AWS Proton pipeline service role. When you use the AWS Proton console to create your roles, this is the AWS Proton pipeline service role policy that AWS Proton creates for you. When scoping down permission on this policy, keep in mind that AWS Proton fails on `Access Denied` errors.

> **Important**
> Be aware that the policies shown in the following examples grant administrator privileges to anyone that can register a template to your account. Because we don't know which resources you will define in your AWS Proton templates, these policies have broad permissions. We recommend that you scope down the permissions to the specific resources that will be deployed in your pipelines.

IAM AWS Proton pipeline service role policy

Replace *123456789012* with your AWS account ID.

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

IAM AWS Proton pipeline service trust policy

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

To see an example of a scoped down policy, see AWS Proton service role (p. 135).

AWS managed policies for AWS Proton

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS Proton provides managed IAM policies and trust relationships that you can attach to IAM users, groups, or roles that allow differing levels of control over resources and API operations. You can apply these policies directly, or you can use them as starting points for creating your own policies.

The following trust relationship is used for each of the AWS Proton managed policies.

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

AWS managed policy: AWSProtonFullAccess

You can attach AWSProtonFullAccess to your IAM entities. AWS Proton also attaches this policy to a service role that allows AWS Proton to perform actions on your behalf.

This policy grants administrative permissions that allow full access to AWS Proton and limited access to AWS Key Management Service, IAM and AWS CodeStar connections services.

Permissions details

This policy includes the following permissions.

This managed policy provides administrative access to the AWS Proton APIs and AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "proton:*",
                "codestar-connections:ListConnections",
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "proton.*.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        }
    ]
}
```

- `proton` – Allows administrators full access to AWS Proton APIs.
- `iam` – Allows administrators to pass roles to AWS Proton. This is required so that AWS Proton can make API calls to other services on the administrator's behalf.
- `kms` – Allows administrators to add a grant to a customer managed key.
- `codestar-connections` – Allows administrators to list and pass codestar-connections so they can be used by AWS Proton.

AWS managed policy: AWSProtonReadOnlyAccess

You can attach AWSProtonReadOnlyAccess to your IAM entities. AWS Proton also attaches this policy to a service role that allows AWS Proton to perform actions on your behalf.

This policy grants read-only permissions that allow read only access to AWS Proton APIs.

Permissions details

This policy includes the following permissions.

This managed policy provides read only access to the AWS Proton APIs and AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "proton:List*",
            "proton:Get*"
        ],
        "Resource": "*"
    }
}
```

- `proton` – Allows contributors read-only access to AWS Proton APIs.

AWS Proton updates to AWS managed policies

View details about updates to AWS managed policies for AWS Proton since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS Proton Document history page.

Change	Description	Date
AWSProtonFullAccess (p. 139) – provides administrative access to the AWS Proton APIs and AWS Management Console.	AWS Proton added a new policy to provide administrative access to the AWS Proton APIs and AWS Management Console.	JUNE 09, 2021
AWSProtonReadOnlyAccess (p. 140) – added a new policy.	AWS Proton added a new policy to allow read-only access to AWS Proton APIs.	JUNE 09, 2021
AWS Proton started tracking changes.	AWS Proton started tracking changes for its AWS managed policies.	JUNE 09, 2021

Troubleshooting AWS Proton identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Proton and IAM.

Topics

I am not authorized to perform an action in AWS Proton

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `proton:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `proton:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS Proton.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Proton. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Important
Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access AWS Proton

To allow others to access AWS Proton, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS Proton.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my AWS Proton resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Proton supports these features, see How AWS Proton works with IAM (p. 126).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Configuration and vulnerability analysis in AWS Proton

AWS Proton does not provide patches or updates for customer provided code. Customers are responsible for updating and applying patches to their own code, including the source code for their services and applications that are running on AWS Proton and the code provided in their service and environment template bundles.

Customers are responsible for updating and patching infrastructure resources in their environments and services. AWS Proton will not automatically update or patch any resources. Customers should consult the documentation for the resources in their architecture to understand their respective patching policies.

