aws

Guía del usuario

AWS Proton

AWS Proton: Guía del usuario

Table of Contents

¿Qué es AWS Proton?

AWS Proton es:

- Infraestructura automatizada como aprovisionamiento de código e implementación de aplicaciones basadas en contenedores y sin servidor

  El servicio de AWS Proton es un marco de automatización doble. Como administrador, el usuario crea plantillas de servicio versionadas que definen la infraestructura estandarizada y las herramientas de implementación para aplicaciones sin servidor y basadas en contenedores. Como desarrollador de aplicaciones, el usuario puede seleccionar entre las plantillas de servicio disponibles para automatizar la implementación de aplicaciones o servicios.

  AWS Proton identifica todas las instancias de servicio existentes que utilizan una versión de plantilla desactualizada. Como administrador, el usuario puede solicitar que AWS Proton las actualice con un solo clic.
- Infraestructura estandarizada

  Los equipos de plataformas pueden utilizar AWS Proton y una infraestructura versionada como plantillas de código. Además pueden utilizar estas plantillas para definir y administrar las pilas de aplicaciones estándar que contengan la arquitectura, los recursos de infraestructura y la canalización de la implementación del software de CI/CD.
- Implementaciones integradas con CI/CD

  Cuando los desarrolladores utilizan la interfaz de autoservicio de AWS Proton para seleccionar una plantilla de servicio, eligen una definición de pila de aplicaciones estandarizada para sus implementaciones de código. AWS Proton aprovisiona automáticamente los recursos, configura la canalización de CI/CD e implementa el código en la infraestructura definida.

AWS Proton para equipos de plataformas

Como administrador, tanto el usuario como los miembros de su equipo de plataforma crean plantillas de entorno y plantillas de servicio que contengan infraestructura como código. La plantilla de entorno define la infraestructura compartida que utilizan varias aplicaciones o recursos. La plantilla de servicio define el tipo de infraestructura que se necesita para implementar y mantener una sola aplicación o microservicio en un entorno. Un servicio de AWS Proton es una instancia de una plantilla de servicio, que normalmente incluye varias instancias de servicio y una canalización. Una instancia

de servicio de AWS Proton es una instancia de una plantilla de servicio en un entorno específico. Tanto el usuario como los miembros de su equipo pueden especificar qué plantillas de entorno son compatibles con una plantilla de servicio determinada. Para obtener más información acerca de las plantillas, consulte AWS ProtonPlantillas de .

Puede utilizar la siguiente infraestructura como proveedores de código con AWS Proton:

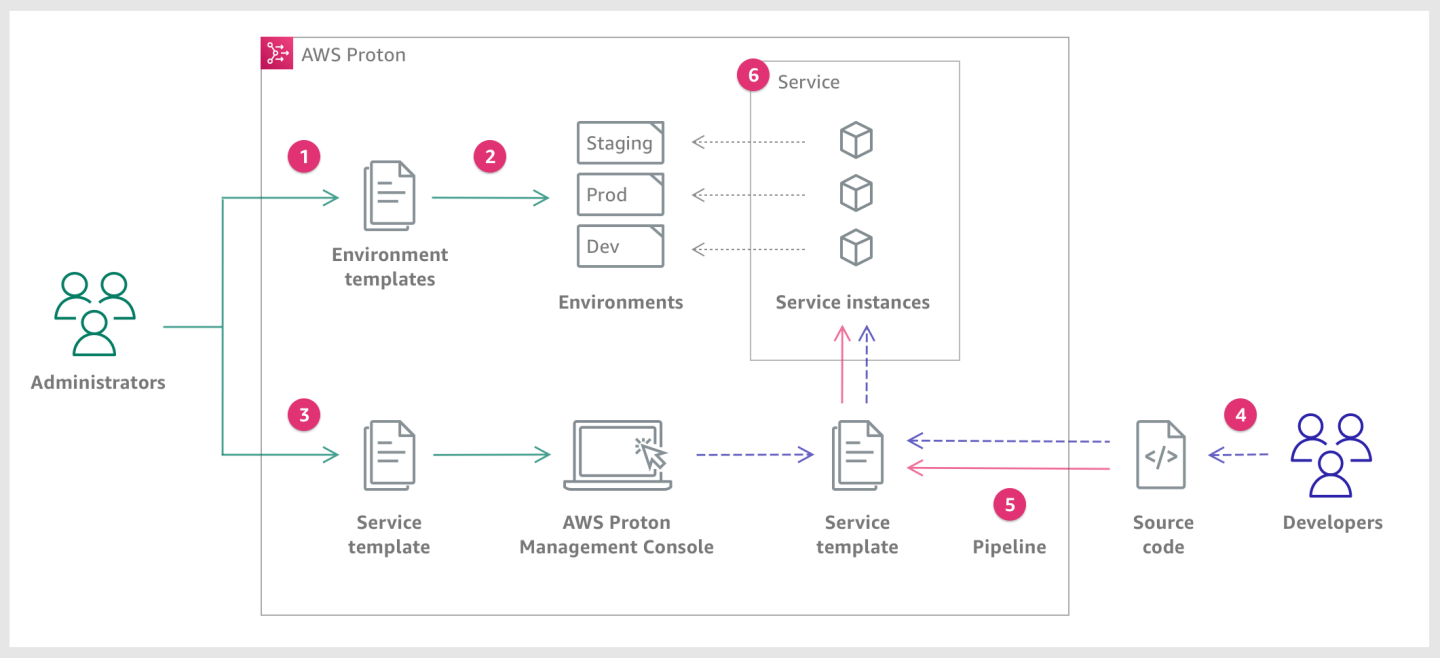
- AWS CloudFormation
- Terraform

AWS Proton para desarrolladores

Como desarrollador de aplicaciones, seleccione una plantilla de servicio estandarizada que AWS Proton utilice para crear un servicio que implemente y administre su aplicación en una instancia de servicio. Un servicio de AWS Proton es una instancia de una plantilla de servicio, que normalmente incluye varias instancias de servicio y una canalización.

Flujo de trabajo de AWS Proton

El siguiente diagrama es una visualización de los principales conceptos de AWS Proton explicados en el párrafo anterior. También brinda información general de lo que constituye un flujo de trabajo simple de AWS Proton.



1. Como administrador, debe crear y registrar una plantilla de entorno con AWS Proton, que define los recursos compartidos.

2. AWS Proton implementa uno o más entornos, en función de una plantilla de entorno.

3. Como administrador, debe crear y registrar una plantilla de servicio con AWS Proton, que define la infraestructura, la supervisión y los recursos de CI/CD relacionados, así como las plantillas de entorno compatibles.

Como desarrollador de

4. , seleccione una plantilla de servicio registrada y proporcione un enlace al repositorio de código fuente.

5. AWS Proton aprovisiona el servicio con una canalización de CI/CD para las instancias de servicio.

6. AWS Proton aprovisiona y administra el servicio y las instancias de servicio que ejecutan el código fuente, tal como se definió en la plantilla de servicio seleccionada. Una instancia de servicio es una instancia de la plantilla de servicio seleccionada en un entorno para una sola etapa de una canalización (como por ejemplo, Prod).

Configuración

Complete las tareas de esta sección para poder crear y registrar plantillas de servicio y de entorno. Los necesita para implementar entornos y servicios con ellos AWS Proton.

> **Note**
>
> Los ofrecemos sin AWS Proton coste adicional. Puede crear, registrar y mantener plantillas de servicio y de entorno sin costo alguno. También puede confiar en AWS Proton que autogestionará sus propias operaciones, como el almacenamiento, la seguridad y la implementación. Los únicos gastos en los que incurre durante el uso AWS Proton son los siguientes.
>
> - Costos de implementación y uso de Nube de AWS los recursos que usted indicó que AWS Proton implementaran y mantuvieran por usted.
> - Costos de mantener una AWS CodeStar conexión a su repositorio de código.
> - Costos de mantenimiento de un bucket de Amazon S3, en el caso de que el usuario utilice un bucket para proporcionar entradas a AWS Proton. El usuario puede evitar estos costos si cambia a the section called “Configuraciones de sincronización de plantillas” mediante los repositorios de Git para su the section called “Paquetes de plantillas”.

Temas

Configuración con IAM

Cuando te registras AWS, Cuenta de AWS se suscribe automáticamente a todos los servicios de AWS, incluidos AWS Proton. Solo se le cobrará por los servicios y recursos que utilice.

> **Note**
>
> Tanto el usuario como su equipo, incluidos los administradores y los desarrolladores, deben tener la misma cuenta.

Inscríbase en AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirse a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en una Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea una. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como una práctica recomendada de seguridad, asigne acceso administrativo a un usuario y solo utilice el usuario raíz para realizar tareas que requieran acceso de usuario raíz.

Creación un usuario de IAM

Para crear un usuario administrador, elija una de las siguientes opciones.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
En IAM Identity Center (recomendado)	Usar credenciales a corto plazo para acceder a AWS. Esto se ajusta a las prácticas recomendadas de seguridad. Para	Siga las instrucciones en Introducción en la Guía del usuario de AWS IAM Identity Center .	Configure el acceso programático configurando el AWS CLI que se utilizará AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
	obtener información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.		
En IAM (no recomendado)	Usar credenciales a largo plazo para acceder a AWS.	Siga las instrucciones en Creación del primer grupo de usuarios y usuario de administrador de IAM en la Guía del usuario de IAM.	Configurar el acceso programático mediante Administración de las claves de acceso de los usuarios de IAM en la Guía del usuario de IAM.

Configuración de funciones de AWS Proton servicio

Puede que desee crear algunas funciones de IAM para distintas partes de la AWS Proton solución. Puede crearlos por adelantado mediante la consola de IAM, o puede usar la AWS Proton consola para crearlos por usted.

Cree funciones de AWS Proton entorno que le permitan AWS Proton realizar llamadas a la API a otros Servicios de AWS servicios informáticos y de almacenamiento AWS CloudFormation AWS CodeBuild, por ejemplo, en su nombre, a fin de proporcionarle recursos. Se requiere un rol de aprovisionamiento administrado por AWS cuando un entorno o cualquiera de las instancias de servicio que se ejecuten en él utilicen el aprovisionamiento administrado por AWS. Se requiere un CodeBuildrol cuando un entorno o cualquiera de sus instancias de servicio utilizan el

CodeBuildaprovisionamiento. Para obtener más información sobre las funciones del AWS Proton entorno, consultethe section called “Roles de IAM”. Al crear un entorno, puede usar la AWS Proton consola para elegir un rol existente para cualquiera de estos dos roles o para crear un rol con privilegios administrativos para usted.

Del mismo modo, cree funciones de AWS Proton canalización AWS Proton para poder realizar llamadas a la API a otros servicios en su nombre a fin de proporcionarle una canalización de CI/CD. Para obtener más información sobre las funciones de AWS Proton canalización, consulte. the section called “Roles de servicio para canalizaciones” Para obtener más información acerca de la configuración de CI/CD, consulte the section called “Configuración de canalizaciones de CI/CD de la cuenta”.

> **Note**
>
> Como no sabemos qué recursos definirá en sus AWS Proton plantillas, las funciones que cree mediante la consola tienen amplios permisos y se pueden utilizar tanto como funciones de servicio de AWS Proton canalización como de AWS Proton servicio. Para las implementaciones de producción, te recomendamos que limites los permisos para los recursos específicos que se van a implementar mediante la creación de políticas personalizadas tanto para las funciones de servicio de AWS Proton canalización como para las funciones de servicio de AWS Proton entorno. Puede crear y personalizar estas funciones mediante el uso de AWS CLI o IAM. Para obtener más información, consulte Funciones de servicio para AWS Proton y Crear un servicio.

Configurando con AWS Proton

Si desea utilizarla AWS CLI para ejecutar AWS Proton las API, compruebe que la ha instalado. Si todavía no la ha instalado, consulte Configurar la AWS CLI.

AWS Proton configuración específica:

- Para crear y administrar plantillas:
  - Si utiliza configuraciones de sincronización de plantillas, configure una conexión de AWS CodeStar.
  - De lo contrario, configure un bucket de Amazon S3.
- Para aprovisionar la infraestructura:
  - Para el aprovisionamiento autoadministrado, debe configurar una conexión de AWS CodeStar.

- (Opcional) Para aprovisionar canalizaciones:
  - Para el AWS aprovisionamiento gestionado y el aprovisionamiento CodeBuild basado, configure las funciones de canalización.
  - Para el aprovisionamiento autoadministrado, configure un repositorio de canalizaciones.

Para obtener más información acerca de los métodos de aprovisionamiento, consulte the section called “Aprovisionamiento administrado por AWS”.

Configuración de un bucket de Amazon S3

Para configurar un bucket de S3, siga las instrucciones que se indican en Cómo crear su primer bucket de S3. Coloca tus entradas AWS Proton en el depósito donde AWS Proton puedas recuperarlas. Estas entradas se conocen como paquetes de plantillas. Puede obtener más información sobre las mismas en otras secciones de esta guía.

Configurar una AWS CodeStar conexión

Para conectarte AWS Proton a un repositorio, debes crear una AWS CodeStar conexión que active una canalización cuando se realice una nueva confirmación en un repositorio de código fuente de terceros.

AWS Proton usa la conexión para:

- Activar una canalización de servicios cuando se realice una nueva confirmación en el código fuente del repositorio del usuario.
- Realizar una solicitud de extracción en una infraestructura como repositorio de código.
- Crear una nueva versión secundaria o principal de la plantilla cada vez que se inserte una confirmación a un repositorio de plantillas que modifique alguna de las plantillas del usuario, siempre y cuando la versión aún no exista.

Puedes conectarte a los repositorios de Bitbucket GitHub, GitHub Enterprise y GitHub Enterprise Server con. CodeConnections Para obtener más información, consulta la CodeConnectionsGuía del AWS CodePipeline usuario.

Para configurar una CodeStar conexión.

1. Abra la consola de AWS Proton.

2. En el panel de navegación, seleccione Configuración y, a continuación, Conexiones de repositorio para ir a la página de Conexiones en la Configuración de las herramientas para desarrolladores. La página muestra una lista de conexiones.
3. Seleccione Crear conexión y siga las instrucciones.

Configuración de canalizaciones de CI/CD de la cuenta

AWS Proton puede aprovisionar canalizaciones de CI/CD para implementar el código de la aplicación en sus instancias de servicio. La AWS Proton configuración que necesita para el aprovisionamiento de la canalización depende del método de aprovisionamiento que elija para la canalización.

AWS-aprovisionamiento gestionado y CodeBuild basado: configura las funciones de canalización

Con el AWS aprovisionamiento y el aprovisionamiento gestionados, aprovisiona las canalizaciones por CodeBuild usted. AWS Proton Por lo tanto, AWS Proton necesita un rol de servicio que proporcione permisos para el aprovisionamiento de canalizaciones. Cada uno de estos dos métodos de aprovisionamiento utiliza su propio rol de servicio. Estas funciones se comparten en todas las canalizaciones AWS Proton de servicios y se configuran una vez en la configuración de la cuenta.

Para crear roles de servicio de canalización mediante la consola

1. Abra la consola de AWS Proton.
2. En el panel de navegación, elija Configuración y, a continuación, elija Configuración de la cuenta.
3. En la página Configuración de CI/CD de la cuenta, seleccione Configurar.
4. Realice una de las siguientes acciones siguientes:
   - Para AWS Proton crear un rol de servicio de canalización para ti

     [Para habilitar el aprovisionamiento de canalizaciones administrado por AWS] En la página Configurar los ajustes de la cuenta, en la sección de roles de canalización de aprovisionamiento administrado por AWS:

     a. Seleccione Nuevo rol de servicio.
     b. Escriba un nombre para el rol, como por ejemplo, **`myProtonPipelineServiceRole`**.
     c. Marque la casilla de verificación para aceptar la creación de un AWS Proton rol con privilegios administrativos en su cuenta.

[Para habilitar el aprovisionamiento CodeBuild basado de canalizaciones] En la página Configurar los ajustes de la cuenta, en la sección Función de CodeBuild canalización, selecciona la función de servicio existente y elige la función de servicio que creaste en la sección Función de CloudFormation canalización. O bien, si no asignaste un rol de CloudFormation canalización, repite los tres pasos anteriores para crear un nuevo rol de servicio.

- Para elegir los roles de servicio de canalización existentes

  [Para habilitar el aprovisionamiento de canalizaciones administradas por AWS] En la página Configurar los ajustes de la cuenta, en la sección Rol de canalización de aprovisionamiento administrada por AWS, seleccione Rol de servicio existente y elija un rol de servicio en su cuenta de AWS .

  [Para habilitar el CodeBuild aprovisionamiento de canalizaciones] En la página Configurar los ajustes de la cuenta, en la sección Función de aprovisionamiento de CodeBuild canalizaciones, selecciona la función de servicio existente y elige una función de servicio en tu cuenta. AWS

5. Elija Guardar cambios.

   Su nuevo rol de servicio de canalización se muestra en la página Configuración de la cuenta.

Aprovisionamiento autoadministrado: configuración de un repositorio de canalizaciones

Con el aprovisionamiento autogestionado, AWS Proton envía una solicitud de incorporación de cambios (PR) a un repositorio de aprovisionamiento que hayas configurado y tu código de automatización se encarga de aprovisionar las canalizaciones. Por lo tanto, AWS Proton no necesita una función de servicio para aprovisionar las canalizaciones. En su lugar, necesita un repositorio de aprovisionamiento registrado. El código de automatización del repositorio debe asumir un rol adecuado que proporcione permisos para el aprovisionamiento de canalizaciones.

Para registrar un repositorio de aprovisionamiento de canalizaciones mediante la consola

1. Cree un repositorio de aprovisionamiento de canalizaciones de CI/CD si aún no lo ha creado. Para obtener más información sobre las canalizaciones del aprovisionamiento autoadministrado, consulte the section called “Aprovisionamiento autoadministrado”.

2. En el panel de navegación, elija Configuración y, a continuación, elija Configuración de la cuenta.
3. En la página Configuración de CI/CD de la cuenta, seleccione Configurar.
4. En la página Configurar los ajustes de la cuenta, en la sección Repositorio de canalización de CI/CD:

   a. Seleccione Nuevo repositorio y, a continuación, elija uno de los proveedores de repositorios.

   b. Para la CodeStar conexión, elige una de tus conexiones.

   > **Note**
   >
   > Si aún no tienes una conexión con la cuenta del proveedor de repositorios correspondiente, selecciona Añadir una nueva CodeStar conexión, completa el proceso de creación de la conexión y, a continuación, pulsa el botón de actualización situado junto al menú de CodeStarconexiones. Ahora debería poder elegir su nueva conexión en el menú.

   c. En Nombre del repositorio, elija el repositorio de aprovisionamiento de canalizaciones. En el menú desplegable se mostrará la lista de repositorios de la cuenta del proveedor.

   d. En Nombre de la ramificación, elija alguna de las ramificaciones del repositorio.
5. Elija Guardar cambios.

   El repositorio de canalizaciones se mostrará en la página Configuración de la cuenta.

Configurar la AWS CLI

Para utilizar el AWS CLI para realizar llamadas a la AWS Proton API, compruebe que ha instalado la última versión del AWS CLI. Para obtener más información, consulte Cómo empezar a trabajar con AWS CLI en la Guía del usuario de AWS Command Line Interface . A continuación, para empezar a utilizar el AWS CLI with AWS Proton, consultethe section called “Introducción a la CLI”.

Introducción a AWS Proton

Antes de empezar, prepárese para utilizar AWS Proton y compruebe que cumple los requisitos previos de introducción.

Para empezar con AWS Proton, elija una o más de las siguientes rutas:

- Siga un ejemplo guiado de flujo de trabajo de la consola o la CLI a través de los enlaces a la documentación.
- Siga un ejemplo guiado de flujo de trabajo de la consola.
- Siga un ejemplo guiado de flujo de trabajo de la AWS CLI.

Temas

Requisitos previos

Antes de empezar a utilizar AWS Proton, asegúrese de que se cumplen los siguientes requisitos previos: Para obtener más información, consulte Configuración.

- Debe tener una cuenta de IAM con permisos de administrador. Para obtener más información, consulte Configuración con IAM.
- El rol de servicio de AWS Proton y el rol de servicio de canalización de AWS Proton deben estar asociados a su cuenta. Para obtener más información, consulte Configuración de funciones de AWS Proton servicio y Funciones de servicio para AWS Proton.
- Debe tener una conexión de AWS CodeStar. Para obtener más información, consulte Configurar una AWS CodeStar conexión.
- Debe estar familiarizado con la creación de plantillas de AWS CloudFormation y la parametrización de Jinja. Para obtener más información, consulte ¿Qué es AWS CloudFormation? en la Guía del usuario de AWS CloudFormation y el sitio web de Jinja.

- Debe tener conocimientos básicos de los servicios de infraestructura de AWS.
- Debe haber iniciado sesión en su Cuenta de AWS.

Flujo de trabajo de introducción

Obtenga información sobre cómo crear paquetes de plantillas, crear y registrar plantillas y crear entornos y servicios siguiendo los pasos y enlaces de ejemplo.

Antes de empezar, compruebe que haya creado un rol de servicio de AWS Proton.

Si la plantilla de servicio incluye una canalización de servicios de AWS Proton, compruebe que haya creado una conexión de AWS CodeStar y un rol de servicio de canalización de AWS Proton.

Para obtener más información, consulte la Referencia de la API del servicio de AWS Proton.

Ejemplo: Introducción al flujo de trabajo

1. Consulte el diagrama en Cómo funciona AWS Proton para obtener una vista general de las entradas y salidas de AWS Proton.
2. Cree un paquete de entorno y un paquete de plantillas de servicio.
    a. Identifique los parámetros de entrada.
    b. Cree un archivo de esquema.
    c. Cree archivos de Infraestructura como código (IaC).
    d. Para completar el paquete de plantillas, cree un archivo de manifiesto y organice los archivos de IaC, los archivos de manifiesto y el archivo de esquema en directorios.
    e. Haga que su paquete de plantillas pueda acceder a AWS Proton.
3. Cree y registre una versión de plantilla de entorno con AWS Proton.

    Al utilizar la consola para crear y registrar una plantilla, se crea automáticamente una versión de la plantilla.

    Al utilizar la AWS CLI para crear y registrar una plantilla:

    a. Cree una plantilla de entorno.
    b. Cree una versión de plantilla de entorno.

Para obtener más información, consulte CreateEnvironmentTemplate y CreateEnvironmentTemplateVersion en la Referencia de la API de AWS Proton.

4. Publique la plantilla de entorno para que esté disponible para su uso.

   Para obtener más información, consulte UpdateEnvironmentTemplateVersion en la Referencia de la API de AWS Proton.

5. Para crear un entorno, seleccione una versión de plantilla de entorno publicada y proporcione los valores de las entradas necesarias.

   Para obtener más información, consulte CreateEnvironment en la Referencia de la API de AWS Proton.

6. Cree y registre una versión de plantilla de servicio con AWS Proton.

   Al utilizar la consola para crear y registrar una plantilla, se crea automáticamente una versión de la plantilla.

   Al utilizar la AWS CLI para crear y registrar una plantilla:

   a. Cree una plantilla de servicio.
   b. Cree una versión de plantilla de servicio.

   Para obtener más información, consulte CreateServiceTemplate and CreateServiceTemplateVersion en la Referencia de la API de AWS Proton.

7. Publique la plantilla de servicio para que esté disponible para su uso.

   Para obtener más información, consulte UpdateServiceTemplateVersion en la Referencia de la API de AWS Proton.

8. Para crear un servicio, seleccione una versión de plantilla de servicio publicada y proporcione los valores de las entradas necesarias.