Other than providing customer requested environment and service updates to minor versions of service and environment templates, AWS Proton does not provide patches or updates to the resources that customers define in their service and environment templates and template bundles.

For more details, see the following resources:

- Shared Responsibility Model
- Amazon Web Services: Overview of Security Processes

Data protection in AWS Proton

AWS Proton conforms to the AWS shared responsibility model which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS Proton or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into AWS Proton or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Server side encryption at rest

If you choose to encrypt sensitive data in your template bundles at rest in the S3 bucket where you store your template bundles, you must use an SSE-S3 or SSE-KMS key to allow AWS Proton to retrieve the template bundles so they can be attached to a registered AWS Proton template.

Encryption in transit

All service to service communication is encrypted in transit using SSL/TLS.

AWS Proton encryption key management

Within AWS Proton, all customer data is encrypted by default using an AWS Proton owned key. If you supply a customer owned and managed AWS KMS key, all customer data is encrypted using the customer provided key as described in the following paragraphs.

When you create an AWS Proton template, you specify your key and AWS Proton uses your credentials to create a grant which allows AWS Proton to use your key.

If you manually retire the grant or, disable or delete your specified key, then AWS Proton is unable to read the data that was encrypted by the specified key and throws `ValidationException`.

AWS Proton encryption context

AWS Proton supports encryption context headers. An encryption context is an optional set of key-value pairs that can contain additional contextual information about the data. For general information about encryption context, see AWS Key Management Service Concepts - Encryption Context in the *AWS Key Management Service Developer Guide*.

An encryption context is a set of key–value pairs that contain arbitrary non-secret data. When including an encryption context in a request to encrypt data, AWS KMS cryptographically binds the encryption context to the encrypted data. To decrypt the data, you must pass in the same encryption context.

Customers can use the encryption context to identify use of their customer managed key in audit records and logs. It also appears in plaintext in logs, such as AWS CloudTrail and Amazon CloudWatch Logs.

AWS Proton does not take in any customer-specified or externally-specified encryption context.

AWS Proton adds the following encryption context.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

The first encryption context identifies the AWS Proton template that the resource is associated with and also serves as a constraint for customer managed key permissions and grants.

The second encryption context identifies the AWS Proton resource that is encrypted.

The following examples show AWS Proton encryption context use.

Developer creating a service instance.

```
{
  "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"
}
```

An administrator creating a template.

```
{
  "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"
}
```

Infrastructure security in AWS Proton

As a managed service, AWS Proton is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS Proton through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

To improve network isolation, you can use AWS PrivateLink as described in the following section.

AWS Proton and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Proton by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access AWS Proton APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with AWS Proton APIs. Traffic between your VPC and AWS Proton does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Considerations for AWS Proton VPC endpoints

Before you set up an interface VPC endpoint for AWS Proton, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

AWS Proton supports making calls to all of its API actions from your VPC.

VPC endpoint policies are supported for AWS Proton. By default, full access to AWS Proton is allowed through the endpoint. For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Creating an interface VPC endpoint for AWS Proton

You can create a VPC endpoint for the AWS Proton service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create a VPC endpoint for AWS Proton using the following service name:

- com.amazonaws.*region*.proton

If you enable private DNS for the endpoint, you can make API requests to AWS Proton using its default DNS name for the Region, for example, `proton.region.amazonaws.com`.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for AWS Proton

You can attach an endpoint policy to your VPC endpoint that controls access to AWS Proton. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for AWS Proton actions

The following is an example of an endpoint policy for AWS Proton. When attached to an endpoint, this policy grants access to the listed AWS Proton actions for all principals on all resources.

```
{
  "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": "*"
```

```
        }
    ]
}
```

Logging and monitoring in AWS Proton

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Proton and your other AWS solutions. AWS provides the following monitoring tools to watch your instances running in AWS Proton, report when something is wrong, and take automatic actions when appropriate.

At this time, AWS Proton itself is not integrated with Amazon CloudWatch Logs or AWS Trusted Advisor. Administrators can configure and use CloudWatch to monitor other AWS services as defined in their service and environment templates. AWS Proton is integrated with AWS CloudTrail.

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.
- *Amazon EventBridge* is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, Software-as-a-Service (SaaS) applications, and AWS services and routes that data to targets such as Lambda. This enables you to monitor events that happen in services, and build event-driven architectures. For more information, see Automate AWS Proton with EventBridge (p. 115) and the EventBridge User Guide.

Resilience in AWS Proton

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS Proton offers features to help support your data resiliency and backup needs.

AWS Proton backups

AWS Proton maintains a backup of all customer data. In the case of a total outage, this backup can be used to restore AWS Proton and customer data from a previous valid state.

Security best practices for AWS Proton

AWS Proton provides security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

Topics

Use IAM to control access

IAM is an AWS service that you can use to manage users and their permissions in AWS. You can use IAM with AWS Proton to specify which AWS Proton actions administrators and developers can perform, such as managing templates, environments or services. You can use IAM service roles to allow AWS Proton to make calls to other services on your behalf.

For more information on AWS Proton and IAM roles, see Identity and Access Management for AWS Proton (p. 122).

Implement least privilege access. For more information, see Policies and permissions in IAM in the *AWS Identity and Access Management User Guide*.

Do not embed credentials in your templates and template bundles

Rather than embedding sensitive information in your AWS CloudFormation templates and template bundles, we recommend you use *dynamic references* in your stack template.

Dynamic references provide a compact, powerful way for you to reference external values that are stored and managed in other services, such as the AWS Systems Manager Parameter Store or AWS Secrets Manager. When you use a dynamic reference, CloudFormation retrieves the value of the specified reference when necessary during stack and change set operations, and passes the value to the appropriate resource. However, CloudFormation never stores the actual reference value. For more information, see Using Dynamic References to Specify Template Values in the *AWS CloudFormation User Guide*.

AWS Secrets Manager helps you to securely encrypt, store, and retrieve credentials for your databases and other services. The AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management.

For more information on defining template parameters, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html in the *AWS CloudFormation User Guide*.

Use encryption to protect sensitive data

Within AWS Proton, all customer data is encrypted by default using an AWS Proton owned key.

As a member of the platform team, you can provide a customer managed key to AWS Proton to encrypt and secure your sensitive data. Encrypt sensitive data at rest in your S3 bucket. For more information, see Data protection in AWS Proton (p. 143).

Use AWS CloudTrail to view and log API calls

AWS CloudTrail tracks anyone making API calls in your AWS account. API calls are logged whenever anyone uses the AWS Proton API, the AWS Proton console or AWS Proton AWS CLI commands. Enable logging and specify an Amazon S3 bucket to store the logs. That way, if you need to, you can audit who made what AWS Proton call in your account. For more information, see Logging and monitoring in AWS Proton (p. 147).

AWS Proton resources and tagging

AWS Proton resources that are assigned an Amazon Resource Name (ARN) include environment templates and their major and minor versions, service templates and their major and minor versions, environments, services and service instances. You can tag these resources to help you organize and identify them. You can use tags to categorize resources by purpose, owner, environment, or other criteria. For more information, see Tagging Strategies. To track and manage your AWS Proton resources, you can use the tagging features of as described in the following sections.

AWS tagging

You can assign metadata to your AWS resources in the form of tags. Each tag consists of a customer defined key and optional value. Tags can help you manage, identify, organize, search for, and filter resources.

Important
Do not add personally identifiable information (PII) or other confidential or sensitive information in tags. Tags are accessible to many AWS services, including billing. Tags are not intended to be used for private or sensitive data.

Each tag has two parts.

- A tag key (for example, `CostCenter`, `Environment`, or `Project`). Tag keys are case sensitive.
- A tag value (optional) (for example, `111122223333` or `Production`). Like tag keys, tag values are case sensitive.