   Para obtener más información, consulte CreateService en la Referencia de la API de AWS Proton.

Introducción al AWS Management Console

Introducción a AWS Proton

- Cree y visualice una plantilla de entorno.
- Cree, visualice y publique una plantilla de servicio que utilice la plantilla de entorno que acaba de crear.
- Cree un entorno y un servicio (opcional).
- Elimine la plantilla de servicio, la plantilla de entorno, el entorno y el servicio, si se crearon.

Paso 1: Abrir la consola de AWS Proton

- Abra la consola de AWS Proton.

Paso 2: Prepararse para utilizar las plantillas de ejemplo

1. Cree una conexión de Codestar a Github y póngale a la conexión el nombre my-proton-connection.
2. Vaya a https://github.com/aws-samples/aws-proton-cloudformation-sample-templates
3. Cree una bifurcación del repositorio en su cuenta de Github.

Paso 3: Crear una plantilla de entorno

En el panel de navegación, elija Plantillas de entorno.

1. En la página Plantillas de entorno, seleccione Crear plantilla de entorno.
2. En la página Crear plantilla de entorno, en la sección Opciones de plantilla, seleccione Crear una plantilla para aprovisionar nuevos entornos.
3. En la sección Origen del paquete de plantillas, seleccione Sincronización de un paquete de plantillas desde Git.
4. En la sección Repositorio de definiciones de plantillas, seleccione Elegir un repositorio de Git vinculado.
5. Seleccione my-proton-connection de la Lista de repositorios.
6. Seleccione main en la Lista de ramificaciones.

7. En la sección de Detalles de la plantilla del entorno de Proton.

   a. Introduzca el nombre de la plantilla como **`fargate-env`**.

   b. Introduzca el nombre para mostrar de la plantilla de entorno como **`My Fargate Environment`**.

   c. (Opcional) Escriba una descripción para la plantilla de entorno.

8. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.

9. Seleccione Crear plantilla de entorno.

   Ahora aparecerá una nueva página que muestra el estado y los detalles de su nueva plantilla de entorno. Estos detalles incluyen una lista de etiquetas administradas por el cliente y por AWS. AWS Proton genera automáticamente etiquetas administradas por AWS cuando se crean recursos de AWS Proton. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

10. El estado de una nueva plantilla de entorno comienza en el estado Borrador. Usted y las personas que gocen de permisos `proton:CreateEnvironment` podrán verla y acceder a ella. Siga el siguiente paso para poner la plantilla a disposición de otras personas.

11. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria de la plantilla que acaba de crear (1.0). Como alternativa, puede elegir Publicar en el banner de alerta de información y omitir el paso siguiente.

12. En la sección Versiones de la plantilla, seleccione Publicar.

13. El estado de la plantilla cambiará a Publicado. Como esta es la versión más reciente de la plantilla, pasará a ser la versión Recomendada.

14. En el panel de navegación, seleccione Plantillas de entorno.

    Una nueva página mostrará una lista de las plantillas de entorno del usuario junto con los detalles de cada plantilla.

Paso 4: Crear una plantilla de servicio

Cree una plantilla de servicio.

1. En el panel de navegación, elija Plantillas de servicio.

2. En la página Plantillas de servicio, elija Crear plantilla de servicio.

3. En la página Crear plantilla de servicio, en la sección Origen del paquete de plantillas, seleccione Sincronización de un paquete de plantillas desde Git.
4. En la sección Plantilla, seleccione Elegir un repositorio de Git vinculado.
5. Seleccione my-proton-connection de la Lista de repositorios.
6. Seleccione main en la Lista de ramificaciones.
7. En la sección de Detalles de la plantilla de servicio de Proton.

   a. Introduzca el nombre de la plantilla de servicio como **`backend-fargate-svc`**.

   b. Introduzca el nombre para mostrar de la plantilla de servicio como **`My Fargate Service`**.

   c. (Opcional) Escriba una descripción para la plantilla de servicio.
8. En la sección Plantillas de entorno compatibles.

   - Marque la casilla de verificación situada a la izquierda de la plantilla de entorno My Fargate Environment para seleccionar la plantilla de entorno compatible para la nueva plantilla de servicio.
9. En Configuración de cifrado, utilice la configuración predeterminada.
10. En la sección Definición de la canalización.

    - Mantenga seleccionado el botón Esta plantilla incluye una canalización de CI/CD.
11. Elija Crear plantilla.

    Ahora se encuentra en una nueva página que muestra el estado y los detalles de su nueva plantilla de servicio, incluida una lista de AWS y las etiquetas administradas por el cliente.
12. El estado inicial de una nueva plantilla de servicio será el estado Borrador. Solo los administradores pueden ver la plantilla y acceder a ella. Para que la plantilla de servicio esté disponible para que la utilicen los desarrolladores, siga el siguiente paso.
13. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria de la plantilla que acaba de crear (1.0). Como alternativa, puede elegir Publicar en el banner de alerta de información y omitir el paso siguiente.
14. En la sección Versiones de la plantilla, seleccione Publicar.
15. El estado de la plantilla cambiará a Publicado.

    La primera versión secundaria de la plantilla de servicio está publicada y disponible para que la utilicen los desarrolladores. Como esta es la versión más reciente de la plantilla, pasará a ser la versión Recomendada.

16. En el panel de navegación, elija Plantillas de servicio.

    Una nueva página muestra una lista de las plantillas de servicio del usuario y los detalles de cada una de ellas.

Paso 5: Crear un entorno

En el panel de navegación, elija Entornos.

1. Seleccione Crear entorno.
2. En la página Elegir una plantilla de entorno, seleccione la plantilla que acaba de crear. Se llama My Fargate Environment. A continuación, seleccione Configurar.
3. En la página Configurar entorno, en la sección Aprovisionamiento, seleccione Aprovisionar mediante AWS Proton.
4. En la sección Cuenta de implementación, seleccione Esta Cuenta de AWS.
5. En Configuración del entorno, introduzca el nombre del entorno como **`my-fargate-environment`**.
6. En la sección Roles de entorno, seleccione Nuevo rol de servicio o, si ya ha creado un rol de servicio de AWS Proton, seleccione Rol de servicio existente.

    a. Seleccione Nuevo rol de servicio para crear un nuevo rol.

        i. Introduzca el Nombre del rol de entorno como **`MyProtonServiceRole`**.

        ii. Marque la casilla de verificación para aceptar la creación de un rol de servicio de AWS Proton con privilegios administrativos para la cuenta.

    b. Seleccione un Rol de servicio existente para utilizar un rol existente.

        - Seleccione el rol en el campo desplegable Nombre del rol de entorno.
7. Elija Siguiente.
8. En la página Configurar ajustes personalizados, utilice los valores predeterminados.
9. Seleccione Siguiente y revise las entradas.
10. Seleccione Crear.

    Consulte los detalles y el estado del entorno, así como las etiquetas administradas por AWS y las etiquetas administradas por el cliente para dicho entorno.
11. En el panel de navegación, elija Entornos.

Una nueva página mostrará una lista de sus entornos junto con el estado y otros detalles del entorno.

Paso 6: (Opcional) Crear un servicio e implementar una aplicación

1. Abra la consola de AWS Proton.
2. En el panel de navegación, elija Servicios.
3. En la página Servicios, seleccione Crear servicio.
4. En la página Elegir una plantilla de servicio, seleccione la plantilla My Fargate Service; para ello, pulse el botón de radio situado en la esquina superior derecha de la tarjeta de plantilla.
5. Seleccione Configurar en la esquina inferior derecha de la página.
6. En la página Configurar servicio, en la sección Configuración del servicio, introduzca el nombre del servicio **`my-service`**.
7. (Opcional) Escriba una descripción para el servicio.
8. En la sección Configuración del repositorio de servicios:
   a. Para Conexión de Codestar, elija una conexión de la lista.
   b. En Nombre del repositorio, elija de la lista el nombre del repositorio del código fuente.
   c. En Nombre de la ramificación, elija de la lista el nombre de la ramificación del repositorio de código fuente.
9. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente. A continuación, elija Next.
10. En la página Configurar ajustes personalizados, en la sección Instancias de servicio, en la sección Nueva instancia, siga los pasos siguientes para proporcionar valores personalizados para los parámetros de la instancia de servicio.
    a. Introduzca el nombre de la instancia **`my-app-service`**.
    b. Elija el entorno **`my-fargate-environment`** para la instancia de servicio.
    c. Mantenga los valores predeterminados para el resto de parámetros de la instancia.
    d. Mantenga los valores predeterminados para las entradas de la canalización.
    e. Seleccione Siguiente y revise las entradas.
    f. Seleccione Crear y consulte el estado y los detalles del servicio.

11. En la página de detalles del servicio, seleccione las pestañas Información general y Canalización para ver el estado de la instancia de servicio y de la canalización. En estas páginas también podrá ver AWS y las etiquetas administradas por el cliente. AWS Proton crea automáticamente etiquetas administradas por AWS. Elija Administrar etiquetas para crear y modificar las etiquetas administradas por el cliente. Para obtener más información acerca del etiquetado, consulte Recursos y etiquetado de AWS Proton.
12. Cuando el servicio esté Activo, en la pestaña Información general, en la sección Instancias de servicio, elija como nombre de la instancia de servicio my-app-service.

    Ahora se encuentra en la página de detalles de la instancia de servicio.
13. Para ver la aplicación, en la sección Salidas, copie el enlace ServiceEndpoint en su navegador.

    Verá un gráfico de AWS Proton en la página web.
14. Una vez creado el servicio, en el panel de navegación, elija Servicios para ver una lista de sus servicios.

Paso 7: Limpiar.

1. Abra la consola de AWS Proton.
2. Eliminación de un servicio (si se creó alguno)
    a. En el panel de navegación, elija Servicios.
    b. En la página Servicios, elija como nombre del servicio my-service.

       Ahora se encuentra en la página de detalles del servicio de my-service.
    c. En la esquina superior derecha de la página, seleccione Acciones y, a continuación, Eliminar.
    d. Un modal le pedirá que confirme la acción de eliminación.
    e. Siga las instrucciones y seleccione Sí, eliminar.
3. Eliminación de un entorno
    a. En el panel de navegación, elija Entornos.
    b. En la página Entornos, seleccione el botón de radio a la izquierda del entorno recién creado.
    c. Elija Acciones y, a continuación, elija Eliminar.
    d. Un modal le pedirá que confirme la acción de eliminación.

e. Siga las instrucciones y seleccione Sí, eliminar.

4. Eliminación de una plantilla de servicio

   a. En el panel de navegación, elija Plantillas de servicio.

   b. En la página Plantillas de servicio, seleccione el botón de radio situado a la izquierda de la plantilla de servicio my-svc-template.

   c. Elija Acciones y, a continuación, elija Eliminar.

   d. Un modal le pedirá que confirme la acción de eliminación.

   e. Siga las instrucciones y seleccione Sí, eliminar. Esto elimina la plantilla de servicio y todas sus versiones.

5. Eliminación de una plantilla de entorno

   a. En el panel de navegación, elija Plantillas de entorno.

   b. En la página Plantillas de entorno, seleccione el botón de radio situado a la izquierda de my-env-template.

   c. Elija Acciones y, a continuación, elija Eliminar.

   d. Un modal le pedirá que confirme la acción de eliminación.

   e. Siga las instrucciones y seleccione Sí, eliminar. Esto elimina la plantilla de entorno y todas sus versiones.

6. Eliminación de la conexión de CodeStar

Introducción al AWS CLI

Para empezar con AWS Proton mediante la AWS CLI, siga este tutorial. El tutorial muestra un servicio de AWS Proton con carga equilibrada y orientado al público basado en AWS Fargate. El tutorial también proporciona una canalización de CI/CD que implementa un sitio web estático con una imagen para mostrar.

Antes de empezar, el usuario debe asegurarse de que esté configurado correctamente. Para obtener más información, consulte the section called “Requisitos previos”.

Paso 1: Registrar una plantilla de entorno

En este paso, como administrador, el usuario registrará una plantilla de entorno de ejemplo que contenga un clúster de Amazon Elastic Container Service (Amazon ECS) y una Amazon Virtual Private Cloud (Amazon VPC) con dos subredes, una pública y otra privada.

Para registrar una plantilla de entorno

1. Cree una adaptación del repositorio de plantillas de CloudFormation de muestra de AWS Proton en la cuenta u organización de GitHub. Este repositorio incluye las plantillas de entorno y de servicio que utilizaremos en este tutorial.

   A continuación, registre el repositorio adaptado con AWS Proton. Para obtener más información, consulte the section called “Creación de un enlace a un repositorio”.

2. Cree una plantilla de entorno.

   El recurso de la plantilla de entorno realiza un seguimiento de las versiones de la plantilla de entorno.

   ```
   $ aws proton create-environment-template \
     --name "fargate-env" \
     --display-name "Public VPC Fargate" \
     --description "VPC with public access and ECS cluster"
   ```

3. Cree una configuración de sincronización de plantillas.

   AWS Proton establece una relación de sincronización entre el repositorio y la plantilla de entorno. A continuación, crea la versión 1.0 de la plantilla en estado `DRAFT`.

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

4. Espere a que la versión de la plantilla de entorno se registre correctamente.

Cuando se devuelva este comando con un estado de salida de `0`, se habrá completado el registro de la versión. Esto resulta útil en los scripts para garantizar que se pueda ejecutar correctamente el comando en el siguiente paso.

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

5. Publique la versión de la plantilla de entorno para que esté disponible para la creación del entorno.

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

Paso 2: Registrar una plantilla de servicio

En este paso, como administrador, el usuario registrará una plantilla de servicio de ejemplo que contenga todos los recursos necesarios para aprovisionar un servicio de Fargate de Amazon ECS detrás de un equilibrador de carga y una canalización de CI/CD que utilice AWS CodePipeline.

Para registrar una plantilla de servicio

1. Cree una plantilla de servicio.

   El recurso de la plantilla de servicio realiza un seguimiento de las versiones de la plantilla de servicio.

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

2. Cree una configuración de sincronización de plantillas.

AWS Proton establece una relación de sincronización entre el repositorio y la plantilla de servicio. A continuación, crea la versión 1.0 de la plantilla en estado DRAFT.

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

3. Espere a que la versión de la plantilla de servicio se registre correctamente.

   Cuando se devuelva este comando con un estado de salida de `0`, se habrá completado el registro de la versión. Esto resulta útil en los scripts para garantizar que se pueda ejecutar correctamente el comando en el siguiente paso.

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

4. Publique la versión de la plantilla de servicio para que esté disponible para la creación del servicio.

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

Paso 3: Implementar un entorno

En este paso, como administrador, el usuario creará una instancia de un entorno de AWS Proton a partir de la plantilla de entorno.

Para implementar un entorno

1. Obtenga un archivo de especificaciones de ejemplo para la plantilla de entorno que ha registrado.

   Puede descargar el archivo `environment-templates/fargate-env/spec/spec.yaml` desde el repositorio de ejemplos de plantillas. Como alternativa, puede obtener todo el repositorio de forma local y ejecutar el comando create-environment desde el directorio `environment-templates/fargate-env`.

2. Cree un entorno.

   AWS Proton lee los valores de entrada de las especificaciones del entorno, los combina con la plantilla de entorno y aprovisiona los recursos del entorno en la cuenta de AWS del usuario mediante su rol de servicio de AWS Proton.

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

3. Espere a que el entorno se implemente correctamente.

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

Paso 4: Implementar un servicio [desarrollador de aplicaciones]

En los pasos anteriores, un administrador registró y publicó una plantilla de servicio e implementó un entorno. Como desarrollador de aplicaciones, ahora puede crear un servicio de AWS Proton e implementarlo en el entorno de AWS Proton

Para implementar un servicio

1. Obtenga un archivo de especificaciones de ejemplo para la plantilla de servicio que registró el administrador.

   Puede descargar el archivo `service-templates/load-balanced-fargate-svc/spec/spec.yaml` desde el repositorio de ejemplos de plantillas. Como alternativa, puede obtener todo

el repositorio de forma local y ejecutar el comando create-service desde el directorio `service-templates/load-balanced-fargate-svc`.

2. Cree una adaptación del repositorio de los servicios de muestra de AWS Proton en la cuenta u organización de GitHub. Este repositorio incluye el código fuente de la aplicación que utilizamos en este tutorial.

3. Cree un servicio.

   AWS Proton lee los valores de entrada de la especificación de servicio, los combina con la plantilla de servicio y aprovisiona los recursos de servicio de la cuenta de AWS del usuario en el entorno especificado en la especificación. Una canalización de AWS CodePipeline implementa el código de la aplicación desde el repositorio que especifique el usuario en el comando.

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

4. Espere a que el servicio se implemente correctamente.

   ```
   $ aws proton wait service-created --name "static-website"
   ```

5. Recupere las salidas y consulte el nuevo sitio web.

   Ejecute el siguiente comando:

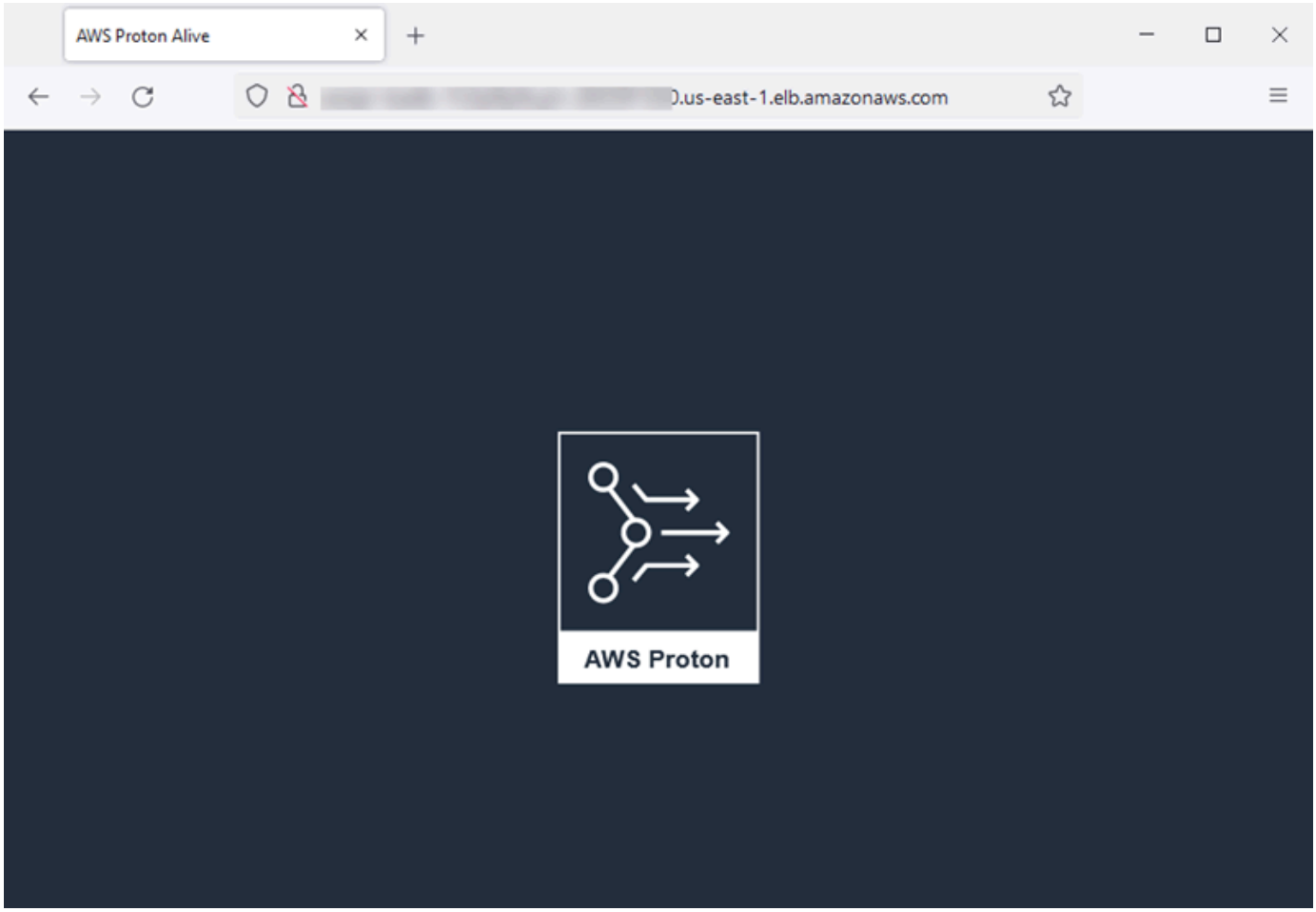
   ```
   $ aws proton list-service-instance-outputs \
     --service-name "static-website" \
     --service-instance-name load-balanced-fargate-svc-prod
   ```

   La salida del comando debería ser similar a la siguiente:

   ```
   {
       "outputs": [
           {
   ```

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

El valor de la salida de la instancia `ServiceURL` es el punto de conexión del nuevo sitio web del servicio. Utilice el navegador para navegar hasta él. Debería ver el siguiente gráfico en una página estática:



Paso 5: Limpieza (opcional)

En este paso, cuando haya terminado de explorar los recursos de AWS que creó como parte de este tutorial y para ahorrar en los costos asociados a estos recursos, procederá a eliminarlos.

Para eliminar los recursos del tutorial

1. Para eliminar el servicio, ejecute el comando siguiente:

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

2. Para eliminar el entorno, ejecute el siguiente comando:

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

3. Para eliminar la plantilla de servicio, ejecute los siguientes comandos:

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

4. Para eliminar la plantilla de entorno, ejecute los siguientes comandos:

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

La biblioteca de plantillas de AWS Proton

El equipo de AWS Proton mantiene una biblioteca de ejemplos de plantillas en GitHub. La biblioteca incluye ejemplos de archivos de infraestructura como código (IaC) para muchos escenarios comunes de infraestructura de aplicaciones y entornos.