The following basic naming and usage requirements apply to tags.

- Each resource can have a maximum of 50 user created tags.

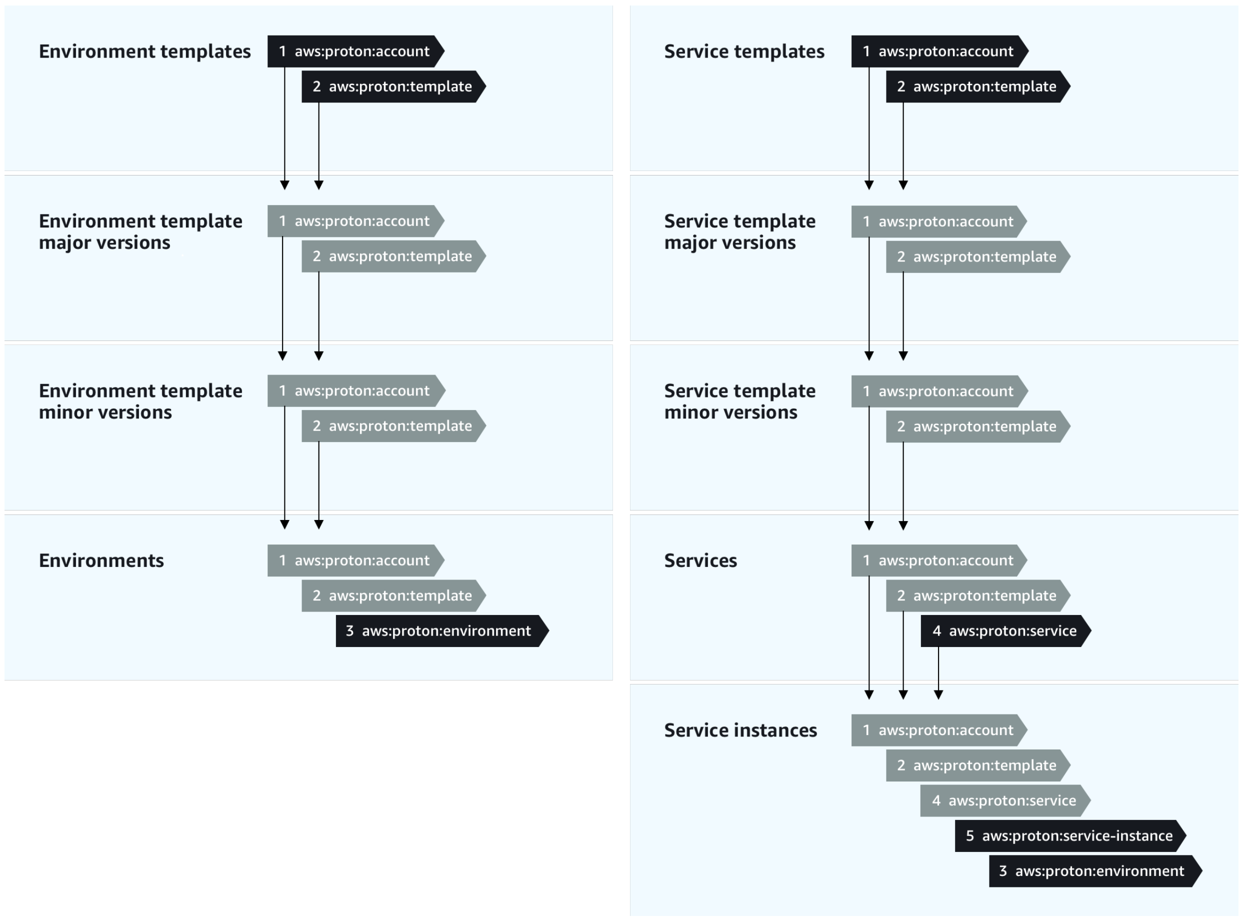
  Note
  System created tags that begin with the `aws:` prefix are reserved for AWS use, and do not count against this limit. You can't edit or delete a tag that begins with the `aws:` prefix.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The tag key must be a minimum of 1 and a maximum of 128 Unicode characters in UTF-8.
- The tag value must be a minimum of 1 and a maximum of 256 Unicode characters in UTF-8.
- Allowed characters in tags are letters, numbers, spaces representable in UTF-8, and the following characters:* _ . : / = + - @.