La biblioteca de plantillas se almacena en dos repositorios de GitHub:

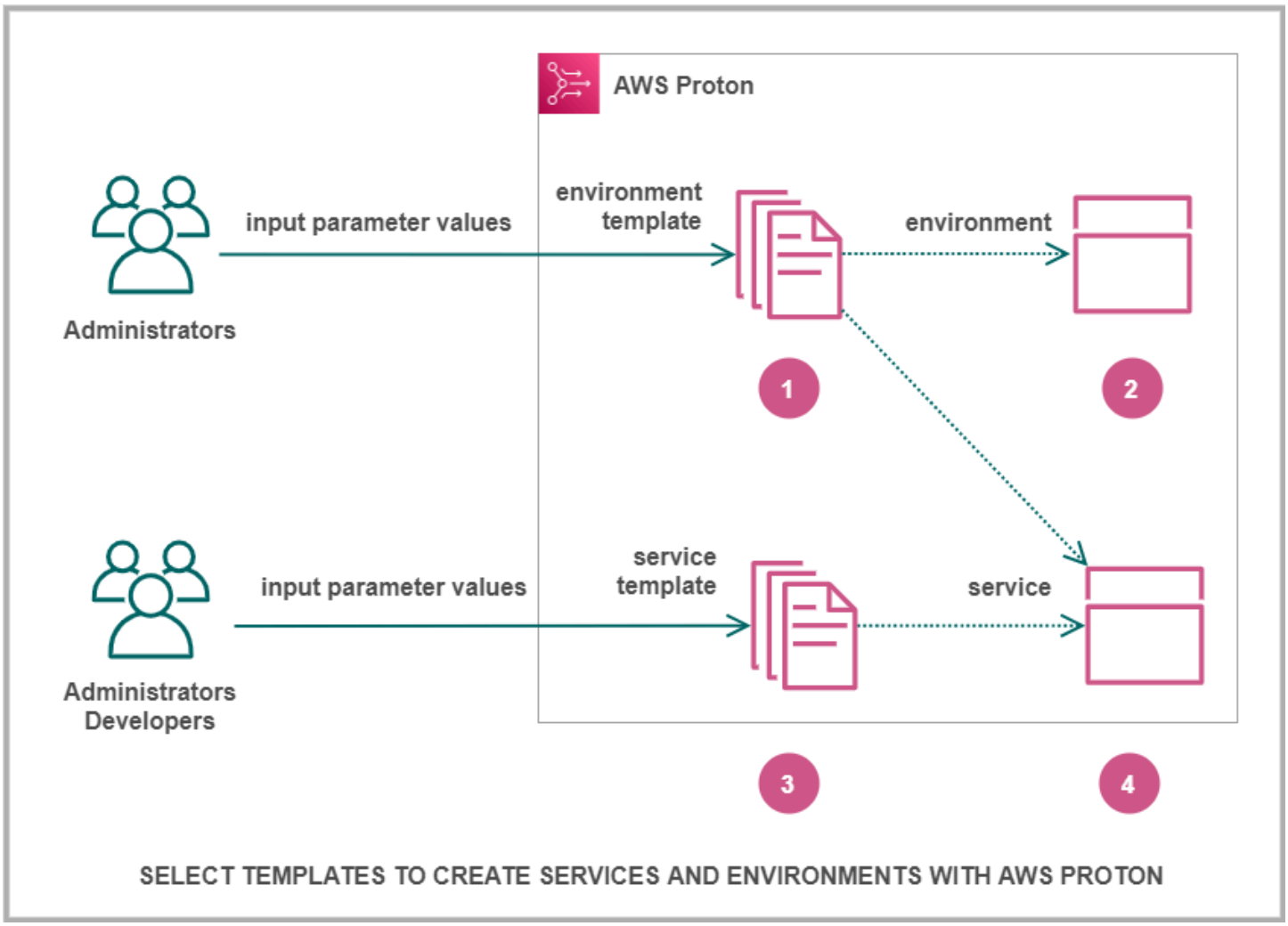
- aws-proton-cloudformation-sample-templates: ejemplos de paquetes de plantillas que utilizan AWS CloudFormation con Jinja como lenguaje de IaC. Puede utilizar estos ejemplos para entornos Aprovisionamiento administrado por AWS.
- aws-proton-terraform-sample-templates: ejemplos de paquetes de plantillas que utilizan Terraform como lenguaje de IaC. Puede utilizar estos ejemplos para entornos Aprovisionamiento autoadministrado.

Cada uno de estos repositorios tiene un archivo `README` con información completa sobre el contenido y la estructura del repositorio. Cada ejemplo contiene información sobre el caso de uso que cubre la plantilla, la arquitectura del ejemplo y los parámetros de entrada que utiliza la plantilla.

Puede utilizar las plantillas de esta biblioteca directamente; para ello, cree una adaptación de alguno de los repositorios de la biblioteca en su cuenta de GitHub. Como alternativa, utilice estos ejemplos como punto de partida para desarrollar sus plantillas de entorno y servicio.

Cómo funciona AWS Proton

Con AWS Proton, el usuario aprovisiona entornos y, a continuación, servicios que se ejecutan en esos entornos. Los entornos y los servicios se basan en las plantillas de entorno y servicio, respectivamente, que el usuario elige en su biblioteca de plantillas versionadas de AWS Proton.



1

Cuando el usuario, como administrador, selecciona una plantilla de entorno con AWS Proton, proporciona valores para los parámetros de entrada necesarios.

2

AWS Proton utiliza la plantilla de entorno y los valores de los parámetros para aprovisionar el entorno.

3

Cuando el usuario, como desarrollador o administrador, selecciona una plantilla de servicio con AWS Proton, proporciona valores para los parámetros de entrada necesarios. También selecciona un entorno en el que implementar la aplicación o el servicio.

4

AWS Proton utiliza la plantilla de servicio y los valores de los parámetros tanto del servicio como del entorno seleccionado para aprovisionar el servicio.

Debe proporcionar valores para los parámetros de entrada a fin de personalizar la plantilla para su reutilización y para varios casos de uso, aplicaciones o servicios.

Para que esto funcione, el usuario debe crear paquetes de plantillas de entorno o servicio y cargarlos en plantillas de entorno o servicio registradas, respectivamente.

Los paquetes de plantillas contienen todo lo que AWS Proton necesita para aprovisionar entornos o servicios.

Al crear una plantilla de entorno o de servicio, se carga un paquete de plantillas que contiene los archivos parametrizados de infraestructura como código (IaC) que AWS Proton utiliza para aprovisionar entornos o servicios.

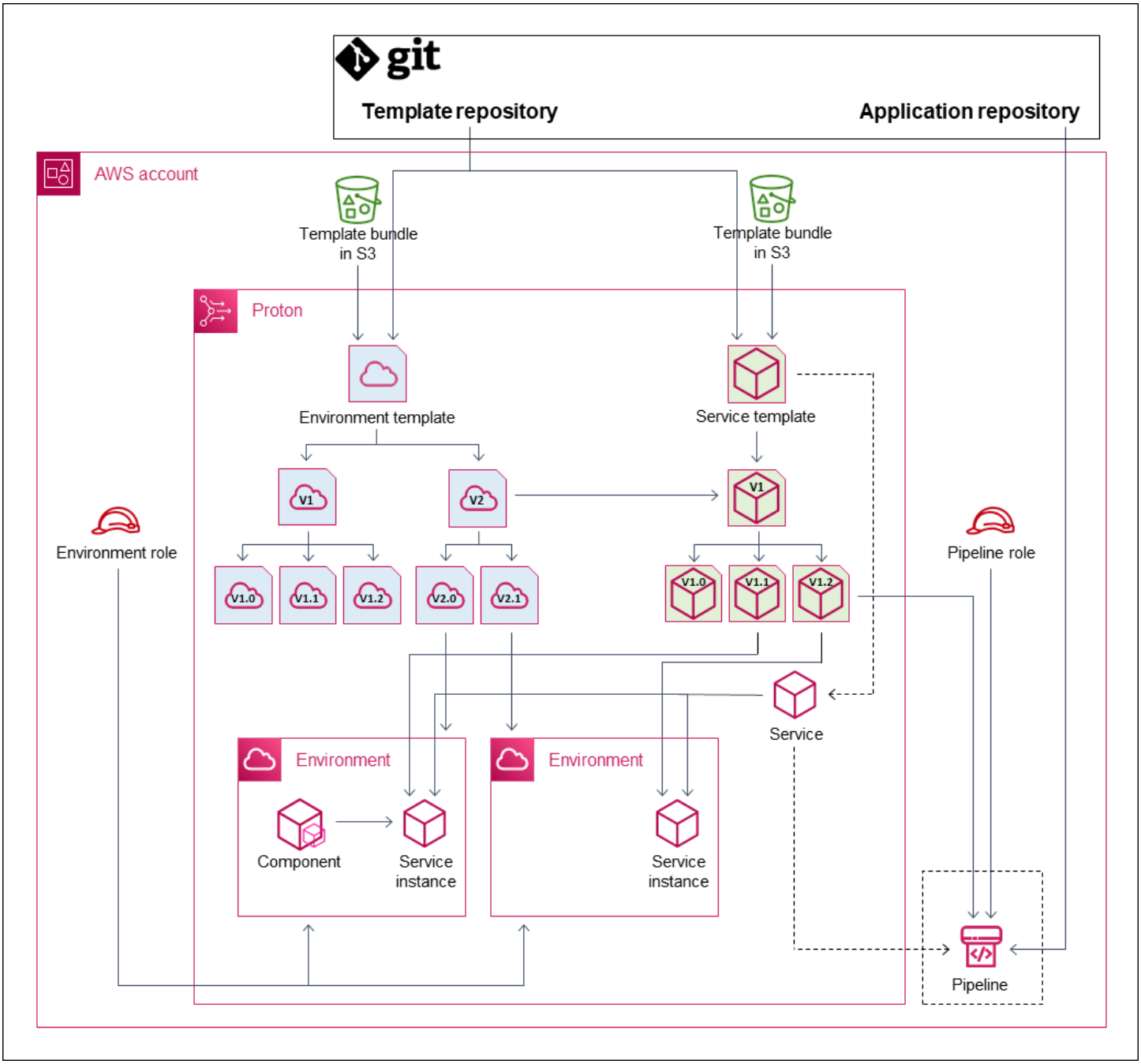
Al seleccionar una plantilla de entorno o de servicio para crear o actualizar un entorno o servicio, el usuario proporciona valores para los parámetros del archivo de IaC del paquete de plantillas.

Temas

Objetos de AWS Proton

El siguiente diagrama muestra los objetos principales de AWS Proton y su relación con otros objetos de AWS y de terceros. Las flechas representan la dirección del flujo de datos (la dirección inversa de la dependencia).

Seguimos el diagrama con breves descripciones y enlaces de referencia para estos objetos de AWS Proton.



- Plantilla de entorno: un conjunto de versiones de plantillas de entorno que se pueden utilizar para crear entornos de AWS Proton.

  Para obtener más información, consulte Creación de plantillas y paquetes y Plantillas.
- Versión de plantilla de entorno: versión específica de una plantilla de entorno. Toma un paquete de plantillas como entrada, ya sea desde un bucket de S3 o desde un repositorio de Git. El paquete especifica la infraestructura como código (IaC) y los parámetros de entrada relacionados para un entorno de AWS Proton.

Para obtener más información, consulte the section called “Versiones”, the section called “Publicación” y the section called “Configuraciones de sincronización de plantillas”.

- Entorno: el conjunto de políticas de acceso y recursos de infraestructura de AWS compartidos en los que se implementan los servicios de AWS Proton. Los recursos de AWS se aprovisionan mediante una versión de plantilla de entorno invocada con valores de parámetros específicos. Las políticas de acceso se proporcionan en un rol de servicio.

  Para obtener más información, consulte Entornos.

- Plantilla de servicio: un conjunto de versiones de plantillas de servicio que se pueden utilizar para crear servicios de AWS Proton.

  Para obtener más información, consulte Creación de plantillas y paquetes y Plantillas.

- Versión de plantilla de servicio: versión específica de una plantilla de servicio. Toma un paquete de plantillas como entrada, ya sea desde un bucket de S3 o desde un repositorio de Git. El paquete especifica la infraestructura como código (IaC) y los parámetros de entrada relacionados para un servicio de AWS Proton.

  Una versión de plantilla de servicio también especifica estas restricciones en las instancias de servicio en función de la versión:

  - Plantillas de entorno compatibles: las instancias solo se pueden ejecutar en entornos basados en estas plantillas de entorno compatibles.
  - Orígenes de componentes compatibles: los tipos de componentes que los desarrolladores pueden asociar a las instancias.

  Para obtener más información, consulte the section called “Versiones”, the section called “Publicación” y the section called “Configuraciones de sincronización de plantillas”.

- Servicio: un conjunto de instancias de servicio que ejecutan una aplicación con los recursos especificados en una plantilla de servicio y, opcionalmente, una canalización de CI/CD que implementa el código de la aplicación en estas instancias.

  En el diagrama, la línea discontinua de la plantilla de servicio significa que el servicio pasa la plantilla a las instancias de servicio y a la canalización.

  Para obtener más información, consulte Servicios.

- Instancia de servicio: conjunto de recursos de la infraestructura de AWS que ejecutan una aplicación en un entorno específico de AWS Proton. Los recursos de AWS se aprovisionan mediante una versión de plantilla de servicio que se invoca con valores de parámetros específicos.

  Para obtener más información, consulte Servicios y the section called “Actualización de una instancia”.
- Canalización: una canalización de CI/CD opcional que implementa una aplicación en las instancias de un servicio, con políticas de acceso para aprovisionar esta canalización. Las políticas de acceso se proporcionan en un rol de servicio. Un servicio no siempre tiene una canalización de AWS Proton asociada; puede optar por administrar las implementaciones de código de la aplicación fuera de AWS Proton.

  En el diagrama, la línea discontinua de Servicio y el recuadro discontinuo que rodea a Canalización significan que si el usuario decidiera administrar sus implementaciones de CI/CD por sí mismo, es posible que no se cree la canalización de AWS Proton y que la canalización propia del usuario no se encuentre en su cuenta de AWS.

  Para obtener más información, consulte Servicios y the section called “Actualización de una canalización”.
- Componente: una extensión de una instancia de servicio definida por el desarrollador. Especifica los recursos de la infraestructura de AWS adicionales que una aplicación concreta podría necesitar, además de los recursos proporcionados por el entorno y la instancia de servicio. Los equipos de plataforma controlan la infraestructura que un componente puede aprovisionar al asociar un rol de componente al entorno.

  Para obtener más información, consulte Componentes.

Cómo AWS Proton aprovisiona la infraestructura

AWS Proton puede aprovisionar la infraestructura de varias maneras:

- Aprovisionamiento administrado por AWS: AWS Proton llama al motor de aprovisionamiento en su nombre. Este método solo admite paquetes de plantillas de AWS CloudFormation. Para obtener más información, consulte the section called “AWS CloudFormation Archivos iAC”.
- Aprovisionamiento de CodeBuild: AWS Proton utiliza AWS CodeBuild para ejecutar los comandos del intérprete de comandos que proporcione el usuario. Los comandos pueden leer las entradas que proporciona AWS Proton, y son responsables de aprovisionar o desaprovisionar la

infraestructura y generar valores de salida. Un paquete de plantillas para este método incluye los comandos en un archivo de manifiesto y todos los programas, scripts u otros archivos que estos comandos puedan necesitar.

Como ejemplo del uso del aprovisionamiento de CodeBuild, puede incluir código que utilice el AWS Cloud Development Kit (AWS CDK) para aprovisionar recursos de AWS y un manifiesto que instale el CDK y ejecute su código de CDK.

Para obtener más información, consulte the section called “CodeBuild paquete”.

> **Note**
>
> Puede utilizar el aprovisionamiento de CodeBuild con entornos y servicios. En este momento, no puede aprovisionar componentes de esta forma.

- Aprovisionamiento autoadministrado: AWS Proton envía una solicitud de extracción (PR) a un repositorio que el usuario proporcione, donde su propio sistema de implementación de infraestructuras ejecute el proceso de aprovisionamiento. Este método solo admite paquetes de plantillas de Terraform. Para obtener más información, consulte the section called “Archivos iAC de Terraform”.

AWS Proton determina y establece el método de aprovisionamiento para cada entorno y servicio por separado. Al crear o actualizar un entorno o un servicio, AWS Proton examina el paquete de plantillas que el usuario proporciona y determina el método de aprovisionamiento que indica el paquete de plantillas. A nivel del entorno, el usuario proporciona los parámetros que el entorno y sus posibles servicios podrían necesitar para sus métodos de aprovisionamiento: roles de AWS Identity and Access Management (IAM), una conexión de cuentas de entorno o un repositorio de infraestructura.

Los desarrolladores que utilizan AWS Proton para aprovisionar un servicio tienen la misma experiencia independientemente del método de aprovisionamiento. Los desarrolladores no necesitan conocer el método de aprovisionamiento ni tienen que cambiar nada en el proceso de aprovisionamiento del servicio. La plantilla de servicio establece el método de aprovisionamiento y cada entorno en el que un desarrollador implementa el servicio proporciona los parámetros necesarios para el aprovisionamiento de las instancias de servicio.

El siguiente diagrama resume algunas de las características principales de los distintos métodos de aprovisionamiento. En las secciones que siguen a la tabla se proporcionan detalles sobre cada método.

Método de aprovisionamiento	Plantillas	Aprovisionado por	Estado registrado por
Administrado por AWS	manifiesto, esquema, archivo de IaC (CloudFormation)	AWS Proton (a través de CloudFormation)	AWS Proton (a través de CloudFormation)
CodeBuild	manifiesto (con comandos), esquema, dependencias de comandos (por ejemplo, código del AWS CDK)	AWS Proton (a través de CodeBuild)	AWS Proton (los comandos del usuario devuelven el estado a través de CodeBuild)
autoadministrado	manifiesto, esquema, archivos de IaC (Terraform)	Código del usuario (a través de acciones de Git)	Código del usuario (transferido a AWS a través de una llamada a la API)

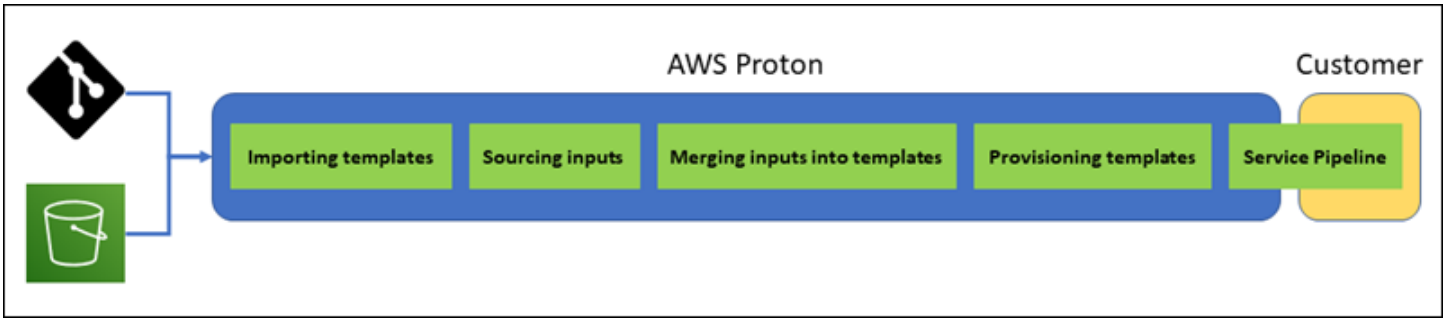
¿Cómo funciona el aprovisionamiento administrado por AWS?

Cuando un entorno o un servicio utiliza el aprovisionamiento administrado por AWS, la infraestructura se aprovisiona de la siguiente manera:

1. Un cliente de AWS Proton (un administrador o un desarrollador) crea el recurso de AWS Proton (un entorno o un servicio). El cliente selecciona una plantilla para el recurso y proporciona los parámetros necesarios. Para obtener más información, consulte la siguiente sección: the section called “Consideraciones sobre el aprovisionamiento administrado por AWS”.
2. AWS Proton representa una plantilla completa de AWS CloudFormation para aprovisionar el recurso.
3. AWS Proton llama a AWS CloudFormation para iniciar el aprovisionamiento mediante la plantilla representada.

4. AWS Proton monitorea continuamente la implementación de AWS CloudFormation.
5. Cuando se completa el aprovisionamiento, AWS Proton informa de los errores en caso de error y captura las salidas del aprovisionamiento, como el ID de la VPC de Amazon, en caso de que se realice correctamente.

En el siguiente diagrama se muestra que AWS Proton se encarga directamente de la mayoría de estos pasos.



Consideraciones sobre el aprovisionamiento administrado por AWS

- Rol de aprovisionamiento de la infraestructura: cuando un entorno o alguna de las instancias de servicio que se ejecutan en él pueda utilizar el aprovisionamiento administrado por AWS, el administrador debe configurar un rol de IAM (directamente o como parte de la conexión de una cuenta de entorno de AWS Proton). AWS Proton utiliza este rol para aprovisionar la infraestructura de estos recursos de aprovisionamiento administrados por AWS. El rol debe tener permisos para utilizar AWS CloudFormation con el fin de crear todos los recursos que incluyan las plantillas de dichos recursos.

  Para obtener más información, consulte the section called “Roles de IAM” y the section called “Ejemplos de políticas de roles de servicio”.
- Aprovisionamiento de servicios: cuando un desarrollador implementa una instancia de servicio que utiliza el aprovisionamiento administrado por AWS en el entorno, AWS Proton utiliza el rol proporcionado a ese entorno para aprovisionar la infraestructura de la instancia de servicio. Los desarrolladores no ven este rol y no pueden cambiarlo.
- Servicio con canalización: una plantilla de servicio que utilice el aprovisionamiento administrado por AWS puede incluir una definición de canalización escrita en el esquema YAML de AWS CloudFormation. AWS Proton también crea la canalización mediante una llamada a AWS CloudFormation. El rol que utiliza AWS Proton para crear una canalización es independiente del rol de cada entorno individual. Este rol se proporciona en AWS Proton por separado, solo una vez

a nivel de cuenta de AWS, y se utiliza para aprovisionar y administrar todas las canalizaciones administradas por AWS. Este rol debe tener permisos para crear canalizaciones y otros recursos que las canalizaciones necesiten.

Los siguientes procedimientos muestran cómo proporcionar el rol de canalización a AWS Proton.

AWS Proton console

Para proporcionar el rol de canalización

1. En la consola de AWS Proton, en el panel de navegación, seleccione Configuración > Configuración de la cuenta y, a continuación, seleccione Configurar.
2. Utilice la sección Rol administrado por AWS de la canalización para configurar un rol de canalización nuevo o existente para el aprovisionamiento administrado por AWS.

AWS Proton API

Para proporcionar el rol de canalización

1. Utilice la acción de la API UpdateAccountSettings.
2. Proporciona el nombre de recurso de Amazon (ARN) de su rol de servicio de canalización en el parámetro `pipelineServiceRoleArn`.

AWS CLI

Para proporcionar el rol de canalización

Ejecute el siguiente comando:

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

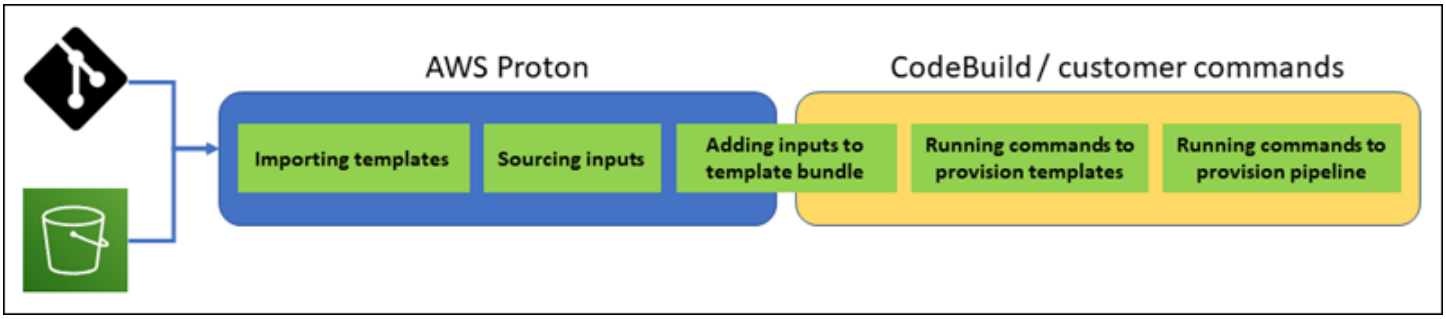
Cómo funciona el aprovisionamiento de CodeBuild

Cuando un entorno o un servicio utiliza el aprovisionamiento de CodeBuild, la infraestructura se aprovisiona de la siguiente manera:

1. Un cliente de AWS Proton (un administrador o un desarrollador) crea el recurso de AWS Proton (un entorno o un servicio). El cliente selecciona una plantilla para el recurso y proporciona los parámetros necesarios. Para obtener más información, consulte la siguiente sección: the section called “Consideraciones sobre el aprovisionamiento de CodeBuild”.
2. AWS Proton representa un archivo de entrada con los valores de los parámetros de entrada para aprovisionar el recurso.
3. AWS Proton llama a CodeBuild para iniciar un trabajo. El trabajo de CodeBuild ejecuta los comandos del intérprete de comandos del cliente especificados en la plantilla. Estos comandos proporcionan la infraestructura deseada y, opcionalmente, leen los valores de entrada.
4. Cuando se completa el aprovisionamiento, el comando final del cliente devuelve el estado del aprovisionamiento a CodeBuild y llama a la acción de la API de AWS Proton NotifyResourceDeploymentStatusChange para proporcionar salidas, como el ID de la VPC de Amazon, si existiera alguno.