AWS Proton tagging

With AWS Proton, you can use both the tags that you create as well as the tags that AWS Proton automatically generates for you.

AWS Proton AWS managed tags

When you create an AWS Proton resource, AWS Proton automatically generates AWS managed tags for your new resource as shown in the following diagram. AWS managed tags propagate to the AWS Proton resources that your new resource creates or deploys.



If provisioned resources, such as those defined in service and environment templates, support AWS tagging, the AWS managed tags propagate as customer managed tags to provisioned resources. These tags won't propagate to a provisioned resource that doesn't support AWS tagging.

AWS Proton applies tags to your resources by AWS Proton accounts, registered templates and deployed environments, as well as services and service instances as described in the following table. You can use AWS managed tags to view and manage your AWS Proton resources, but you can't modify them.

AWS managed tag key	Propagated customer managed key	Description
`aws:proton:account`	`proton:account`	The ARN of the account that creates and deploys AWS Proton resources.
`aws:proton:template`	`proton:template`	The ARN of a selected template.
`aws:proton:service`	`proton:service`	The ARN of a selected service.
`aws:proton:service-instance`	`proton:service-instance`	The ARN of a selected service instance.
`aws:proton:environment`	`proton:environment`	The ARN of a selected environment.

The following is an example of an AWS managed tag for an AWS Proton resource.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-
template"
```

The following is an example of a customer managed tag applied to a provisioned resource that was propagated from an AWS managed tag.

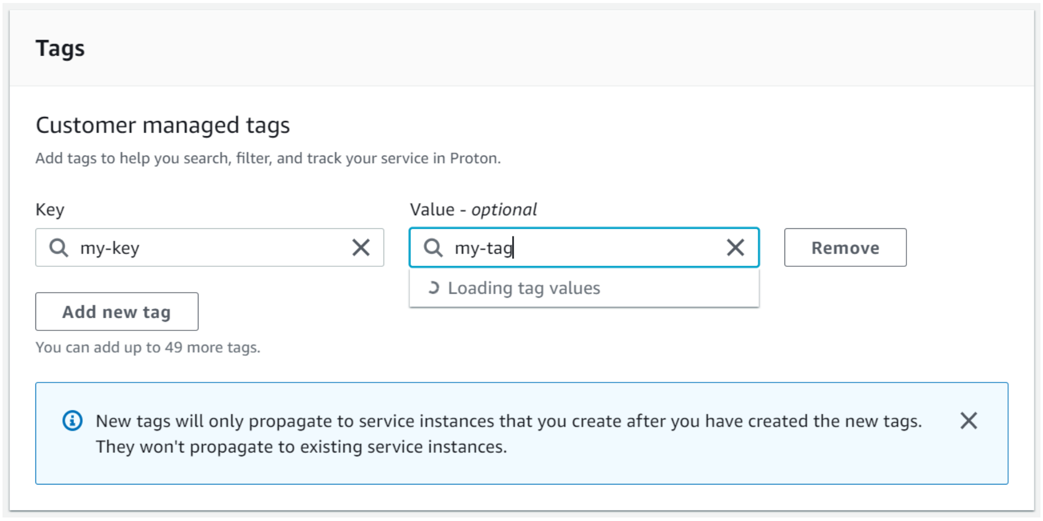
```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

Customer managed tags

Each AWS Proton resource has a maximum quota of 50 customer managed tags. Customer managed tags propagate to child AWS Proton resources in the same way that AWS managed tags do, except they don't propagate to existing AWS Proton resources or to provisioned resources. If you apply a new tag to an AWS Proton resource with existing child resources and you want the existing child resources to be tagged with the new tag, you need to tag each existing child resource manually, using the console or AWS CLI.