> **Important**
>
> Asegúrese de que los comandos devuelvan correctamente el estado de aprovisionamiento a CodeBuild y proporcionen las salidas. Si no lo hacen, AWS Proton no podrá realizar un seguimiento adecuado del estado del aprovisionamiento y no podrá proporcionar las salidas correctas a las instancias de servicio.

El siguiente diagrama ilustra los pasos que AWS Proton lleva a cabo y los pasos que realizan los comandos del usuario en un trabajo de CodeBuild.



Consideraciones sobre el aprovisionamiento de CodeBuild

- Rol de aprovisionamiento de la infraestructura: cuando un entorno o cualquiera de las instancias de servicio que se ejecutan en él pueda utilizar el aprovisionamiento basado en CodeBuild, el administrador debe configurar un rol de IAM (directamente o como parte de una conexión de

cuenta de entorno de AWS Proton). AWS Proton utiliza este rol para aprovisionar la infraestructura de estos recursos de aprovisionamiento de CodeBuild. El rol debe tener permisos para utilizar CodeBuild a fin de crear todos los recursos que aprovisionan los comandos del usuario en las plantillas de dichos recursos.

Para obtener más información, consulte the section called “Roles de IAM” y the section called “Ejemplos de políticas de roles de servicio”.

- Aprovisionamiento de servicios: cuando un desarrollador implementa una instancia de servicio que utiliza el aprovisionamiento de CodeBuild en el entorno, AWS Proton utiliza el rol proporcionado a ese entorno para aprovisionar la infraestructura de la instancia de servicio. Los desarrolladores no ven este rol y no pueden cambiarlo.
- Servicio con canalización: una plantilla de servicio que utiliza el aprovisionamiento de CodeBuild puede incluir comandos para aprovisionar una canalización. AWS Proton también crea la canalización mediante llamadas a CodeBuild. El rol que utiliza AWS Proton para crear una canalización es independiente del rol de cada entorno individual. Este rol se proporciona a AWS Proton por separado, solo una vez a nivel de cuenta de AWS, y se utiliza para aprovisionar y administrar todas las canalizaciones basadas en CodeBuild. Este rol debe tener permisos para crear canalizaciones y otros recursos que las canalizaciones necesiten.

  Los siguientes procedimientos muestran cómo proporcionar el rol de canalización a AWS Proton.

  AWS Proton console

  Para proporcionar el rol de canalización

  1. En la consola de AWS Proton, en el panel de navegación, seleccione Configuración > Configuración de la cuenta y, a continuación, seleccione Configurar.
  2. Consulte la sección Roles de aprovisionamiento de canalizaciones de CodeBuild para configurar un rol de canalización nuevo o existente para el aprovisionamiento de CodeBuild.

  AWS Proton API

  Para proporcionar el rol de canalización

  1. Utilice la acción de la API UpdateAccountSettings.
  2. Proporciona el nombre de recurso de Amazon (ARN) de su rol de servicio de canalización en el parámetro `pipelineCodebuildRoleArn`.

AWS CLI

Para proporcionar el rol de canalización

Ejecute el siguiente comando:

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

Cómo funciona el aprovisionamiento autoadministrado

Cuando un entorno se configura para utilizar el aprovisionamiento autoadministrado, la infraestructura se aprovisiona de la siguiente manera:

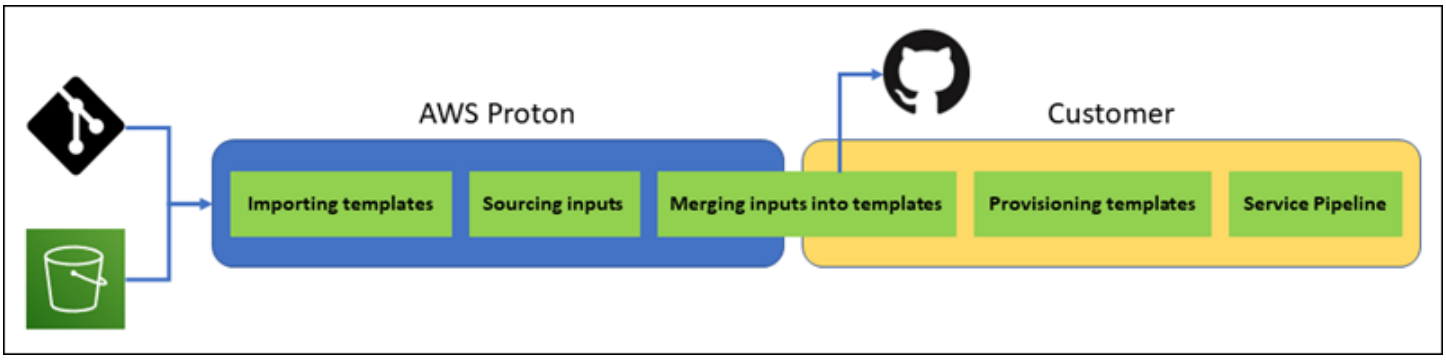
1. Un cliente de AWS Proton (un administrador o un desarrollador) crea el recurso de AWS Proton (un entorno o un servicio). El cliente selecciona una plantilla para el recurso y proporciona los parámetros necesarios. Para un entorno, el cliente también proporciona un repositorio de infraestructura vinculado. Para obtener más información, consulte la siguiente sección: the section called “Consideraciones sobre el aprovisionamiento autoadministrado”.
2. AWS Proton representa una plantilla de Terraform completa. Consta de uno o más archivos de Terraform, posiblemente en varias carpetas, y un archivo de variables de `.tfvars`. AWS Proton escribe los valores de los parámetros proporcionados en la llamada de creación del recurso en este archivo de variables.
3. AWS Proton envía una solicitud de extracción (PR) al repositorio de infraestructura con la plantilla de Terraform representada.
4. Cuando el cliente (administrador o desarrollador) fusiona la PR, la automatización del cliente activa el motor de aprovisionamiento para comenzar a aprovisionar la infraestructura mediante la plantilla fusionada.

> **Note**
>
> Si el cliente (administrador o desarrollador) cierra la solicitud de extracción (PR), AWS Proton la reconocerá como cerrada y marcará la implementación como cancelada.

5. Cuando se completa el aprovisionamiento, la automatización del cliente llama a la acción de la API de AWS Proton NotifyResourceDeploymentStatusChange para indicar que se ha completado y proporcionar tanto el estado (éxito o error) como las salidas, como el ID de la VPC de Amazon, si existiera alguna.

> **Important**
>
> Asegúrese de que el código de automatización vuelva a llamar a AWS Proton con el estado del aprovisionamiento y las salidas. Si no es así, AWS Proton podría considerar que el aprovisionamiento está pendiente durante más tiempo del debido y seguir mostrando el estado En curso.

El siguiente diagrama ilustra los pasos que AWS Proton lleva a cabo y los pasos que lleva a cabo su propio sistema de aprovisionamiento.



Consideraciones sobre el aprovisionamiento autoadministrado

- Repositorio de infraestructura: cuando un administrador configura un entorno para el aprovisionamiento autoadministrado, debe proporcionar un repositorio de infraestructura vinculado. AWS Proton envía las solicitudes de extracción (PR) a este repositorio para aprovisionar la infraestructura del entorno y todas las instancias de servicio que se implementan en él. La acción de automatización del repositorio propiedad del cliente debe asumir un rol de IAM con permisos para crear todos los recursos que incluyen las plantillas de servicio y entorno, y una identidad que refleje la cuenta de AWS de destino. Para ver un ejemplo de la acción de GitHub que asume un rol, consulte Asumir un rol en la documentación de la Acción “Configurar credenciales de AWS” para acciones de GitHub.

- Permisos: el código de aprovisionamiento del usuario debe autenticarse con una cuenta según sea necesario (por ejemplo, autenticarse en una cuenta de AWS) y proporcionar una autorización de aprovisionamiento de recursos (por ejemplo, proporcionar un rol).
- Aprovisionamiento de servicios: cuando un desarrollador implementa una instancia de servicio que utiliza el aprovisionamiento autoadministrado en el entorno, AWS Proton envía una solicitud de extracción (PR) al repositorio asociado al entorno con el fin de aprovisionar la infraestructura para la instancia de servicio. Los desarrolladores no verán el repositorio ni podrán cambiarlo.

  > **Note**
  >
  > Los desarrolladores que crean servicios utilizan el mismo proceso independientemente del método de aprovisionamiento, y la diferencia se extrae de ellos. Sin embargo, con el aprovisionamiento autoadministrado, los desarrolladores pueden experimentar una respuesta más lenta, ya que tienen que esperar a que alguien (que podría no ser ellos mismos) fusione la PR en el repositorio de infraestructura para poder iniciar el aprovisionamiento.

- Servicio con canalización: una plantilla de servicio para un entorno con aprovisionamiento autoadministrado puede incluir una definición de canalización (por ejemplo, una canalización de AWS CodePipeline) escrita en HCL de Terraform. Para permitir que AWS Proton aprovisione estas canalizaciones, un administrador debe proporcionar un repositorio de canalizaciones vinculado a AWS Proton. Al aprovisionar una canalización, la acción de automatización propiedad del cliente en el repositorio debe asumir un rol de IAM con permisos para aprovisionar la canalización y una identidad que refleje la cuenta de AWS de destino. El repositorio y el rol de la canalización son independientes de los que se utilizan para cada entorno individual. El repositorio vinculado se proporciona a AWS Proton por separado, solo una vez a nivel de cuenta de AWS, y se utiliza para aprovisionar y administrar todos los procesos. El rol debe tener permisos para crear canalizaciones y otros recursos que las canalizaciones necesiten.

  Los siguientes procedimientos muestran cómo proporcionar el repositorio y el rol de canalización a AWS Proton.

  AWS Proton console

  Para proporcionar el rol de canalización

  1. En la consola de AWS Proton, en el panel de navegación, seleccione Configuración > Configuración de la cuenta y, a continuación, seleccione Configurar.

2. Utilice la sección Repositorio de canalización de CI/CD para configurar un enlace de repositorio nuevo o existente.

AWS Proton API

Para proporcionar el rol de canalización

1. Utilice la acción de la API UpdateAccountSettings.
2. Proporciona el proveedor, el nombre y la ramificación del repositorio de la canalización en el parámetro `pipelineProvisioningRepository`.

AWS CLI

Para proporcionar el rol de canalización

Ejecute el siguiente comando:

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

- Eliminación de los recursos aprovisionados de forma autoadministrada: los módulos de Terraform pueden incluir elementos de configuración necesarios para el funcionamiento de Terraform, además de definiciones de recursos. Por lo tanto, AWS Proton no puede eliminar todos los archivos de Terraform de un entorno o instancia de servicio. En su lugar, AWS Proton marca los archivos para su eliminación y actualiza una marca en los metadatos de la solicitud de extracción (PR). La automatización puede leer esa marca y utilizarla para activar un comando de destrucción de Terraform.

Terminología de AWS Proton

Plantilla de entorno

Define la infraestructura compartida, como una VPC o un clúster, que utilizan varias aplicaciones o recursos.

Paquetes de plantillas de entorno

Un conjunto de archivos que se cargan para crear y registrar una plantilla de entorno en AWS Proton. Un paquete de plantillas de entorno contiene lo siguiente:

1. Un archivo de esquema que define la infraestructura como parámetros de entrada de código.
2. Un archivo de infraestructura como código (IaC) que define la infraestructura compartida, como una VPC o un clúster, que varias aplicaciones o recursos utilizan.
3. Un archivo de manifiesto que incluye el archivo de IaC.

Entorno

Infraestructura compartida aprovisionada, como una VPC o un clúster, que utilizan varias aplicaciones o recursos.

Plantilla de servicio

Define el tipo de infraestructura que se necesita para implementar y mantener una aplicación o un microservicio en un entorno.

Paquete de plantillas de servicio

Un conjunto de archivos que se cargan para crear y registrar una plantilla de servicio de AWS Proton. Un paquete de plantillas de servicio contiene lo siguiente:

1. Un archivo de esquema que define los parámetros de entrada de la infraestructura como código (IaC).
2. Un archivo de IaC que define la infraestructura necesaria para implementar y mantener una aplicación o un microservicio en un entorno.
3. Un archivo de manifiesto que incluye el archivo de IaC.
4. Opcional
   a. Un archivo de IaC que define la infraestructura de la canalización de servicios.
   b. Un archivo de manifiesto que incluye el archivo de IaC.

Servicio

Infraestructura aprovisionada necesaria para implementar y mantener una aplicación o un microservicio en un entorno.

Instancia de servicio

Infraestructura aprovisionada que admite una aplicación o un microservicio en un entorno.

Canalización de servicios

Infraestructura aprovisionada que admite una canalización.

Versión de plantilla

Versión principal o secundaria de una plantilla. Para obtener más información, consulte Plantillas versionadas.

Parámetros de entrada

Se define en un archivo de esquema y se utiliza en un archivo de infraestructura como código (IaC) para que el archivo de IaC se pueda utilizar de forma repetida y para una variedad de casos de uso.

Archivo de esquema

Define la infraestructura como parámetros de entrada del archivo de código.

Archivo de especificaciones

Especifica los valores de la infraestructura como parámetros de entrada de un archivo de código, tal como se define en un archivo de esquema.

Archivo de manifiesto

Muestra una infraestructura como archivo de código.

Creación de plantillas y creación de paquetes para AWS Proton

AWS Proton aprovisiona recursos para usted en función de la infraestructura como archivos de código (IaC). Describe la infraestructura en los archivos IaC reutilizables. Para que los archivos sean reutilizables para diferentes entornos y aplicaciones, debe crearlos como plantillas, definir los parámetros de entrada y utilizar estos parámetros en las definiciones de IaC. Al crear posteriormente un recurso de aprovisionamiento (entorno, instancia de servicio o componente), AWS Proton utiliza un motor de renderizado, que combina los valores de entrada con una plantilla para crear un archivo IaC listo para el aprovisionamiento.

Los administradores crean la mayoría de las plantillas como paquetes de plantillas y, a continuación, las cargan y las registran. AWS Proton En el resto de esta página se describen estos paquetes AWS Proton de plantillas. Los componentes definidos directamente son una excepción: los desarrolladores los crean y proporcionan directamente los archivos de plantillas de IaC. Para obtener más información sobre los componentes, consulte Componentes.

Temas

Paquetes de plantillas

Como administrador, puede crear y registrar plantillas con AWS Proton. Estas plantillas se utilizan para crear entornos y servicios. Al crear un servicio, AWS Proton aprovisiona e implementa instancias de servicio en entornos seleccionados. Para obtener más información, consulte AWS Proton para equipos de plataformas.

Para crear y registrar una plantilla AWS Proton, debe cargar un paquete de plantillas que contenga los archivos de infraestructura como código (IaC) AWS Proton necesarios para aprovisionar un entorno o servicio.

Un paquete de plantillas contiene lo siguiente:

- Un archivo de infraestructura como código (IaC) con un archivo YAML de manifiesto que incluye el archivo IaC.
- Un archivo YAML de esquema para las definiciones de los parámetros de entrada del archivo IaC.

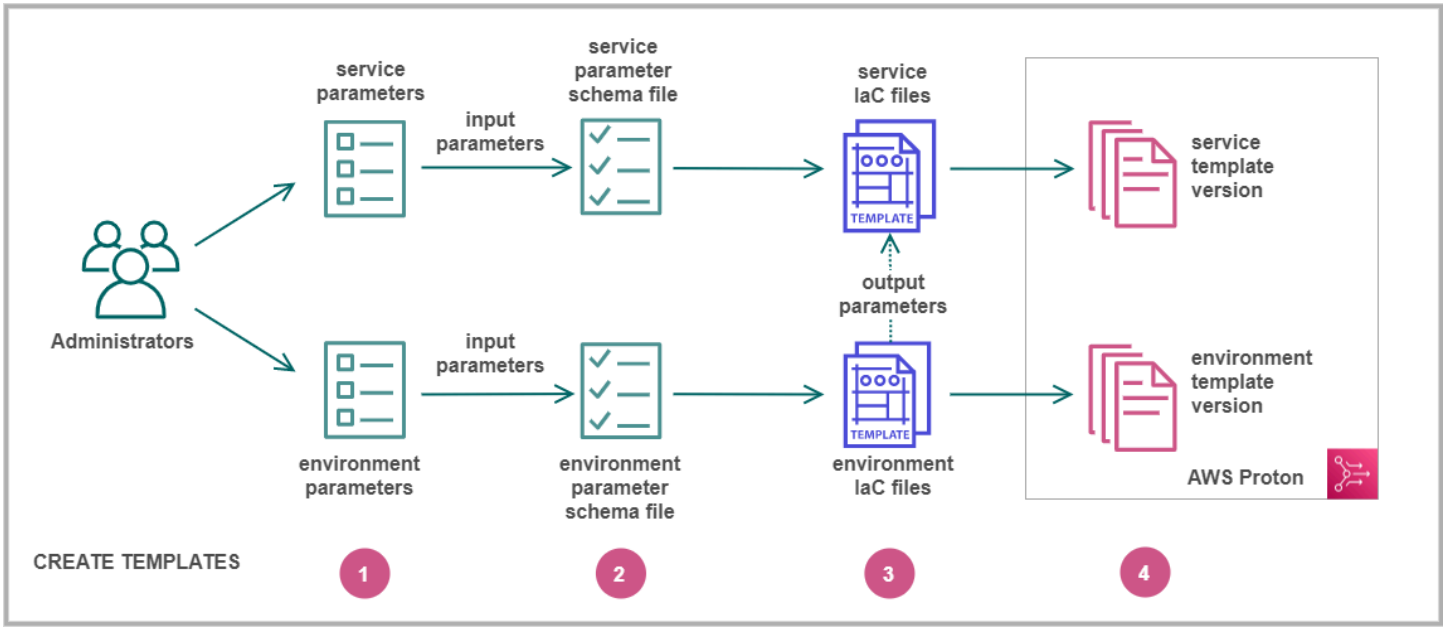
Un paquete de plantillas de CloudFormation entorno contiene un archivo IaC.

Un paquete CloudFormation de plantillas de servicio contiene un archivo IaC para las definiciones de las instancias de servicio y otro archivo IaC opcional para la definición de la canalización.

Los paquetes de plantillas de servicios y entornos de Terraform pueden contener varios archivos IaC cada uno.

AWS Proton requiere un archivo de esquema de parámetros de entrada. Cuando se crean AWS CloudFormation los archivos IaC, se utiliza la sintaxis de Jinja para hacer referencia a los parámetros de entrada. AWS Proton proporciona espacios de nombres de parámetros que puede utilizar para hacer referencia a los parámetros de sus archivos IaC.

El siguiente diagrama muestra un ejemplo de los pasos que puede seguir para crear una plantilla. AWS Proton



1 Identifi

los parámetros de entrada.

2 Cree un archivo de esquema para definir los parámetros de entrada.

3 Cree archivos IaC que hagan referencia a sus parámetros de entrada. Puede hacer referencia a las salidas de los archivos de IaC del entorno como entradas para los archivos de IaC del servicio.

4 Registr una versión de plantilla AWS Proton y cargue su paquete de plantillas.

AWS Proton parámetros

Puede definir y usar los parámetros de su infraestructura como archivos de código (IaC) para hacerlos flexibles y reutilizables. Para leer el valor de un parámetro en los archivos IaC, consulte el nombre del parámetro en el espacio de nombres del parámetro. AWS Proton AWS Proton inyecta valores de parámetros en los archivos IaC renderizados que genera durante el aprovisionamiento de recursos. Para procesar los parámetros de AWS CloudFormation IaC, usa Jinja. AWS Proton Para procesar los parámetros del IaC de Terraform, AWS Proton genera un archivo de valores de parámetros de Terraform y se basa en la capacidad de parametrización integrada en el HCL.

ConAprovisionamiento de CodeBuild, AWS Proton genera un archivo de entrada que su código puede importar. El archivo es un archivo JSON o HCL, según una propiedad del manifiesto de la plantilla. Para obtener más información, consulte the section called “CodeBuild parámetros de aprovisionamiento”.

Puede hacer referencia a los parámetros de los archivos IaC de su entorno, servicio y componente o al código de aprovisionamiento si cumple los siguientes requisitos:

- La longitud del nombre de cada parámetro no supera los 100 caracteres.
- La longitud combinada del espacio de nombres del parámetro y del nombre del recurso no supera el límite de caracteres del nombre del recurso.

AWS Proton el aprovisionamiento falla si se superan estas cuotas.

Tipos de parámetros

Los siguientes tipos de parámetros están disponibles como referencia en los archivos AWS Proton IaC:

Parámetro de entrada

Los entornos y las instancias de servicio pueden tomar los parámetros de entrada que usted defina en un archivo de esquema que asocie a la plantilla de entorno o servicio. Puede consultar los parámetros de entrada de un recurso en el archivo IaC del recurso. Los archivos IaC de los componentes pueden hacer referencia a los parámetros de entrada de la instancia de servicio a la que está conectado el componente.

AWS Proton compara los nombres de los parámetros de entrada con el archivo de esquema y los compara con los parámetros a los que se hace referencia en los archivos IaC para introducir los valores de entrada que se proporcionan en un archivo de especificaciones durante el aprovisionamiento de recursos.

Parámetro de salida

Puede definir las salidas en cualquiera de sus archivos IaC. Una salida puede ser, por ejemplo, el nombre, el ID o el ARN de uno de los recursos que proporciona la plantilla, o puede ser una forma de pasar por una de las entradas de la plantilla. Puede hacer referencia a estas salidas en los archivos de IaC de otros recursos.

En los archivos CloudFormation IaC, defina los parámetros de salida en el `Outputs:` bloque. En un archivo iAC de Terraform, defina cada parámetro de salida mediante una sentencia. `output`

Parámetro de recurso

AWS Proton crea automáticamente los parámetros AWS Proton de los recursos. Estos parámetros exponen las propiedades del objeto AWS Proton de recurso. Un ejemplo de parámetro de recurso es`environment.name`.

Uso de AWS Proton parámetros en los archivos IaC

Para leer el valor de un parámetro en un archivo IaC, consulte el nombre del parámetro en el espacio de nombres del AWS Proton parámetro. En los archivos AWS CloudFormation IaC, se utiliza la sintaxis Jinja y se coloca el parámetro entre pares de llaves y comillas.

En la siguiente tabla se muestra la sintaxis de referencia para cada lenguaje de plantillas compatible, con un ejemplo.

Lenguaje de plantillas	Sintaxis	Ejemplo: entrada de entorno denominada «VPC»
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
Terraform	`var.parameter-name`	`var.environment.inputs.VPC` Definiciones de variables de Terraform generadas

> **Note**
>
> Si utiliza parámetros CloudFormation dinámicos en su archivo IaC, debe evitarlos para evitar errores de interpretación errónea de Jinja. Para obtener más información, consulte Solución de problemas de AWS Proton

En la siguiente tabla se muestran los nombres de los espacios de nombres de todos los parámetros de los recursos. AWS Proton Cada tipo de archivo de plantilla puede utilizar un subconjunto diferente del espacio de nombres de parámetros.

Archivo de plantilla	Tipo de parámetro	Nombre del parámetro	Descripción
Entorno	recurso	`environment.` **`name`**	Nombre del entorno
	input	`environment.inputs.` *`input-name`*	Entradas de entorno definidas por el esquema
Servicio	recurso	`environment.` **`name`** `environment.` **`account_id`**	Nombre e ID del entorno Cuenta de AWS

Archivo de plantilla	Tipo de parámetro	Nombre del parámetro	Descripción
	salida	`environment.outputs.` *`output-name`*	Salidas del archivo IaC del entorno
	recurso	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nombre del servicio y repositorio de códigos
	recurso	`service_instance.` **`name`**	Nombre de la instancia de servicio
	input	`service_instance.inputs.` *`input-name`*	Entradas de instancia de servicio definidas por el esquema
	recurso	`service_instance.components.default.` **`name`**	Nombre del componente predeterminado adjunto
	salida	`service_instance.components.default.outputs.` *`output-name`*	Las salidas del archivo IaC del componente predeterminado adjunto
Canalización	recurso	`service_instance.environment.` **`name`** `service_instance.environment.` **`account_id`**	Nombre e Cuenta de AWS ID del entorno de la instancia de servicio
	salida	`service_instance.environment.outputs.` *`output-name`*	Salidas del archivo IaC del entorno de la instancia de servicio

Archivo de plantilla	Tipo de parámetro	Nombre del parámetro	Descripción
Componente	input	`pipeline.inputs.` *`input-name`*	Entradas de canalización definidas por el esquema
	recurso	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nombre del servicio y repositorio de códigos
	input	`service_instance.inputs.` *`input-name`*	Entradas de instancia de servicio definidas por el esquema
	collection	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	Un conjunto de instancias de servicio que puede recorrer en bucle
	recurso	`environment.` **`name`** `environment.` **`account_id`**	Nombre del entorno e ID Cuenta de AWS de cuenta
	salida	`environment.outputs.` *`output-name`*	Salidas del archivo IaC del entorno
	recurso	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nombre del servicio y repositorio de códigos (componentes adjuntos)

Archivo de plantilla	Tipo de parámetro	Nombre del parámetro	Descripción
	recurso	`service_instance.` **`name`**	Nombre de la instancia de servicio (componentes adjuntos)
	input	`service_instance.inputs.` *`input-name`*	Entradas de instancia de servicio definidas por el esquema (componentes adjuntos)
	recurso	`component.` **`name`**	Nombre del componente

Para obtener más información y ejemplos, consulte los subtemas sobre los parámetros de los archivos de plantilla de IaC para los distintos tipos de recursos y lenguajes de plantillas.

Temas

- Detalles y ejemplos de los parámetros del archivo CloudFormation IaC de entorno
- Detalles y ejemplos de los parámetros del archivo CloudFormation IAC del servicio
- Detalles y ejemplos de los parámetros del archivo CloudFormation IaC del componente
- Filtros de parámetros para archivos CloudFormation IaC
- CodeBuild detalles y ejemplos de los parámetros de aprovisionamiento
- Detalles y ejemplos de los parámetros del archivo de infraestructura como código (IaC) de Terraform

Detalles y ejemplos de los parámetros del archivo CloudFormation IaC de entorno

Puede definir y hacer referencia a los parámetros de la infraestructura de su entorno como archivos de código (IaC). Para obtener una descripción detallada de AWS Proton los parámetros, los tipos de parámetros, el espacio de nombres de los parámetros y cómo utilizar los parámetros en los archivos de IaC, consulte. the section called “Parámetros”

Defina los parámetros del entorno

Puede definir los parámetros de entrada y salida para los archivos IaC del entorno.

- Parámetros de entrada: defina los parámetros de entrada del entorno en el archivo de esquema.

  La siguiente lista incluye ejemplos de parámetros de entrada del entorno para casos de uso típicos.

  - Valores CIDR de VPC
  - Configuración del balanceador de carga
  - Configuración de base de datos
  - Se ha agotado el tiempo de espera de un chequeo

  Como administrador, puede proporcionar valores para los parámetros de entrada al crear un entorno:

  - Utilice la consola para rellenar un formulario basado en un esquema que AWS Proton proporciona.
  - Utilice la CLI para proporcionar una especificación que incluya los valores.
- Parámetros de salida: defina las salidas del entorno en los archivos IaC de su entorno. A continuación, puede consultar estos resultados en los archivos IaC de otros recursos.

Lea los valores de los parámetros en los archivos IaC del entorno

Puede leer los parámetros relacionados con el entorno en los archivos IaC del entorno. Para leer el valor de un parámetro, haga referencia al nombre del parámetro en el espacio de nombres del AWS Proton parámetro.

- Parámetros de entrada: lee un valor de entrada del entorno haciendo referencia a él. `environment.inputs.input-name`
- Parámetros de recursos: lea los parámetros de los AWS Proton recursos haciendo referencia a nombres como. `environment.name`

> **Note**
>
> No hay parámetros de salida de otros recursos disponibles para los archivos IaC del entorno.

Ejemplos de archivos IaC de entorno y servicio con parámetros

El siguiente ejemplo muestra la definición y la referencia de los parámetros en un archivo IaC de un entorno. A continuación, el ejemplo muestra cómo se puede hacer referencia a los parámetros de salida del entorno definidos en el archivo IaC del entorno en un archivo IaC de servicio.

Example Medio ambiente CloudFormation (archivo IaC).

Tenga en cuenta lo siguiente en este ejemplo:

- El espacio de `environment.inputs.` nombres hace referencia a los parámetros de entrada del entorno.
- El `StoreInputValue` parámetro Amazon EC2 Systems Manager (SSM) concatena las entradas del entorno.
- La `MyEnvParameterValue` salida expone la misma concatenación de parámetros de entrada que un parámetro de salida. Tres parámetros de salida adicionales también exponen los parámetros de entrada de forma individual.
- Seis parámetros de salida adicionales exponen los recursos que proporciona el entorno.

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

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'environment.outputs' namespace, for example,
 service_instance.environment.outputs.ClusterName.
Outputs:
  MyEnvParameterValue:                                    # output definition
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                     # output definition
    Value: "{{ environment.inputs.my_sample_input }}"         #   input parameter
 reference
  MyOtherSampleInputValue:                                # output definition
```

```
    Value: "{{ environment.inputs.my_other_sample_input }}"   #   input parameter
 reference
  AnotherOptionalInputValue:                                  # output definition
    Value: "{{ environment.inputs.another_optional_input }}"  #   input parameter
 reference
  ClusterName:                                                # output definition
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'                                  #   provisioned resource
  ECSTaskExecutionRole:                                       # output definition
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'                 #   provisioned resource
  VpcId:                                                      # output definition
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'                                         #   provisioned resource
  PublicSubnetOne:                                            # output definition
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'                             #   provisioned resource
  PublicSubnetTwo:                                            # output definition
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'                             #   provisioned resource
  ContainerSecurityGroup:                                     # output definition
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'                      #   provisioned resource
```

Example Archivo CloudFormation IaC del servicio

El espacio de nombres `environment.outputs.` se refiere a las salidas del entorno de un archivo de IaC del entorno. Por ejemplo, el nombre `environment.outputs.ClusterName` lee el valor del parámetro de salida del `ClusterName` entorno.

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

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

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

  # The service_instance. The service is a resource which allows you to run multiple
```

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

Detalles y ejemplos de los parámetros del archivo CloudFormation IAC del servicio

Puede definir y hacer referencia a los parámetros de su infraestructura de servicios y canalizaciones como archivos de código (IaC). Para obtener una descripción detallada de AWS Proton los parámetros, los tipos de parámetros, el espacio de nombres de los parámetros y cómo utilizar los parámetros en los archivos de IaC, consulte. the section called “Parámetros”

Defina los parámetros de servicio

Puede definir los parámetros de entrada y salida para los archivos IaC del servicio.

- Parámetros de entrada: defina los parámetros de entrada de la instancia de servicio en el archivo de esquema.

  La siguiente lista incluye ejemplos de parámetros de entrada del servicio para casos de uso típicos.

  - Puerto
  - Tamaño de tarea
  - Imagen
  - Recuento deseado
  - Archivo de Docker
  - Comando de prueba unitaria

  Al crear un servicio, se proporcionan valores para los parámetros de entrada:

  - Utilice la consola para rellenar un formulario basado en un esquema que AWS Proton proporciona.
  - Utilice la CLI para proporcionar una especificación que incluya los valores.
- Parámetros de salida: defina las salidas de las instancias de servicio en sus archivos IaC de servicio. A continuación, puede consultar estos resultados en los archivos IaC de otros recursos.

Lea los valores de los parámetros en los archivos IaC del servicio

Puede leer los parámetros relacionados con el servicio y con otros recursos en los archivos IaC del servicio. Para leer el valor de un parámetro, haga referencia al nombre del parámetro en el espacio de nombres del AWS Proton parámetro.

- Parámetros de entrada: lee el valor de entrada de una instancia de servicio haciendo referencia a él. `service_instance.inputs.input-name`
- Parámetros de recursos: lea los parámetros de los AWS Proton recursos haciendo referencia a nombres como `service.name`service_instance.name, y. `environment.name`
- Parámetros de salida: lea las salidas de otros recursos haciendo referencia a `environment.outputs.output-name` o. `service_instance.components.default.outputs.output-name`

Ejemplo de archivo IaC de servicio con parámetros

El siguiente ejemplo es un fragmento de un archivo IaC de un servicio CloudFormation . El espacio de `environment.outputs.` nombres hace referencia a las salidas del archivo IaC del entorno. El espacio de `service_instance.inputs.` nombres hace referencia a los parámetros de entrada de la instancia de servicio. La `service_instance.name` propiedad hace referencia a un parámetro de AWS Proton recurso.

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

Detalles y ejemplos de los parámetros del archivo CloudFormation IaC del componente

Puede definir los parámetros de la infraestructura de componentes y hacer referencia a ellos como archivos de código (IaC). Para obtener una descripción detallada de AWS Proton los parámetros, los tipos de parámetros, el espacio de nombres de los parámetros y cómo utilizar los parámetros en los archivos de IaC, consulte. the section called “Parámetros” Para obtener más información sobre los componentes, consulte Componentes.

Definición de los parámetros de salida de los componentes

Puede definir los parámetros de salida en los archivos iAc de sus componentes. A continuación, puede hacer referencia a estas salidas en los archivos IaC de servicio.

> **Note**
>
> No puede definir las entradas para los archivos IaC de los componentes. Los componentes conectados pueden obtener entradas de la instancia de servicio a la que están conectados. Los componentes separados no tienen entradas.

Lea los valores de los parámetros en los archivos IaC de los componentes

Puede leer los parámetros relacionados con el componente y con otros recursos en los archivos IaC del componente. Para leer el valor de un parámetro, haga referencia al nombre del parámetro en el espacio de nombres del AWS Proton parámetro.

- Parámetros de entrada: lee el valor de entrada de una instancia de servicio adjunta haciendo referencia a él. `service_instance.inputs.`*`input-name`*
- Parámetros de AWS Proton recursos: lea los parámetros de los recursos haciendo referencia a nombres como`component.name`, `service.name`y `service_instance.name`, y. `environment.name`
- Parámetros de salida: lea las salidas del entorno mediante referencias`environment.outputs.`*`output-name`*.

Ejemplos de archivos IaC de componentes y servicios con parámetros

En el siguiente ejemplo, se muestra un componente que aprovisiona un bucket de Amazon Simple Storage Service (Amazon S3) y una política de acceso relacionada, y expone los nombres de recurso de Amazon (ARN) de ambos recursos como salidas de componentes. Una plantilla de IaC de servicio añade las salidas de los componentes como variables de entorno de contenedor de una tarea de Amazon Elastic Container Service (Amazon ECS) para que las salidas estén disponibles para el código que se ejecute en el contenedor, y añade la política de acceso del bucket al rol de la tarea. El nombre del bucket se basa en los nombres del entorno, el servicio, la instancia de servicio y el componente, lo que significa que el bucket está asociado a una instancia específica de la plantilla del componente que amplía una instancia de servicio específica. Los desarrolladores pueden crear varios componentes personalizados en función de esta plantilla de componentes para aprovisionar buckets de Amazon S3 para distintas instancias de servicio y necesidades funcionales.

El ejemplo muestra cómo se utiliza la sintaxis `{{ ... }}` de Jinja para hacer referencia a los parámetros de los componentes y de otros recursos en el archivo de IaC del servicio. Puede utilizar instrucciones `{% if ... %}` para añadir bloques de instrucciones solo cuando un componente esté conectado a la instancia del servicio. Las palabras clave `proton_cfn_*` son filtros que se pueden utilizar para borrar y formatear los valores de los parámetros de salida. Para obtener más información acerca de los filtros, consulte the section called “CloudFormation filtros de parámetros”.

Como administrador, usted crea el archivo de plantilla IaC del servicio.

Example CloudFormation repare el archivo IaC mediante un componente

```
# service/instance_infrastructure/cloudformation.yaml

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

```
  # ...

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

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

Como desarrollador, usted crea el archivo de plantilla IaC del componente.

Example fichero CloudFormation IaC del componente

```
# cloudformation.yaml

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

```
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

Al AWS Proton renderizar una AWS CloudFormation plantilla para la instancia de servicio y reemplazar todos los parámetros por valores reales, la plantilla podría tener un aspecto similar al siguiente archivo.

Example la instancia de servicio CloudFormation renderizó un archivo iAC

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

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

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

Filtros de parámetros para archivos CloudFormation IaC

Al hacer referencia a AWS Proton los parámetros de los archivos AWS CloudFormation IaC, puede utilizar los modificadores de Jinja, conocidos como filtros, para validar, filtrar y formatear los valores de los parámetros antes de insertarlos en la plantilla renderizada. Las validaciones de filtros son especialmente útiles cuando se hace referencia a los parámetros de salida de los componentes, ya que los desarrolladores se encargan de crear y conectar los componentes, y un administrador que utilice las salidas de los componentes en una plantilla de instancia de servicio podría querer comprobar su existencia y validez. Sin embargo, puede utilizar filtros en cualquier archivo IaC de Jinja.

En las siguientes secciones se describen y definen los filtros de parámetros disponibles y se proporcionan ejemplos. AWS Proton define la mayoría de estos filtros. El `default` filtro es un filtro integrado por Jinja.

Formatear las propiedades del entorno para las tareas de Amazon ECS

Declaración

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

Descripción

Este filtro da formato a una lista de salidas que se van a utilizar en una propiedad del entorno en la `ContainerDefinition` sección de una definición de tarea de Amazon Elastic Container Service (Amazon ECS).

`rawFalse`Configúrelo para validar también el valor del parámetro. En este caso, el valor debe coincidir con la expresión regular`^[a-zA-Z0-9_-]*$`. Si el valor no pasa esta validación, se produce un error en la representación de la plantilla.

Ejemplo

Con la siguiente plantilla de componentes personalizados:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
```

```
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Y la siguiente plantilla de servicio:

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

La plantilla de servicio renderizada es la siguiente:

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

Propiedades del entorno de formato para funciones Lambda

Declaración

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

Descripción

Este filtro formatea una lista de salidas que se utilizarán en una propiedad de entorno en la `Properties` sección de la definición de una AWS Lambda función.

`rawFalse`Configúrelo en para validar también el valor del parámetro. En este caso, el valor debe coincidir con la expresión regular`^[a-zA-Z0-9_-]*$`. Si el valor no pasa esta validación, se produce un error en la representación de la plantilla.

Ejemplo

Con la siguiente plantilla de componentes personalizados:

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

Y la siguiente plantilla de servicio:

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

La plantilla de servicio renderizada es la siguiente:

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

Extraiga los ARN de las políticas de IAM para incluirlos en las funciones de IAM

Declaración

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

Descripción

Este filtro formatea una lista de resultados que se van a utilizar en una ManagedPolicyArns propiedad en la `Properties` sección de una definición de rol AWS Identity and Access Management (IAM). El filtro utiliza la expresión regular `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` para extraer los ARN de política de IAM válidos de la lista de parámetros de salida. Puede utilizar este filtro para añadir las políticas de los valores de los parámetros de salida a una definición de función de IAM en una plantilla de servicio.

Ejemplo

Con la siguiente plantilla de componentes personalizados:

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

  # ...

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

```
    Description: "ARN of policy 2"
    Value: !Ref ExamplePolicy2
```

Y la siguiente plantilla de servicio:

```
Resources:

  # ...

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

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

La plantilla de servicio renderizada es la siguiente:

```
Resources:

  # ...

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

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

```
      # ...
```

Desinfecte los valores de las propiedades

Declaración

```
string # proton_cfn_sanitize # string
```

Descripción

Se trata de un filtro de uso general. Utilícelo para validar la seguridad del valor de un parámetro. El filtro valida que el valor coincide con la expresión regular `^[a-zA-Z0-9_-]*$` o que es un nombre de recurso de Amazon (ARN) válido. Si el valor no pasa esta validación, se produce un error en la representación de la plantilla.

Ejemplo

Con la siguiente plantilla de componentes personalizados:

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

- La siguiente referencia en una plantilla de servicio:

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

  Se representa de la siguiente manera:

  ```
  # ...
  ```

```
  This-is_valid_37
```

- La siguiente referencia en una plantilla de servicio:

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

  Resultados con el siguiente error de representación:

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

- La siguiente referencia en una plantilla de servicio:

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

  Se representa de la siguiente manera:

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

Provisión de valores predeterminados para referencias inexistentes

Descripción

El `default` filtro proporciona un valor predeterminado cuando no existe una referencia a un espacio de nombres. Úselo para escribir plantillas sólidas que puedan renderizarse sin errores incluso cuando falte el parámetro al que hace referencia.

Ejemplo

La siguiente referencia en una plantilla de servicio provoca un error en la representación de la plantilla si la instancia de servicio no tiene un componente adjunto definido directamente (predeterminado) o si el componente adjunto no tiene un nombre de salida`test`.

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

Para evitar este problema, agrega el `default` filtro.

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

CodeBuild detalles y ejemplos de los parámetros de aprovisionamiento

Puede definir los parámetros en las plantillas para los AWS Proton recursos CodeBuild basados y hacer referencia a estos parámetros en el código de aprovisionamiento. Para obtener una descripción detallada de AWS Proton los parámetros, los tipos de parámetros, el espacio de nombres de los parámetros y cómo utilizar los parámetros en los archivos de IaC, consulte. the section called “Parámetros”

> **Note**
>
> Puede utilizar el CodeBuild aprovisionamiento con entornos y servicios. En este momento, no puede aprovisionar componentes de esta forma.

Parámetros de entrada

Cuando crea un AWS Proton recurso, como un entorno o un servicio, proporciona valores para los parámetros de entrada que se definen en el archivo de esquema de la plantilla. Cuando el recurso que cree los utilizaAprovisionamiento de CodeBuild, AWS Proton renderiza estos valores de entrada en un archivo de entrada. El código de aprovisionamiento puede importar y obtener valores de parámetros de este archivo.

Para ver un ejemplo de CodeBuild plantilla, consultethe section called “CodeBuild paquete”. Para obtener más información sobre los archivos de manifiesto, consulte the section called “Manifieste y resuma”.

El siguiente ejemplo es un archivo de entrada JSON generado durante el aprovisionamiento CodeBuild basado de una instancia de servicio.

Ejemplo: usar el AWS CDK con aprovisionamiento CodeBuild

```
{
  "service_instance": {
    "name": "my-service-staging",
    "inputs": {
```

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

Parámetros de salida

Para volver a comunicar los resultados del aprovisionamiento de recursos AWS Proton, el código de aprovisionamiento puede generar un archivo JSON `proton-outputs.json` con el nombre de los valores de los parámetros de salida definidos en el archivo de esquema de la plantilla. Por ejemplo, el `cdk deploy` comando tiene el `--outputs-file` argumento que le indica que genere un archivo JSON con AWS CDK los resultados del aprovisionamiento. Si tu recurso usa el AWS CDK, especifica el siguiente comando en el manifiesto de la CodeBuild plantilla:

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

AWS Proton busca este archivo JSON. Si el archivo existe después de que el código de aprovisionamiento se haya completado correctamente, AWS Proton lee los valores de los parámetros de salida del mismo.

Detalles y ejemplos de los parámetros del archivo de infraestructura como código (IaC) de Terraform

Puede incluir variables de entrada de Terraform en `variable.tf` los archivos de su paquete de plantillas. También puedes crear un esquema para crear variables AWS Proton administradas. AWS Proton crea una variable `.tf files` a partir del archivo de esquema. Para obtener más información, consulte the section called “Archivos iAC de Terraform”.

Para hacer referencia a AWS Proton las variables definidas por el esquema en su infraestructura `.tf files`, utilice los espacios de AWS Proton nombres que se muestran en la tabla Parámetros y espacios de nombres de Terraform IaC. Por ejemplo, puede utilizar `var.environment.inputs.vpc_cidr`. Entre comillas, ponga estas variables entre corchetes simples y añada un signo de dólar delante del primer corchete (por ejemplo,). `“${var.environment.inputs.vpc_cidr}”`

El siguiente ejemplo muestra cómo utilizar los espacios de nombres para incluir AWS Proton parámetros en un entorno. `.tf file`

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

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

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

AWS Proton infraestructura como archivos de código

Las partes principales del paquete de plantillas son los archivos de infraestructura como código (IaC) que definen los recursos y propiedades de la infraestructura que desea aprovisionar. AWS CloudFormation y otras infraestructuras, ya que los motores de código utilizan estos tipos de archivos para aprovisionar los recursos de infraestructura.

> **Note**
>
> Un archivo IaC también se puede utilizar independientemente de los paquetes de plantillas, como entrada directa a los componentes definidos directamente. Para obtener más información sobre los componentes, consulte Componentes.

AWS Proton actualmente admite dos tipos de archivos IaC:

- CloudFormationarchivos: se utilizan para el aprovisionamiento AWS gestionado. AWS Proton utiliza Jinja sobre el formato de archivo de CloudFormation plantilla para la parametrización.
- Archivos HCL de Terraform: se utilizan para el aprovisionamiento autogestionado. El HCL admite la parametrización de forma nativa.

No puede aprovisionar AWS Proton recursos mediante una combinación de métodos de aprovisionamiento. Debe usar uno u otro. No puede implementar un servicio de AWS aprovisionamiento gestionado en un entorno de aprovisionamiento autogestionado ni viceversa.

Para obtener más información, consulte the section called “Métodos de aprovisionamiento”, Entornos, Servicios y Componentes.

AWS CloudFormation Archivos iAC

Aprenda a usar la AWS CloudFormation infraestructura como archivos de código con. AWS Proton AWS CloudFormation es un servicio de infraestructura como código (IaC) que le ayuda a modelar y configurar sus AWS recursos. Los recursos de la infraestructura se definen en plantillas, utilizando Jinja sobre el formato del archivo de CloudFormation plantilla para la parametrización. AWS Proton expande los parámetros y renderiza la plantilla completa. CloudFormation CloudFormation aprovisiona los recursos definidos como una CloudFormation pila. Para obtener más información, consulte Contenido de AWS CloudFormationla Guía del AWS CloudFormation usuario.

AWS Proton admite el AWS aprovisionamiento gestionado para CloudFormation iAC.

Comience con su propia infraestructura existente en forma de archivos de código

Puede adaptar su propia infraestructura existente como archivos de código (IaC) para utilizarlos con AWS Proton ellos.

Los siguientes AWS CloudFormation ejemplos, el ejemplo 1 y el ejemplo 2, representan sus propios archivos CloudFormation IaC existentes. CloudFormation puede usar estos archivos para crear dos CloudFormation pilas diferentes.