Create tags using the console

When you create an AWS Proton resource using the console, you're given the opportunity to create customer managed tags either on the first or second page of the create procedure as shown in the following console snapshot. choose **Add new tag**, enter the key and value and proceed.


Tags

Customer managed tags

Add tags to help you search, filter, and track your service in Proton.

Key: my-key

Value - *optional*: my-tag

Loading tag values

You can add up to 49 more tags.

New tags will only propagate to service instances that you create after you have created the new tags. They won't propagate to existing service instances.

After you create a new resource using the AWS Proton console, you can view its list of AWS managed and customer managed tags from the detail page.

Create or edit a tag

1. In the AWS Proton console, open an AWS Proton resource detail page where you can see a list of tags.
2. Choose **Manage tags**.
3. In the **Manage tags** page, you can view, create, remove and edit tags. You can't modify the AWS managed tags listed at the top. However, you can add to and modify the customer managed tags with editing fields, listed after the AWS managed tags.

   Choose **Add new tag** to create a new tag.

4. Enter a key and value for the new tag.
5. To edit a tag, enter a value in the tag value field for a selected key.
6. To delete a tag choose **Remove** for a selected tag.
7. When you have completed your changes, choose **Save changes**.

Create tags using the AWS Proton AWS CLI

You can view, create, remove and edit tags using the AWS Proton AWS CLI.

You can create or edit a tag for a resource as shown in the following example.

```
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"}]'
```

You can remove a tag for a resource as shown in the next example.

```
aws proton untag-resource --resource-arn "arn:aws:proton:region-id:account-id:service-
template/webservice" --tag-keys '["mykey1","mykey2"]'
```

You can list tags for a resource as shown in the final example.

```
aws proton list-tags-for-resource --resource-arn "arn:aws:proton:region-id:account-
id:service-template/webservice"
```

AWS Proton quotas

The following table lists AWS Proton quotas.

Resource	**Default limit**
Maximum size of template bundle	5 MB
Maximum size of template manifest file	50 KB
Maximum size of template schema file	50 KB
Maximum size of each template file other than those files with lower limits	200 KB
Maximum length of each template name	100 characters
Maximum number of template files per bundle	10
Maximum number of registered templates per account, service and environment templates combined	100
Maximum number of template versions registered per template	500
Maximum number of environments per account	100
Maximum number of services per account	1000
Maximum number of service instances per service	100
Maximum number of environment account connections per environment account	5

Document history

The following table describes the important changes to the documentation related to the latest release of AWS Proton and customer feedback. For notification about updates to this documentation, you can subscribe to an RSS feed.

- **API version: 2020-07-20**

update-history-change	update-history-description	update-history-date
Documentation updates (p. 155)	Added EventBridge tutorial, Getting started workflow, How AWS Proton works, and Template bundle section enhancements.	September 17, 2021
AWS Proton console help panels release (p. 155)	Help panels added to the console. Console template version delete no longer deletes lower versions. The API migration guide has been removed.	September 8, 2021
AWS Proton General availability (GA) release (p. 155)	Adds cross account environments, EventBridge monitoring, IAM condition keys, idempotency support, and increased quotas.	June 9, 2021
Add and delete service instances for a service and use existing external infrastructure for environments with AWS Proton (p. 155)	This public preview release includes updates that make it possible for you to add and delete service instances from a service, to use your existing external infrastructure in an AWS Proton environment and to cancel environment, service instance and pipeline deployments. AWS Proton now supports PrivateLink. An additional deletion validation has been added to prevent a minor version from being mistakenly deleted while a resource is using it.	April 27, 2021
Tagging with AWS Proton (p. 155)	Public preview release 2 includes AWS Proton tagging and the ability to launch services without a service pipeline.	March 5, 2021
Initial release (p. 155)	Public preview release is now available in selected regions.	December 1, 2020

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.