En el ejemplo 1, el archivo CloudFormation IaC está configurado para aprovisionar la infraestructura que se compartirá entre las aplicaciones contenedoras. En este ejemplo, se agregan parámetros de entrada para que pueda usar el mismo archivo iAC para crear varios conjuntos de infraestructura aprovisionada. Cada conjunto puede tener nombres diferentes junto con un conjunto diferente de valores CIDR de VPC y subred. Como administrador o desarrollador, usted proporciona valores para estos parámetros cuando utiliza un archivo IaC para aprovisionar recursos de infraestructura. CloudFormation Para su comodidad, estos parámetros de entrada están marcados con comentarios y se hace referencia a ellos varias veces en el ejemplo. Las salidas se definen al final de la plantilla. Se puede hacer referencia a ellos en otros archivos CloudFormation IaC.

En el ejemplo 2, el archivo CloudFormation IaC está configurado para implementar una aplicación en la infraestructura aprovisionada en el ejemplo 1. Los parámetros se comentan para su comodidad.

Ejemplo 1: archivo CloudFormation iAC

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

```
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock:
        Ref: 'VpcCIDR'

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

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

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
```

```
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

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

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

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
```

```
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values will be available to other templates to use.
Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSCluster"
  ECSTaskExecutionRole:                                  # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSTaskExecutionRole"
  VpcId:                                                 # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-VPC"
  PublicSubnetOne:                                       # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetOne"
  PublicSubnetTwo:                                       # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetTwo"
  ContainerSecurityGroup:                                # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ContainerSecurityGroup"
```

Ejemplo 2: archivo CloudFormation iAC

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Parameters:
    ContainerPortInput:  # input parameter
        Description: The port to route traffic to
        Type: Number
        Default: 80
    TaskCountInput:      # input parameter
        Description: The default number of Fargate tasks you want running
        Type: Number
        Default: 1
    TaskSizeInput:       # input parameter
        Description: The size of the task you want to run
        Type: String
        Default: x-small
    ContainerImageInput: # input parameter
        Description: The name/url of the container image
        Type: String
        Default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
    TaskNameInput:       # input parameter
        Description: Name for your task
        Type: String
        Default: "my-fargate-instance"
    StackName:           # input parameter
        Description: Name of the environment stack to deploy to
        Type: String
        Default: "my-fargate-environment"
Mappings:
  TaskSizeMap:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
```

```
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName:
        Ref: 'TaskNameInput' # input parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref 'TaskNameInput'
      Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
      Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn:
        Fn::ImportValue:
          !Sub "${StackName}-ECSTaskExecutionRole"    # output parameter from another
 CloudFormation template
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: !Ref 'TaskNameInput'


  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: !Ref 'TaskNameInput'
      Cluster:
        Fn::ImportValue:
          !Sub "${StackName}-ECSCluster"  # output parameter from another
 CloudFormation template
      LaunchType: FARGATE
      DeploymentConfiguration:
```

```
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: !Ref 'TaskCountInput'
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - Fn::ImportValue:
                !Sub "${StackName}-ContainerSecurityGroup"    # output parameter from
another CloudFormation template
          Subnets:
            - Fn::ImportValue:r CloudFormation template
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: !Ref 'TaskNameInput'
          Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
          Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
          Image: !Ref 'ContainerImageInput'             # input parameter
          PortMappings:
            - ContainerPort: !Ref 'ContainerPortInput'  # input parameter

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

  # A target group. This is used for keeping track of all the tasks, and
  # what IP addresses / port numbers they have. You can query it yourself,
  # to use the addresses yourself, but most often this target group is just
  # connected to an application load balancer, or network load balancer, so
  # it can automatically distribute traffic across all the targets.
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
```

```
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Name: !Ref 'TaskNameInput'
      Port: !Ref 'ContainerPortInput'
      Protocol: HTTP
      UnhealthyThresholdCount: 2
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"    # output parameter from another CloudFormation
template

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

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      MinCapacity: 1
```

```
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"  # output parameter from another
 CloudFormation template
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
```

```
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60

  # Create alarms to trigger these policies
  LowCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - low-cpu
            - !Ref 'TaskNameInput'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "Low CPU utilization for service"
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
```

```
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 20
      ComparisonOperator: LessThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleDownPolicy

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

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
```

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

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

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

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
```

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

Puede adaptar estos archivos para usarlos con AWS Proton.

Utilice su infraestructura como código para AWS Proton

Con pequeñas modificaciones, puede utilizar el ejemplo 1 como un archivo de infraestructura como código (IaC) para un paquete de plantillas de entorno que se AWS Proton utiliza para implementar un entorno (como se muestra en el ejemplo 3).

En lugar de utilizar los CloudFormation parámetros, utiliza la sintaxis de Jinja para hacer referencia a los parámetros que ha definido en un archivo de esquema basado en una API abierta. Estos parámetros de entrada se comentan para su comodidad y se hace referencia a ellos varias veces en el archivo de IaC. De esta forma, AWS Proton puede auditar y comprobar los valores de los parámetros. También puede hacer coincidir e insertar los valores de los parámetros de salida de un archivo de IaC con los parámetros de otro archivo de IaC.

Como administrador, puede añadir el espacio de AWS Proton `environment.inputs.` nombres a los parámetros de entrada. Al hacer referencia a las salidas de un archivo IaC del entorno en un archivo IaC de un servicio, puede añadir el espacio de `environment.outputs.` nombres a las salidas (por ejemplo,). `environment.outputs.ClusterName` Por último, debe rodearlos con corchetes y comillas.

Con estas modificaciones, sus archivos CloudFormation iAC pueden ser utilizados por. AWS Proton

Ejemplo 3: infraestructura de AWS Proton entorno como archivo de código

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery prefixes.
```

```
Mappings:
  # The VPC and subnet configuration is passed in via the environment spec.
  SubnetConfig:
    VPC:
      CIDR: '{{ environment.inputs.vpc_cidr}}'        # input parameter
    PublicOne:
      CIDR: '{{ environment.inputs.subnet_one_cidr}}' # input parameter
    PublicTwo:
      CIDR: '{{ environment.inputs.subnet_two_cidr}}' # input parameter
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock: !FindInMap ['SubnetConfig', 'VPC', 'CIDR']

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

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

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
```

```
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

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

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

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
```

```
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

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

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

Los archivos IaC del ejemplo 1 y el ejemplo 3 producen pilas ligeramente diferentes CloudFormation . Los parámetros se muestran de forma diferente en los archivos de plantillas de pila. El archivo de plantilla de CloudFormation pila del ejemplo 1 muestra las etiquetas de los parámetros (claves) en la vista de plantilla de pila. El archivo de plantilla de pila de AWS Proton CloudFormation infraestructura del ejemplo 3 muestra los valores de los parámetros. AWS Proton los parámetros de entrada no aparecen en la vista de parámetros de la CloudFormation pila de la consola.

En el ejemplo 4, el archivo IaC del AWS Proton servicio corresponde al ejemplo 2.

Ejemplo 4: archivo IaC de la instancia de AWS Proton servicio

```
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}"
```

En el ejemplo 5, el archivo IaC de AWS Proton canalización aprovisiona la infraestructura de canalización para admitir las instancias de servicio aprovisionadas en el ejemplo 4.

Ejemplo 5: archivo IaC AWS Proton de service pipeline

```
Resources:
  ECRRepo:
    Type: AWS::ECR::Repository
    DeletionPolicy: Retain
  BuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: true
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: repo_name
          Type: PLAINTEXT
          Value: !Ref ECRRepo
        - Name: service_name
          Type: PLAINTEXT
          Value: '{{ service.name }}'    # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - PublishRole
          - Arn
      Source:
```

```
        BuildSpec:
          Fn::Join:
            - ""
            - - >-
                {
                  "version": "0.2",
                  "phases": {
                    "install": {
                      "runtime-versions": {
                        "docker": 18
                      },
                      "commands": [
                        "pip3 install --upgrade --user awscli",
                        "echo
 'f6bd1536a743ab170b35c94ed4c7c4479763356bd543af5d391122f4af852460  yq_linux_amd64' >
 yq_linux_amd64.sha",
                        "wget https://github.com/mikefarah/yq/releases/download/3.4.0/
yq_linux_amd64",
                        "sha256sum -c yq_linux_amd64.sha",
                        "mv yq_linux_amd64 /usr/bin/yq",
                        "chmod +x /usr/bin/yq"
                      ]
                    },
                    "pre_build": {
                      "commands": [
                        "cd $CODEBUILD_SRC_DIR",
                        "$(aws ecr get-login --no-include-email --region
 $AWS_DEFAULT_REGION)",
                        "{{ pipeline.inputs.unit_test_command }}",    # input parameter
                      ]
                    },
                    "build": {
                      "commands": [
                        "IMAGE_REPO_NAME=$repo_name",
                        "IMAGE_TAG=$CODEBUILD_BUILD_NUMBER",
                        "IMAGE_ID=
              - Ref: AWS::AccountId
              - >-
                .dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG",
                        "docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG -f
 {{ pipeline.inputs.dockerfile }} .",     # input parameter
                        "docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_ID;",
                        "docker push $IMAGE_ID"
```

```
                    ]
                  },
                  "post_build": {
                    "commands": [
                      "aws proton --region $AWS_DEFAULT_REGION get-service --name
 $service_name | jq -r .service.spec > service.yaml",
                      "yq w service.yaml 'instances[*].spec.image' \"$IMAGE_ID\" >
 rendered_service.yaml"
                    ]
                  }
                },
                "artifacts": {
                  "files": [
                    "rendered_service.yaml"
                  ]
                }
              }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% for service_instance in service_instances %}
  Deploy{{loop.index}}Project:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: false
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: service_name
          Type: PLAINTEXT
          Value:  '{{service.name}}'          # resource parameter
        - Name: service_instance_name
          Type: PLAINTEXT
          Value: '{{service_instance.name}}'  # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - DeploymentRole
          - Arn
```

```
      Source:
        BuildSpec: >-
          {
            "version": "0.2",
            "phases": {
              "build": {
                "commands": [
                  "pip3 install --upgrade --user awscli",
                  "aws proton --region $AWS_DEFAULT_REGION update-service-instance
 --deployment-type CURRENT_VERSION --name $service_instance_name --service-name
 $service_name --spec file://rendered_service.yaml",
                  "aws proton --region $AWS_DEFAULT_REGION wait service-instance-
deployed --name $service_instance_name --service-name $service_name"
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
```

```
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
```

```
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
```

```
              - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole

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

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

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

```
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
```

```
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
      Version: "2012-10-17"
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineArtifactsBucket:
  Type: AWS::S3::Bucket
  Properties:
    VersioningConfiguration:
      Status: Enabled
    BucketEncryption:
      ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            KMSMasterKeyID:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
```

```
                - Arn
            SSEAlgorithm: aws:kms
    PublicAccessBlockConfiguration:
      BlockPublicAcls: true
      BlockPublicPolicy: true
      IgnorePublicAcls: true
      RestrictPublicBuckets: true
  UpdateReplacePolicy: Retain
  DeletionPolicy: Retain
PipelineArtifactsBucketEncryptionKeyAlias:
  Type: AWS::KMS::Alias
  Properties:
    AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'
    TargetKeyId:
      Fn::GetAtt:
        - PipelineArtifactsBucketEncryptionKey
        - Arn
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
          Effect: Allow
          Principal:
            Service: codepipeline.amazonaws.com
      Version: "2012-10-17"
PipelineRoleDefaultPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Statement:
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
            - s3:DeleteObject*
            - s3:PutObject*
            - s3:Abort*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
```

```
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action: codestar-connections:*
          Effect: Allow
          Resource: "*"
        - Action: sts:AssumeRole
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineBuildCodePipelineActionRole
              - Arn
        - Action: sts:AssumeRole
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineDeployCodePipelineActionRole
              - Arn
      Version: "2012-10-17"
    PolicyName: PipelineRoleDefaultPolicy
    Roles:
      - Ref: PipelineRole
Pipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Fn::GetAtt:
        - PipelineRole
```

```
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'
                FullRepositoryId: '{{ service.repository_id }}'
                BranchName: '{{ service.branch_name }}'
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
```

```
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
```

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

```
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"

                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
```

```
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
```

```
            - Ref: BuildProject
            - -*
        - Action:
            - ecr:GetAuthorizationToken
          Effect: Allow
          Resource: "*"
        - Action:
            - ecr:BatchCheckLayerAvailability
            - ecr:CompleteLayerUpload
            - ecr:GetAuthorizationToken
            - ecr:InitiateLayerUpload
            - ecr:PutImage
            - ecr:UploadLayerPart
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - ECRRepo
              - Arn
        - Action:
            - proton:GetService
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
            - s3:DeleteObject*
            - s3:PutObject*
            - s3:Abort*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
```

```
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: PublishRoleDefaultPolicy
    Roles:
      - Ref: PublishRole

DeploymentRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
          Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
      Version: "2012-10-17"
DeploymentRoleDefaultPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Statement:
        - Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
          Effect: Allow
          Resource:
            - Fn::Join:
```

```
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":logs:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":logs:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/Deploy*Project
                  - -*
          - Action:
              - proton:UpdateServiceInstance
              - proton:GetServiceInstance
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
```

```
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
              - kms:Enable*
              - kms:List*
              - kms:Put*
              - kms:Update*
```

```
            - kms:Revoke*
            - kms:Disable*
            - kms:Get*
            - kms:Delete*
            - kms:ScheduleKeyDeletion
            - kms:CancelKeyDeletion
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
```

```
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
      Version: "2012-10-17"
  UpdateReplacePolicy: Delete
  DeletionPolicy: Delete
PipelineArtifactsBucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketEncryption:
      ServerSideEncryptionConfiguration:
```

```
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'     # resource
parameter
      TargetKeyId:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
```

```
            - s3:PutObject*
            - s3:Abort*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action: codestar-connections:*
          Effect: Allow
          Resource: "*"
        - Action: sts:AssumeRole
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineBuildCodePipelineActionRole
              - Arn
        - Action: sts:AssumeRole
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineDeployCodePipelineActionRole
              - Arn
      Version: "2012-10-17"
    PolicyName: PipelineRoleDefaultPolicy
    Roles:
      - Ref: PipelineRole
Pipeline:
```

```
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'   # resource
parameter
                FullRepositoryId: '{{ service.repository_id }}'            # resource
parameter
                BranchName: '{{ service.branch_name }}'                    # resource
parameter
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
```

```
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'         # resource parameter
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
```

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

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

CodeBuild paquete de plantillas de aprovisionamiento

Con el CodeBuild aprovisionamiento, en lugar de utilizar plantillas de IaC para renderizar los archivos de IaC y ejecutarlos mediante un motor de aprovisionamiento de IaC, AWS Proton basta con ejecutar los comandos del shell. Para ello, AWS Proton crea un AWS CodeBuild proyecto para el entorno, en la cuenta de entorno, e inicia un trabajo para ejecutar los comandos correspondientes a cada AWS Proton creación o actualización de recursos. Al crear un paquete de plantillas, se proporciona un manifiesto en el que se especifican los comandos de aprovisionamiento y desaprovisionamiento de la infraestructura, así como los programas, scripts y otros archivos que puedan necesitar estos comandos. Los comandos pueden leer las entradas que proporciona AWS Proton , y son responsables de aprovisionar o desaprovisionar la infraestructura y generar valores de salida.

El manifiesto también especifica cómo se AWS Proton debe representar el archivo de entrada en el que el código puede introducir y del que puede obtener valores de entrada. Se puede renderizar en

JSON o HCL. Para obtener más información sobre los parámetros de entrada, consultethe section called “CodeBuild parámetros de aprovisionamiento”. Para obtener más información sobre los archivos de manifiesto, consulte the section called “Manifieste y resuma”.

> **Note**
>
> Puede utilizar el CodeBuild aprovisionamiento con entornos y servicios. En este momento, no puede aprovisionar componentes de esta forma.

Ejemplo: usar el AWS CDK con aprovisionamiento CodeBuild

Como ejemplo del uso del CodeBuild aprovisionamiento, puedes incluir un código que lo utilice AWS Cloud Development Kit (AWS CDK) para aprovisionar (implementar) y desaprovisionar (destruir) AWS los recursos, y un manifiesto que instale la CDK y ejecute el código de la CDK.

En las siguientes secciones se enumeran ejemplos de archivos que puede incluir en un paquete de plantillas de CodeBuild aprovisionamiento que aprovisione un entorno mediante. AWS CDK

Manifiesto

El siguiente archivo de manifiesto especifica el CodeBuild aprovisionamiento e incluye los comandos necesarios para instalarlo y usarlo AWS CDK, procesar el archivo de salida y generar informes sobre los resultados. AWS Proton

Example infrastructure/manifest.yaml

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
```

```
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Esquema

El siguiente archivo de esquema define los parámetros del entorno. AWS CDK El código puede hacer referencia a los valores de estos parámetros durante la implementación.

Example schema/schema.yaml

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

AWS CDK archivos

Los siguientes archivos son un ejemplo de un proyecto de CDK de Node.js.

Example infraestructura/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironmente": "bin/ProtonEnvironment.js"
  },
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk"
  },
  "devDependencies": {
    "@types/jest": "^28.1.7",
    "@types/node": "18.7.6",
    "jest": "^28.1.3",
    "ts-jest": "^28.0.8",
    "aws-cdk": "2.37.1",
    "ts-node": "^10.9.1",
    "typescript": "~4.7.4"
  },
  "dependencies": {
    "aws-cdk-lib": "2.37.1",
    "constructs": "^10.1.77",
    "source-map-support": "^0.5.21"
  }
}
```

Example infraestructura/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2018",
    "module": "commonjs",
    "lib": [
      "es2018"
    ],
    "declaration": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
```

```
    "noImplicitThis": true,
    "alwaysStrict": true,
    "noUnusedLocals": false,
    "noUnusedParameters": false,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": false,
    "inlineSourceMap": true,
    "inlineSources": true,
    "experimentalDecorators": true,
    "strictPropertyInitialization": false,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "typeRoots": [
      "./node_modules/@types"
    ]
  },
  "exclude": [
    "node_modules",
    "cdk.out"
  ]
}
```

Example infraestructura/cdk.json

```
{
  "app": "npx ts-node --prefer-ts-exts bin/ProtonEnvironment.ts",
  "outputsFile": "proton-outputs.json",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
      "yarn.lock",
      "node_modules",
      "test"
    ]
  },
```

```
  "context": {
    "@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId": true,
    "@aws-cdk/core:stackRelativeExports": true,
    "@aws-cdk/aws-rds:lowercaseDbIdentifier": true,
    "@aws-cdk/aws-lambda:recognizeVersionProps": true,
    "@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2_2021": true,
    "@aws-cdk-containers/ecs-service-extensions:enableDefaultLogDriver": true,
    "@aws-cdk/aws-ec2:uniqueImdsv2TemplateName": true,
    "@aws-cdk/core:target-partitions": [
      "aws",
      "aws-cn"
    ]
  }
}
```

Example infraestructura/bin/.ts ProtonEnvironment

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { ProtonEnvironmentStack } from '../lib/ProtonEnvironmentStack';

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

Example infraestructura/lib/ .ts ProtonEnvironmentStack

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

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

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

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

Archivo de entrada renderizado

Cuando crea un entorno mediante una plantilla de aprovisionamiento CodeBuild basada en ella, representa un archivo de entrada con AWS Proton los valores de los parámetros de entrada que haya proporcionado. El código puede hacer referencia a estos valores. El siguiente archivo es un ejemplo de un archivo de entrada renderizado.

Example infrastructure/proton-inputs.json

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

Archivos iAC de Terraform

Aprenda a usar la infraestructura de Terraform como archivos de código (IaC) con. AWS ProtonTerraform es un motor iAC de código abierto muy utilizado que fue desarrollado por. HashiCorp Los módulos de Terraform se desarrollan en HashiCorp el lenguaje HCL y son compatibles con varios proveedores de infraestructuras de back-end, incluido Amazon Web Services.

AWS Proton admite el aprovisionamiento autogestionado para Terraform IaC.

Para ver un ejemplo completo de un repositorio de aprovisionamiento que responde a las solicitudes de cambios e implementa el aprovisionamiento de infraestructura, consulta la plantilla de automatización de Terraform OpenSource GitHub Actions para ver más información. AWS Proton GitHub

Cómo funciona el aprovisionamiento autogestionado con los archivos del paquete de plantillas iAC de Terraform:

1. Cuando crea un entorno a partir de paquetes de plantillas de Terraform, AWS Proton compila sus `.tf` archivos con parámetros de entrada o de consola. `spec file`
2. Realiza una solicitud de selección para fusionar los archivos IaC compilados con el repositorio en el que te has registrado. AWS Proton
3. Si se aprueba la solicitud, AWS Proton espera al estado de aprovisionamiento que nos proporciones.
4. Si se rechaza la solicitud, se cancela la creación del entorno.
5. Si se agota el tiempo de espera de la solicitud de extracción, la creación del entorno no se completará.

AWS Proton teniendo en cuenta las consideraciones de Terraform IaC:

- AWS Proton no administra su aprovisionamiento de Terraform.
- Debe registrar un repositorio de aprovisionamiento con. AWS Proton AWS Proton realiza solicitudes de incorporación de cambios en este repositorio.
- Debe crear una CodeStar conexión para conectarse AWS Proton con su repositorio de aprovisionamiento.
- Para aprovisionar a partir de archivos IaC AWS Proton compilados, debes responder a las solicitudes de AWS Proton extracción. AWS Proton realiza solicitudes de extracción después de las acciones de creación y actualización del entorno y el servicio. Para obtener más información, consulte Entornos de AWS Proton y Servicios de AWS Proton.
- Para aprovisionar una canalización a partir de archivos IaC AWS Proton compilados, debe crear un repositorio de canalizaciones de CI/CD.
- La automatización del aprovisionamiento basada en solicitudes de extracción debe incluir pasos para notificar cualquier cambio en el estado AWS Proton de los recursos AWS Proton aprovisionados. Puedes usar la AWS Proton NotifyResourceDeploymentStatusChange API.
- No puede implementar servicios, canalizaciones y componentes creados a partir de archivos CloudFormation IaC en entornos creados a partir de archivos IaC de Terraform.
- No puede implementar servicios, canalizaciones y componentes creados a partir de archivos iAC de Terraform en entornos creados a partir de archivos iAC. CloudFormation

Al preparar los archivos iAC de Terraform AWS Proton, debe adjuntar espacios de nombres a las variables de entrada, como se muestra en los siguientes ejemplos. Para obtener más información, consulte Parámetros.

Ejemplo 1: AWS Proton archivo iAC de Terraform de entorno

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

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

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

Infraestructura compilada como código

Al crear un entorno o un servicio, AWS Proton compila la infraestructura como archivos de código con consola o `spec file` entradas. Crea `proton.`*`resource-type`*`.variables.tf` y `proton.auto.tfvars.json` archiva tus entradas para que Terraform pueda utilizarlas, como se

muestra en los siguientes ejemplos. Estos archivos se encuentran en un repositorio específico en una carpeta que coincide con el nombre del entorno o de la instancia de servicio.

En el ejemplo se muestra cómo se AWS Proton incluyen las etiquetas en la definición y los valores de las variables, y cómo se pueden propagar estas AWS Proton etiquetas a los recursos aprovisionados. Para obtener más información, consulte the section called “Propagación de etiquetas a recursos aprovisionados”.

Ejemplo 2: archivos IaC compilados para un entorno denominado «dev».

dev/environment.tf:

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

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

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

dev/proton.environment.variables.tf:

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

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

dev/proton.auto.tfvars.json:

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

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

Rutas de repositorio

AWS Proton utiliza las entradas de la consola o de las especificaciones del entorno o las acciones de creación de servicios para encontrar el repositorio y la ruta en la que se encuentran los archivos IaC compilados. Los valores de entrada se pasan a parámetros de entrada espaciados por nombres.

AWS Proton admite dos diseños de rutas de repositorio. En los siguientes ejemplos, las rutas se nombran según los parámetros de recursos con espacios de nombres de dos entornos. Cada entorno tiene instancias de servicio de dos servicios y las instancias de servicio de uno de los servicios tienen componentes definidos directamente.

Tipo de recurso	Nombre (parámetro)	=	Nombre del recurso
Entorno	`environment.name`		`"env-prod"`
Entorno	`environment.name`		`"env-staged"`
Servicio	`service.name`		`"service-one"`
Instancia de servicio	`service_instance.name`		`"instance-one-prod"`
Instancia de servicio	`service_instance.name`		`"instance-one-staged"`
Servicio	`service.name`	=	`"service-two"`
Instancia de servicio	`service_instance.name`		`"instance-two-prod"`
Componente	`service_instance.components.default.name`		`"component-prod"`
Instancia de servicio	`service_instance.name`		`"instance-two-staged"`
Componente	`service_instance.components.default.name`		`"component-staged"`

Layout 1

Si AWS Proton encuentra el repositorio especificado con una `environments` carpeta, crea una carpeta que incluye los archivos IaC compilados y se nombra con. `environment.name`

Si AWS Proton encuentra el repositorio especificado con una `environments` carpeta que contiene un nombre de carpeta que coincide con el nombre de un entorno compatible con una instancia de servicio, crea una carpeta que incluye los archivos IaC de la instancia compilada y recibe el nombre de. `service_instance.name`

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

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

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

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

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

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

            /service-two-instance-two-staged  # instance folder
```

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

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

Layout 2

Si AWS Proton encuentra el repositorio especificado sin una `environments` carpeta, crea una `environment.name` carpeta en la que ubica los archivos IaC del entorno compilado.

Si AWS Proton encuentra el repositorio especificado con un nombre de carpeta que coincide con el nombre de un entorno compatible con una instancia de servicio, crea una `service_instance.name` carpeta en la que ubica los archivos IaC de la instancia compilada.

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

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

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

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

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

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

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

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

Archivo de esquema

Como administrador, cuando utilizas la sección de modelos de datos de API abierta (esquemas) para definir un archivo YAML de esquema de parámetros para tu paquete de plantillas, AWS Proton puedes validar las entradas de los valores de los parámetros según los requisitos que hayas definido en tu esquema.

Para obtener más información sobre los formatos y las palabras clave disponibles, consulte la sección de objetos Schema de OpenAPI.

Requisitos de esquema para los paquetes de plantillas de entorno

El esquema debe seguir la sección de modelos de datos (esquemas) de OpenAPI en formato YAML. También debe formar parte del paquete de plantillas de entorno.

En el esquema de su entorno, debe incluir los encabezados formateados para comprobar que está utilizando la sección de modelos de datos (esquemas) de la API abierta. En los siguientes ejemplos de esquemas de entorno, estos encabezados aparecen en las tres primeras líneas.

`environment_input_type`Debe incluirse y definirse con un nombre que usted proporcione. En los ejemplos siguientes, esto se define en la línea 5. Al definir este parámetro, lo asocia a un recurso AWS Proton del entorno.

Para seguir el modelo de esquema de API abierta, debe incluir`types`. En el siguiente ejemplo, se trata de la línea 6.

A continuacióntypes, debe definir un environment_input_type tipo. Defina los parámetros de entrada de su entorno como propiedades delenvironment_input_type. Debe incluir al menos una propiedad con un nombre que coincida con al menos un parámetro que aparezca en la infraestructura del entorno como un archivo de código (IaC) asociado al esquema.

Al crear un entorno y proporcionar valores de parámetros personalizados, AWS Proton utiliza el archivo de esquema para compararlos, validarlos e incorporarlos a los parámetros entre corchetes del archivo IaC asociado. CloudFormation Para cada propiedad (parámetro), introduzca una y. name type Si lo desea, proporcione también un descriptiondefault, ypattern.

Los parámetros definidos para el siguiente ejemplo de esquema de plantilla de entorno estándar incluyen vpc_cidrsubnet_one_cidr, y subnet_two_cidr con la default palabra clave y los valores predeterminados. Al crear un entorno con este esquema de paquete de plantillas de entorno, puede aceptar los valores predeterminados o proporcionar los suyos propios. Si un parámetro no tiene un valor predeterminado y aparece como una required propiedad (parámetro), debe proporcionarle valores al crear un entorno.

El segundo ejemplo de esquema de plantilla de entorno estándar muestra el required parámetromy_other_sample_input.

Puede crear un esquema para dos tipos de plantillas de entorno. Para obtener más información, consulte Registro y publicación de plantillas.

- Plantillas de entorno estándar

  En el siguiente ejemplo, se define un tipo de entrada de entorno con una descripción y propiedades de entrada. Este ejemplo de esquema se puede utilizar con el archivo AWS Proton CloudFormation IaC que se muestra en el ejemplo 3.

  Ejemplo de esquema para una plantilla de entorno estándar:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "PublicEnvironmentInput"
  types:                          # required
    # defined by administrator
    PublicEnvironmentInput:
      type: object
      description: "Input properties for my environment"
```

```
    properties:
      vpc_cidr:                  # parameter
        type: string
        description: "This CIDR range for your VPC"
        default: 10.0.0.0/16
        pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
      subnet_one_cidr:           # parameter
        type: string
        description: "The CIDR range for subnet one"
        default: 10.0.0.0/24
        pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
      subnet_two_cidr:           # parameter
        type: string
        description: "The CIDR range for subnet one"
        default: 10.0.1.0/24
        pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
```

Ejemplo de esquema para una plantilla de entorno estándar que incluye un `required` parámetro:

```
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
```

- Plantillas de entorno gestionadas por el cliente

En el siguiente ejemplo, el esquema solo incluye una lista de salidas que replican las salidas del IaC que utilizó para aprovisionar la infraestructura gestionada por el cliente. Debe definir los tipos de valores de salida únicamente como cadenas (no como listas, matrices u otros tipos). Por ejemplo, el siguiente fragmento de código muestra la sección de salidas de una plantilla externa. AWS CloudFormation Esto es de la plantilla que se muestra en el ejemplo 1. Se puede utilizar para crear una infraestructura externa gestionada por el cliente para un servicio de AWS Proton Fargate creado a partir del ejemplo 4.

> **Important**
>
> Como administrador, debe asegurarse de que la infraestructura aprovisionada y gestionada y todos los parámetros de salida sean compatibles con las plantillas de entorno gestionado por el cliente asociadas. AWS Proton no puede contabilizar los cambios en su nombre porque estos cambios no son visibles para AWS Proton usted. Las inconsistencias producen errores.

Ejemplos de resultados de un archivo CloudFormation IaC para una plantilla de entorno gestionado por el cliente:

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

El esquema del paquete de plantillas de entorno gestionado por el AWS Proton cliente correspondiente se muestra en el siguiente ejemplo. Cada valor de salida se define como una cadena.

Ejemplo de esquema para una plantilla de entorno gestionado por el cliente:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "EnvironmentOutput"
  types:                          # required
    # defined by administrator
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:              # parameter
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:     # parameter
          type: string
          description: "The ARN of the ECS role"
        VpcId:                    # parameter
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Requisitos del esquema para los paquetes de plantillas de servicio

El esquema debe seguir la sección de modelos de datos (esquemas) de OpenAPI en formato YAML, como se muestra en los siguientes ejemplos. Debes incluir un archivo de esquema en tu paquete de plantillas de servicio.

En los siguientes ejemplos de esquemas de servicio, debe incluir los encabezados formateados. En el ejemplo siguiente, se encuentra en las tres primeras líneas. Esto sirve para establecer que está utilizando la sección de modelos de datos (esquemas) de la API abierta.

`service_input_type`Debe incluirse y definirse con el nombre que usted proporcione. En el siguiente ejemplo, se encuentra en la línea 5. Esto asocia los parámetros a un recurso AWS Proton de servicio.

De forma predeterminada, se incluye una canalización de AWS Proton servicios cuando se utiliza la consola o la CLI para crear un servicio. Cuando incluye una canalización de servicios para su

servicio, debe `pipeline_input_type` incluirla con el nombre que proporcione. En el siguiente ejemplo, se encuentra en la línea 7. No incluyas este parámetro si no vas a incluir una canalización AWS Proton de servicios. Para obtener más información, consulte Registro y publicación de plantillas.

Para seguir el modelo de esquema de API abierta, debes incluir: `types` en el siguiente ejemplo, esto está en la línea 9.

A continuación`types`, debe definir un `service_input_type` tipo. Defina los parámetros de entrada de su servicio como propiedades del`service_input_type`. Debe incluir al menos una propiedad con un nombre que coincida con al menos uno de los parámetros enumerados en la infraestructura del servicio como archivo de código (IaC) asociado al esquema.

Para definir una canalización de servicios, debajo de su `service_input_type` definición, debe definir una`pipeline_input_type`. Como se indica anteriormente, debe incluir al menos una propiedad con un nombre que coincida con al menos un parámetro incluido en un archivo IaC de canalización asociado al esquema. No incluyas esta definición si no vas a incluir una canalización AWS Proton de servicios.

Cuando usted, como administrador o desarrollador, crea un servicio y proporciona valores de parámetros personalizados, AWS Proton utiliza el archivo de esquema para compararlos, validarlos e insertarlos en los parámetros entrecruzados del archivo CloudFormation IaC asociado. Para cada propiedad (parámetro), proporcione a y a. `name` `type` Si lo desea, proporcione también un `description`, `default`, `pattern`.

Los parámetros definidos para el esquema de ejemplo incluyen `port`, `desired_count`, `task_size` y `image` con la `default` palabra clave y los valores predeterminados. Al crear un servicio con este esquema de paquete de plantillas de servicio, puede aceptar los valores predeterminados o proporcionar los suyos propios. El parámetro también `unique_name` se incluye en el ejemplo y no tiene un valor predeterminado. Se muestra como una `required` propiedad (parámetro). Usted, como administrador o desarrollador, debe proporcionar valores para `required` los parámetros al crear los servicios.

Si desea crear una plantilla de servicio con una canalización de servicios, `pipeline_input_type` inclúyala en su esquema.

Ejemplo de archivo de esquema de servicio para un servicio que incluye una canalización AWS Proton de servicios.

Este ejemplo de esquema se puede utilizar con los archivos AWS Proton IaC que se muestran en los ejemplos 4 y 5. Se incluye una canalización de servicios.

```
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
```

Si quieres crear una plantilla de servicio sin una canalización de servicios, no la incluyas `pipeline_input_type` en tu esquema, como se muestra en el siguiente ejemplo.

Ejemplo de archivo de esquema de servicio para un servicio que no incluye una canalización AWS Proton de servicios

```
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"
```

Resuma los archivos de plantilla para AWS Proton

Tras preparar el entorno y la infraestructura de servicios como archivos de código (IaC) y sus respectivos archivos de esquema, debe organizarlos en directorios. También debe crear un archivo YAML de manifiesto. El archivo de manifiesto muestra los archivos IaC de un directorio, el motor de renderizado y el lenguaje de plantillas utilizado para desarrollar el IaC en esta plantilla.

> **Note**
>
> Un archivo de manifiesto también se puede utilizar independientemente de los paquetes de plantillas, como entrada directa a los componentes definidos directamente. En este caso, siempre especifica un único archivo de plantilla de IaC, tanto para Terraform como para Terraform CloudFormation . Para obtener más información sobre los componentes, consulte Componentes.

El archivo de manifiesto debe seguir el formato y el contenido que se muestran en el siguiente ejemplo.

CloudFormation formato de archivo de manifiesto:

Con CloudFormation, se muestra un único archivo.

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

Formato de archivo de manifiesto de Terraform:

Con terraform, puede enumerar explícitamente un solo archivo o usar el comodín * para enumerar cada uno de los archivos de un directorio.

> **Note**
>
> El comodín solo incluye los archivos cuyos nombres terminan en. `.tf` Los demás archivos se omiten.

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

CodeBuildformato de archivo de manifiesto de aprovisionamiento basado en:

Con el aprovisionamiento CodeBuild basado, se especifican los comandos de shell de aprovisionamiento y desaprovisionamiento.

> **Note**
>
> Además del manifiesto, el paquete debe incluir todos los archivos de los que dependan los comandos.

El siguiente ejemplo de manifiesto utiliza el aprovisionamiento CodeBuild basado para aprovisionar (implementar) y desaprovisionar (destruir) recursos mediante AWS Cloud Development Kit (AWS CDK) (AWS CDK). El paquete de plantillas también debe incluir el código CDK.

Durante el aprovisionamiento, AWS Proton crea un archivo de entrada con valores para los parámetros de entrada que usted definió en el esquema de la plantilla con el nombre. `proton-input.json`

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
```

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

Tras configurar los directorios y los archivos de manifiesto para su entorno o paquete de plantillas de servicio, comprima los directorios en forma de bola de alquitrán y los cargue en un depósito de Amazon Simple Storage Service (Amazon S3), AWS Proton donde podrá recuperarlos, o en un repositorio Git de sincronización de plantillas.

Cuando crea una versión secundaria de un entorno o plantilla de servicio en el que se ha registrado AWS Proton, proporciona la ruta a su entorno o paquete de plantillas de servicio (tar ball) que se encuentra en su depósito de S3. AWS Proton lo guarda con la nueva versión secundaria de la plantilla. Puede seleccionar la nueva versión secundaria de la plantilla para crear o actualizar entornos o servicios con ella AWS Proton.

Resumen del paquete de plantillas de entorno

Hay dos tipos de paquetes de plantillas de entorno para AWS Proton los que puede crear.

- Para crear un paquete de plantillas de entorno para una plantilla de entorno estándar, organice el esquema, los archivos de infraestructura como código (IaC) y el archivo de manifiesto en directorios, tal y como se muestra en la siguiente estructura de directorios de paquetes de plantillas de entorno.
- Para crear un paquete de plantillas de entorno para una plantilla de entorno gestionada por el cliente, proporcione únicamente el archivo y el directorio del esquema. No incluya el directorio ni los archivos de infraestructura. AWS Proton arroja un error si se incluyen el directorio y los archivos de la infraestructura.

Para obtener más información, consulte Registro y publicación de plantillas.

CloudFormation estructura de directorios del paquete de plantillas de entorno:

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

Estructura del directorio del paquete de plantillas de entorno Terraform:

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

Resumen del paquete de plantillas de servicio

Para crear un paquete de plantillas de servicio, debe organizar el esquema, los archivos de infraestructura como código (IaC) y los archivos de manifiesto en directorios, como se muestra en el ejemplo de estructura de directorios del paquete de plantillas de servicio.

Si no incluye una canalización de servicios en el paquete de plantillas, no incluya el directorio ni los archivos ni el conjunto de la canalización `"pipelineProvisioning": "CUSTOMER_MANAGED"` al crear la plantilla de servicio que se va a asociar a este paquete de plantillas.

> **Note**
>
> No puede modificar `pipelineProvisioning` una vez se haya creado la plantilla de servicio.

Para obtener más información, consulte Registro y publicación de plantillas.

CloudFormation Estructura de directorios del paquete de plantillas de servicio:

```
/schema
  schema.yaml
```

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

Estructura de directorios del paquete de plantillas de servicio de Terraform:

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

Consideraciones sobre el paquete de plantillas

- Archivos de infraestructura como código (IaC)

  AWS Proton audita las plantillas para determinar el formato de archivo correcto. Sin embargo, AWS Proton no comprueba si hay errores lógicos, de dependencia o de desarrollo de la plantilla. Por ejemplo, supongamos que especificó la creación de un bucket de Amazon S3 en su archivo AWS CloudFormation IaC como parte de su plantilla de servicio o entorno. Se crea un servicio en función de esas plantillas. Ahora, supongamos que en algún momento desea eliminar el servicio. Si el depósito de S3 especificado no está vacío y el archivo CloudFormation iAC no lo marca como tal`DeletionPolicy`, se produce un AWS Proton error `Retain` en la operación de eliminación del servicio.
- Límites de tamaño y formato de los archivos del paquete
  - Los límites de tamaño, número y tamaño del nombre de los archivos del paquete se encuentran enCuotas de AWS Proton.
  - Los directorios de archivos del paquete de plantillas se comprimen con gzip en forma de bola de alquitrán y se ubican en un depósito de Amazon Simple Storage Service (Amazon S3).
  - Cada archivo del paquete debe ser un archivo YAML con un formato válido.
- Cifrado de paquetes de plantillas de bucket S3

Si desea cifrar los datos confidenciales de los paquetes de plantillas que están en reposo en su depósito de S3, utilice las claves SSE-S3 o SSE-KMS para poder recuperarlos. AWS Proton

AWS ProtonPlantillas de

Para añadir el paquete de plantillas a la biblioteca de plantillas de AWS Proton, cree una versión secundaria de la plantilla y regístrela en AWS Proton. Al crear la plantilla, proporcione el nombre del bucket de Amazon S3 y la ruta del paquete de plantillas. Una vez publicadas las plantillas, los miembros del equipo de la plataforma y los desarrolladores podrán seleccionarlas. Una vez seleccionada una plantilla, AWS Proton la utiliza para crear y aprovisionar la infraestructura y las aplicaciones.

Como administrador, puede crear y registrar una plantilla de entorno con AWS Proton. A continuación, esta plantilla de entorno se puede utilizar para implementar varios entornos. Por ejemplo, se puede utilizar para implementar entornos de “desarrollo”, de “ensayo” y de “producción”. El entorno de “desarrollo” puede incluir una VPC con subredes privadas y una política de acceso restrictiva a todos los recursos. Las salidas del entorno se pueden utilizar como entradas para los servicios.

Puede crear y registrar plantillas de entorno para crear dos tipos diferentes de entornos. Tanto el usuario como los desarrolladores pueden utilizar AWS Proton para implementar servicios en ambos tipos de entornos.

- Registre y publique una plantilla de entorno estándar que AWS Proton utilice para crear un entorno estándar que aprovisione y administre la infraestructura del entorno.
- Registre y publique una plantilla de entorno administrado por el cliente que AWS Proton utilice para crear un entorno administrado por el cliente que se conecte a la infraestructura aprovisionada existente. AWS Proton no administra la infraestructura aprovisionada existente.

Puede crear y registrar plantillas de servicio con AWS Proton para implementar servicios en los entornos. Se debe crear un entorno de AWS Proton antes de poder implementar un servicio en él.

En la siguiente lista se describe cómo se crean y administran las plantillas con AWS Proton.

- (Opcional) Prepare un rol de IAM para controlar el acceso de los desarrolladores a las llamadas a la API de AWS Proton y a los roles de servicio de IAM de AWS Proton. Para obtener más información, consulte the section called “Roles de IAM”.
- Cree un paquete de plantillas. Para obtener más información, consulte Paquetes de plantillas.
- Cree y registre una plantilla con AWS Proton después de crear, comprimir y guardar el paquete de plantillas en un bucket de Amazon S3. Puede hacerlo mediante la consola o la AWS CLI.

- Pruebe y utilice la plantilla para crear y administrar los recursos aprovisionados de AWS Proton una vez que se haya registrado en AWS Proton.
- Cree y administre versiones principales y secundarias de la plantilla a lo largo de su vida útil.

Puede administrar las versiones de la plantilla manualmente o con las configuraciones de sincronización de plantillas:

- Utilice la consola de AWS Proton o la AWS CLI para crear una nueva versión secundaria o principal.
- Cree una configuración de sincronización de plantillas que permita a AWS Proton crear automáticamente una nueva versión secundaria o principal cuando detecte un cambio en el paquete de plantillas en un repositorio que se haya definido.

Para obtener más información, consulte la Referencia de la API del servicio de AWS Proton.

Temas

Plantillas versionadas

Como administrador o miembro de un equipo de plataforma, el usuario define, crea y administra una biblioteca de plantillas versionadas que se utilizan para aprovisionar recursos de infraestructura. Hay dos tipos de versiones de plantillas: versiones secundarias y versiones principales.

- Versiones secundarias: cambios en la plantilla que tienen un esquema compatible con versiones anteriores. Estos cambios no requieren que el desarrollador proporcione nueva información al actualizar a la nueva versión de la plantilla.

Cuando intente realizar algún cambio en una versión secundaria, AWS Proton hará todo lo posible por determinar si el esquema de la nueva versión es compatible con versiones secundarias anteriores de la plantilla. Si el nuevo esquema no es compatible con versiones anteriores, AWS Proton no podrá registrar la nueva versión secundaria.

> **Note**
>
> La compatibilidad se determina únicamente en función del esquema. AWS Proton no comprueba si el archivo de infraestructura como código (IaC) del paquete de plantillas es compatible con versiones secundarias anteriores. Por ejemplo, AWS Proton no comprueba si el nuevo archivo de IaC provoca cambios importantes en las aplicaciones que se ejecutan en la infraestructura aprovisionada por una versión secundaria anterior de la plantilla.

- Versiones principales: cambios en la plantilla que pueden no ser compatibles con versiones anteriores. Estos cambios suelen requerir nuevas entradas por parte del desarrollador y suelen implicar cambios en el esquema de la plantilla.

  En ocasiones, el usuario puede decidir si desea designar un cambio compatible con versiones anteriores como una versión principal en función del modelo operativo de su equipo.

La forma en que AWS Proton determina si una solicitud de versión de plantilla es para una versión secundaria o principal dependerá de la forma en que se realice el seguimiento de los cambios en la plantilla:

- Cuando se solicita explícitamente la creación de una nueva versión de plantilla, se solicita una versión principal al especificar un número de versión principal, y se solicita una versión secundaria sin especificar ningún número de versión principal.
- Cuando se utiliza la sincronización de plantillas (y, por lo tanto, no se solicitan versiones de plantillas explícitas), AWS Proton intenta crear nuevas versiones secundarias para los cambios de la plantilla que se produzcan en el archivo YAML existente. AWS Proton crea una versión principal al crear un directorio nuevo para el nuevo cambio de la plantilla (por ejemplo, al pasar de la v1 a la v2).

> **Note**
>
> El registro de una nueva versión secundaria basado en la sincronización de la plantilla seguirá produciendo un error si AWS Proton determina que el cambio no es compatible con versiones anteriores.

Cuando se publica una nueva versión de una plantilla, esta se convierte en la versión recomendada si se trata de la versión principal y secundaria superior. Los nuevos recursos de AWS Proton se crean con la nueva versión recomendada, y AWS Proton pide a los administradores que utilicen la nueva versión y que actualicen los recursos de AWS Proton existentes que utilicen una versión desactualizada.

Registro y publicación de plantillas

Puede registrar y publicar plantillas de entorno y servicio con AWS Proton, tal y como se describe en las siguientes secciones.

Puede crear una nueva versión de una plantilla con la consola o con la AWS CLI.

Como alternativa, puede utilizar la consola o la AWS CLI para crear una plantilla y configurar una sincronización de plantillas para dicha plantilla. Esta configuración permite a AWS Proton sincronizar desde paquetes de plantillas ubicados en los repositorios Git registrados que se hayan definido. Cada vez que se inserta una confirmación al repositorio que cambia alguno de los paquetes de plantillas, se crea una nueva versión secundaria o principal de la plantilla, siempre y cuando la versión aún no exista. Para obtener más información sobre los requisitos previos y los requisitos de la configuración de sincronización de plantillas, consulte Configuraciones de sincronización de plantillas.

Registro y publicación de plantillas de entorno

Puede registrar y publicar los siguientes tipos de plantillas de entorno.

- Registre y publique una plantilla de entorno estándar que AWS Proton utilice para implementar y administrar la infraestructura del entorno.
- Registre y publique una plantilla de entorno administrado por el cliente que AWS Proton utilice para conectarse a la infraestructura aprovisionada existente que usted administre. AWS Proton no administra la infraestructura aprovisionada existente.

> **Important**
>
> Como administrador, asegúrese de que tanto la infraestructura aprovisionada y administrada como todos los parámetros de salida sean compatibles con las plantillas de entorno asociadas y administradas por el cliente. AWS Proton no puede contabilizar los cambios en su nombre porque estos cambios no son visibles para AWS Proton. Las inconsistencias producen errores.

Puede utilizar la consola o la AWS CLI para registrar y publicar una plantilla de entorno.

AWS Management Console

Utilice la consola para registrar y publicar una nueva plantilla de entorno.

1. En la consola de AWS Proton, elija Plantillas de entorno.
2. Seleccione Crear plantilla de entorno.
3. En la página Crear plantilla de entorno, en la sección Opciones de plantilla, elija una de las dos opciones de plantilla disponibles.
   - Crear una plantilla para aprovisionar nuevos entornos.
   - Crear una plantilla para utilizar la infraestructura aprovisionada que administre el usuario.
4. Si ha seleccionado Crear una plantilla para aprovisionar nuevos entornos, en la sección Origen del paquete de plantillas, elija una de las tres opciones de código fuente del paquete de plantillas disponibles. Para obtener más información sobre los requisitos y requisitos previos para la sincronización de plantillas, consulte Configuraciones de sincronización de plantillas.
   - Utilice uno de nuestros paquetes de plantillas de muestra.
   - Usar su propio paquete de plantillas.
   - Sincronizar plantillas de Git.
5. Proporcione una ruta a un paquete de plantillas.
   a. Si elige Utilice uno de nuestros paquetes de plantillas de muestra:

      En la sección Paquete de plantillas de muestra, seleccione un paquete de plantillas de muestra.

b. Si ha seleccionado Sincronizar plantillas de Git, en la sección Código fuente:

   i. Seleccione el repositorio para configurar la sincronización de plantillas.

   ii. Introduzca el nombre de la ramificación del repositorio desde la que desee llevar a cabo la sincronización.

   iii. (Opcional) Introduzca el nombre de un directorio para limitar la búsqueda del paquete de plantillas.

c. De lo contrario, en la sección de Ubicación del paquete de S3, se proporcionará una ruta al paquete de plantillas.

6. En la sección Detalles de la plantilla.

   a. Introduzca un Nombre de plantilla.

   b. (Opcional) Introduzca un Nombre de visualización de la plantilla.

   c. (Opcional) Introduzca una Descripción de la plantilla para la plantilla de entorno.

7. (Opcional) Marque la casilla Personalización de la configuración de cifrado (avanzada) en la sección Configuración de cifrado para proporcionar su propia clave de cifrado.

8. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.

9. Seleccione Crear plantilla de entorno.

   Ahora aparecerá una nueva página que muestra el estado y los detalles de su nueva plantilla de entorno. Estos detalles incluyen una lista de etiquetas administradas por el cliente y por AWS. AWS Proton genera automáticamente etiquetas administradas por AWS cuando se crean recursos de AWS Proton. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

10. El estado de una nueva plantilla de entorno comienza en el estado Borrador. Usted y las personas que gocen de permisos `proton:CreateEnvironment` podrán verla y acceder a ella. Siga el siguiente paso para poner la plantilla a disposición de otras personas.

11. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria de la plantilla que acaba de crear (1.0). Como alternativa, puede seleccionar Publicar en la alerta de información y omitir el siguiente paso.

12. En la sección Versiones de la plantilla, seleccione Publicar.

13. El estado de la plantilla cambiará a Publicada. Como esta es la versión más reciente de la plantilla, pasará a ser la versión Recomendada.

14. En el panel de navegación, seleccione Plantillas de entorno para ver una lista de las plantillas de entorno y sus detalles.

Utilice la consola para registrar las nuevas versiones principales y secundarias de una plantilla de entorno.

Para obtener más información, consulte Plantillas versionadas.

1. En la consola de AWS Proton, elija Plantillas de entorno.
2. En la lista de plantillas de entorno, elija el nombre de la plantilla de entorno para la que desee crear una versión principal o secundaria.
3. En la vista detallada de la plantilla de entorno, elija Crear nueva versión en la sección Versiones de la plantilla.
4. En la página Crear una nueva versión de plantilla de entorno, en la sección Origen del paquete de plantillas, elija una de las dos opciones de código fuente del paquete de plantillas disponibles.
   - Utilice uno de nuestros paquetes de plantillas de muestra.
   - Usar su propio paquete de plantillas.
5. Proporcione una ruta al paquete de plantillas seleccionado.
   - Si eligió Utilice uno de nuestros paquetes de plantillas de muestra, en la sección Paquete de plantillas de muestra, seleccione un paquete de plantillas de muestra.
   - Si eligió Usar su propio paquete de plantillas, en la sección Ubicación del paquete de S3, elija la ruta al paquete de plantillas.
6. En la sección Detalles de la plantilla.
   a. (Opcional) Introduzca un Nombre de visualización de la plantilla.
   b. (Opcional) Introduzca una Descripción de la plantilla para la plantilla de servicio.
7. En la sección Detalles de la plantilla, elija una de las siguientes opciones.
   - Para crear una versión secundaria, desmarque la casilla Marcar para crear una nueva versión principal.
   - Para crear una versión principal, marque la casilla Marcar para crear una nueva versión principal.

8. Continúe con los pasos de la consola para crear la nueva versión secundaria o principal y seleccione Crear nueva versión.

AWS CLI

Utilice la CLI para registrar y publicar una nueva plantilla de entorno, como se muestra en los pasos siguientes.

1. Cree una plantilla de entorno estándar o administrado por el cliente; para ello, debe especificar la región, el nombre, el nombre para mostrar (opcional) y una descripción (opcional).

   a. Cree una plantilla de entorno estándar.

      Ejecute el siguiente comando:

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

      Respuesta:

      ```
      {
          "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. Cree una plantilla de entorno administrado por el cliente; para ello, añada el parámetro `provisioning` con el valor `CUSTOMER_MANAGED`.

      Ejecute el siguiente comando:

      ```
      $ aws proton create-environment-template \
      ```

```
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access" \
    --provisioning "CUSTOMER_MANAGED"
```

Respuesta:

```
{
    "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. Creación de una versión secundaria 0 de la versión principal 1 de la plantilla de entorno

   Este paso y el resto son los mismos para las plantillas de entorno estándar y administrado por el cliente.

   Incluya el nombre de la plantilla, la versión principal y el nombre y la clave del bucket de S3 que contenga el paquete de plantillas de entorno.

   Ejecute el siguiente comando:

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

   Respuesta:

```
{
    "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. Utilice el comando “get” para comprobar el estado de los registros.

   Ejecute el siguiente comando:

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

   Respuesta:

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

```
}
```

4. Publique la versión secundaria 0 de la versión principal 1 de la plantilla de entorno; para ello, proporcione el nombre de la plantilla y las versiones principal y secundaria. Esta versión es la versión `Recommended`.

   Ejecute el siguiente comando:

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

   Respuesta:

```
{
    "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"
    }
}
```

Tras crear una nueva plantilla con la AWS CLI, podrá ver una lista de AWS y de etiquetas administradas por el cliente. AWS Proton genera automáticamente etiquetas administradas por AWS para el usuario. También puede modificar y crear etiquetas administradas por el cliente mediante la AWS CLI. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

Ejecute el siguiente comando:

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

Registro y publicación de plantillas de servicio

Al crear una versión de una plantilla de servicio, debe especificar una lista de plantillas de entorno compatibles. De esta forma, cuando los desarrolladores seleccionen una plantilla de servicio, tendrán opciones para el entorno en el que implementar su servicio.

Antes de crear un servicio a partir de una plantilla de servicio o antes de publicar una plantilla de servicio, confirme que los entornos se implementen a partir de las plantillas de entorno compatibles de la lista.

No puede actualizar un servicio a la nueva versión principal si se ha implementado en un entorno creado a partir de una plantilla de entorno compatible eliminada.

Para añadir o eliminar plantillas de entorno compatibles para una versión de plantilla de servicio, debe crear una nueva versión principal de la misma.

Puede utilizar la consola o la AWS CLI para registrar y publicar una plantilla de servicio.

AWS Management Console

Utilice la consola para registrar y publicar una nueva plantilla de servicio.

1. En la consola de AWS Proton, elija Plantillas de servicio.
2. Elija Crear plantilla de servicio.
3. En la página Crear plantilla de servicio, en la sección Origen del paquete de plantillas, elija una de las opciones de plantilla disponibles.

   - Usar su propio paquete de plantillas.

- Sincronización de plantillas desde Git.

4. Proporcione una ruta a un paquete de plantillas.

   a. Si ha seleccionado Sincronizar plantillas de Git, en la sección Repositorio de código fuente:

      i. Seleccione el repositorio para configurar la sincronización de plantillas.

      ii. Introduzca el nombre de la ramificación del repositorio desde la que desee llevar a cabo la sincronización.

      iii. (Opcional) Introduzca el nombre de un directorio para limitar la búsqueda del paquete de plantillas.

   b. De lo contrario, en la sección de Ubicación del paquete de S3, se proporcionará una ruta al paquete de plantillas.

5. En la sección Detalles de la plantilla.

   a. Introduzca un Nombre de plantilla.

   b. (Opcional) Introduzca un Nombre de visualización de la plantilla.

   c. (Opcional) Introduzca una Descripción de la plantilla para la plantilla de servicio.

6. En la sección Plantillas de entorno compatibles, elija una plantilla de la lista de plantillas de entorno compatibles.

7. (Opcional) En la sección Configuración de cifrado, elija Personalización de la configuración de cifrado (avanzada) para proporcionar su propia clave de cifrado.

8. (Opcional) En la sección Canalización:

   Si no incluye una definición de canalización de servicios en la plantilla de servicio, desmarque la casilla de Canalización: opcional, situada en la parte inferior de la página. No podrá cambiar esta opción una vez creada la plantilla de servicio. Para obtener más información, consulte Paquetes de plantillas.

9. (Opcional) En la sección Orígenes de componentes compatibles, en Orígenes de componentes, seleccione Definición directa para asociar los componentes definidos directamente a las instancias de servicio.

10. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.

11. Elija Crear una plantilla de servicio.

Ahora aparecerá una nueva página que muestra el estado y los detalles de su nueva plantilla de servicio. Estos detalles incluyen una lista de etiquetas administradas por el cliente y por AWS. AWS Proton genera automáticamente etiquetas administradas por AWS cuando se crean recursos de AWS Proton. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

12. El estado inicial de una nueva plantilla de servicio será el estado Borrador. Usted y las personas que gocen de permisos `proton:CreateService` podrán verla y acceder a ella. Siga el siguiente paso para poner la plantilla a disposición de otras personas.
13. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria de la plantilla que acaba de crear (1.0). Como alternativa, puede seleccionar Publicar en la alerta de información y omitir el siguiente paso.
14. En la sección Versiones de la plantilla, seleccione Publicar.
15. El estado de la plantilla cambiará a Publicada. Como esta es la versión más reciente de la plantilla, pasará a ser la versión Recomendada.
16. En el panel de navegación, seleccione Plantillas de servicio para ver una lista de sus plantillas y detalles de servicio.

Utilice la consola para registrar las nuevas versiones principales y secundarias de una plantilla de servicio.

Para obtener más información, consulte Plantillas versionadas.

1. En la consola de AWS Proton, elija Plantillas de servicio.
2. En la lista de plantillas de servicio, elija el nombre de la plantilla de servicio para la que desee crear una versión principal o secundaria.
3. En la vista detallada de la plantilla de servicio, elija Crear nueva versión en la sección Versiones de la plantilla.
4. En la página Crear una nueva versión de plantillas de servicio, en la sección Origen del paquete, seleccione Usar su propio paquete de plantillas.
5. En la sección Ubicación del paquete de S3, elija la ruta al paquete de plantillas.
6. En la sección Detalles de la plantilla.
    a. (Opcional) Introduzca un Nombre de visualización de la plantilla.
    b. (Opcional) Introduzca una Descripción de la plantilla para la plantilla de servicio.

7. En la sección Detalles de la plantilla, elija una de las siguientes opciones.
   - Para crear una versión secundaria, desmarque la casilla Marcar para crear una nueva versión principal.
   - Para crear una versión principal, marque la casilla Marcar para crear una nueva versión principal.
8. Continúe con los pasos de la consola para crear la nueva versión secundaria o principal y seleccione Crear nueva versión.

AWS CLI

Para crear una plantilla de servicio que implemente un servicio sin una canalización de servicios, agregue el parámetro y el valor `--pipeline-provisioning "CUSTOMER_MANAGED"` al comando `create-service-template`. Configure los paquetes de plantillas como se describe en creación de Paquetes de plantillas y Requisitos del esquema para los paquetes de plantillas de servicio.

> **Note**
>
> No puede modificar `pipelineProvisioning` una vez se haya creado la plantilla de servicio.

1. Utilice la CLI para registrar y publicar una nueva plantilla de servicio, con o sin una canalización de servicios, como se muestra en los pasos siguientes.
   a. Cree una plantilla de servicio con una canalización de servicios mediante la CLI.

      Introduzca el nombre, el nombre para mostrar (opcional) y una descripción (opcional).

      Ejecute el siguiente comando:

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

      Respuesta:

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

b. Cree una plantilla de servicio sin una canalización de servicios.

Add `--pipeline-provisioning`.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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. Cree una versión secundaria 0 de la versión principal 1 de la plantilla de servicio.

   Incluya el nombre de la plantilla, las plantillas de entorno compatibles, la versión principal y el nombre y la clave del bucket de S3 que contenga el paquete de plantillas de servicio.

   Ejecute el siguiente comando:

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

   Respuesta:

```
{
    "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. Utilice el comando get para comprobar el estado de los registros.

   Ejecute el siguiente comando:

```
$ aws proton get-service-template-version \
```

```
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Respuesta:


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

4. Publique la plantilla de servicio mediante el comando “update” para cambiar el estado a `"PUBLISHED"`.

   Ejecute el siguiente comando:

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

   Respuesta:

```
{
    "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. Compruebe que AWS Proton haya publicado la versión 1.0 mediante el comando “get” para recuperar los datos detallados de la plantilla de servicio.

   Ejecute el siguiente comando:

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

   Respuesta:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:03:04.767000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
```

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

Visualización de datos de la plantilla

Puede ver listas de plantillas con detalles y visualizar plantillas individuales con datos detallados mediante la consola de AWS Proton y la AWS CLI.

Los datos de la plantilla de entorno administrado por el cliente incluyen el parámetro `provisioned` con el valor `CUSTOMER_MANAGED`.

Si una plantilla de servicio no incluye ninguna canalización de servicios, los datos de la plantilla de servicio incluirán el parámetro `pipelineProvisioning` con el valor `CUSTOMER_MANAGED`.

Para obtener más información, consulte Registro y publicación de plantillas.

Puede utilizar la consola o la AWS CLI para enumerar y ver los datos de la plantilla.

AWS Management Console

Utilice la consola para enumerar y ver las plantillas.

1. Para ver una lista de plantillas, elija Plantillas (de entorno o de servicio).
2. Para ver los datos detallados, elija el nombre de una plantilla.

   Vea los datos detallados de la plantilla, una lista de las versiones principales y secundarias de la plantilla y una lista de los recursos de AWS Proton que se implementaron mediante las versiones de la plantilla y las etiquetas de la plantilla.

La versión principal y la versión secundaria recomendadas están etiquetadas como Recomendadas.

AWS CLI

Utilice la AWS CLI para enumerar y ver las plantillas.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Ejecute el siguiente comando:

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

Respuesta:

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

Vea una versión secundaria de una plantilla de servicio.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "serviceTemplateMinorVersion": {
```

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


Vea una plantilla de servicio sin una canalización de servicios, como se muestra en el siguiente ejemplo de comando y respuesta.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Actualizar una plantilla

Puede actualizar una plantilla tal y como se describe en la siguiente lista.

- Edite los valores `description` o `display name` de una plantilla cuando utilice la consola o la AWS CLI. No puede editar el valor `name` de una plantilla.
- Actualice el estado de la versión secundaria de una plantilla cuando utilice la consola o la AWS CLI. Solo se puede cambiar el estado de `DRAFT` a `PUBLISHED`.
- Edite el nombre para mostrar y la descripción de una versión secundaria o principal de una plantilla cuando utilice la AWS CLI.

AWS Management Console

Edite la descripción y el nombre para mostrar de una plantilla mediante la consola tal y como se describe en los pasos siguientes.

En la lista de plantillas.

1. En la consola de AWS Proton, elija Plantillas (de entorno o de servicio).
2. En la lista de plantillas, elija el botón de radio situado a la izquierda de la plantilla cuya descripción o nombre para mostrar desee actualizar.
3. Elija Acciones y, a continuación, elija Editar.
4. En la página Editar plantilla (de entorno o de servicio), en la sección Detalles de la plantilla, introduzca las modificaciones en el formulario y seleccione Guardar cambios.

Cambie el estado de una versión secundaria de una plantilla mediante la consola para publicar una plantilla, tal y como se describe a continuación. Solo se puede cambiar el estado de `DRAFT` a `PUBLISHED`.

En la página de detalles de la plantilla (de entorno o de servicio).

1. En la consola de AWS Proton, elija Plantillas (de entorno o de servicio).
2. En la lista de plantillas, elija el nombre de la plantilla desde la que desee actualizar el estado de una versión secundaria de Borrador a Publicada.
3. En la página de detalles de la plantilla (entorno o servicio), en la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria que desee publicar.
4. Seleccione Publicar en la sección Versiones de la plantilla. El estado cambiará de Borrador a Publicada.

AWS CLI

El siguiente ejemplo de comando y respuesta muestra cómo se puede editar la descripción de una plantilla de entorno.

Ejecute el siguiente comando.

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

Respuesta:

```
{
    "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"
    }
}
```

También puede utilizar la AWS CLI para actualizar las plantillas de servicio. Consulte el paso 5 en Registro y publicación de plantillas de servicio para ver un ejemplo de cómo actualizar el estado de una versión secundaria de una plantilla de servicio.

Eliminación de plantillas

Las plantillas se pueden eliminar mediante la consola y la AWS CLI.

Puede eliminar una versión secundaria de una plantilla de entorno si no hay ningún otro entorno implementado en esa versión.

Puede eliminar una versión secundaria de una plantilla de servicio si no hay instancias de servicio o canalizaciones implementadas en esa versión. La canalización se puede implementar en una versión de plantilla diferente a la de la instancia de servicio. Por ejemplo, si la instancia de servicio se ha actualizado a la versión 1.1 desde la versión 1.0 y la canalización sigue implementada en la versión 1.0, no será posible eliminar la plantilla de servicio de versión 1.0.

AWS Management Console

Puede utilizar la consola para eliminar la plantilla completa o las versiones secundarias y principales individuales de una plantilla.

Utilice la consola para eliminar plantillas como se muestra a continuación.

> **Note**
>
> Al utilizar la consola para eliminar plantillas.
>
> - Al eliminar la plantilla completa, también se eliminarán las versiones principal y secundaria de dicha plantilla.

En la lista de plantillas (de entorno o de servicio).

1. En la consola de AWS Proton, elija Plantillas (de entorno o de servicio).
2. En la lista de plantillas, seleccione el botón de radio situado a la izquierda de la plantilla que desee eliminar.

   Solo puede eliminar una plantilla completa si no hay recursos de AWS Proton implementados en sus versiones.
3. Seleccione Acciones y, a continuación, Eliminar para eliminar toda la plantilla.
4. Un modal le pedirá que confirme la acción de eliminación.
5. Siga las instrucciones y seleccione Sí, eliminar.

En la página de detalles de la plantilla (de entorno o de servicio).

1. En la consola de AWS Proton, elija Plantillas (de entorno o de servicio).
2. En la lista de plantillas, elija el nombre de la plantilla que desee eliminar por completo o elimine las versiones principales o secundarias individuales de la misma.
3. Para eliminar la plantilla completa.

   Solo puede eliminar una plantilla completa si no hay recursos de AWS Proton implementados en sus versiones.

   a. Seleccione Eliminar, en la esquina superior derecha de la página.
   b. Un modal le pedirá que confirme la acción de eliminación.
   c. Siga las instrucciones y seleccione Sí, eliminar.

4. Para eliminar las versiones principales o secundarias de una plantilla.

   Solo puede eliminar una versión secundaria de una plantilla si no hay recursos de AWS Proton implementados en esa versión.

   a. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión que desee eliminar.
   b. Seleccione Eliminar en la sección Versiones de la plantilla.
   c. Un modal le pedirá que confirme la acción de eliminación.
   d. Siga las instrucciones y seleccione Sí, eliminar.

AWS CLI

Las operaciones de eliminación de plantillas de la AWS CLI no incluyen la eliminación de otras versiones de una plantilla. Cuando utilice la AWS CLI, elimine las plantillas con las siguientes condiciones.

- Elimine una plantilla completa si no existen versiones secundarias o principales de la plantilla.
- Elimine una versión principal al eliminar la última versión secundaria restante.
- Elimine una versión secundaria de una plantilla si no hay recursos de AWS Proton implementados en esa versión.
- Elimine la versión secundaria recomendada de una plantilla si no existen otras versiones secundarias de la plantilla y si no hay recursos de AWS Proton implementados en esa versión.

Los siguientes comandos y respuestas de ejemplo muestran cómo utilizar la AWS CLI para eliminar plantillas.

Ejecute el siguiente comando:

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

Respuesta:

```
{
```

```
    "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"
    }
}
```

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Configuraciones de sincronización de plantillas

Aprende a configurar una plantilla para permitir la AWS Proton sincronización de los paquetes de plantillas ubicados en los repositorios de git registrados que tú definas. Cuando se inserta una confirmación al repositorio, AWS Proton comprueba si hay cambios en los paquetes de plantillas de dicho repositorio. Si detecta un cambio en un paquete de plantillas, se crea una nueva versión secundaria o principal de su plantilla, si la versión aún no existe. AWS Proton actualmente es compatible con GitHub GitHub Enterprise y BitBucket.

Inserción de una confirmación a un paquete de plantillas sincronizadas

Cuando se inserta una confirmación a una ramificación a la que alguna de las plantillas está realizando un seguimiento, AWS Proton clona el repositorio y determina qué plantillas necesita sincronizar. Escanea los archivos del directorio para encontrar directorios que coincidan con la convención de `{template-name}/{major-version}/`.

Después de AWS Proton determinar qué plantillas y versiones principales están asociadas a tu repositorio y rama, comienza a intentar sincronizar todas esas plantillas en paralelo.

Durante cada sincronización con una plantilla concreta, AWS Proton primero comprueba si el contenido del directorio de plantillas ha cambiado desde la última sincronización correcta. Si el contenido no ha cambiado, AWS Proton omite el registro de un paquete duplicado. Esto garantiza la creación de una nueva versión secundaria de la plantilla si el contenido del paquete de plantillas

cambia. Si el contenido del paquete de plantillas ha cambiado, el paquete se registra en él. AWS Proton

Una vez registrado el paquete de plantillas, AWS Proton supervisa el estado del registro hasta que se complete el registro.

Solo se puede realizar una sincronización con una versión secundaria y una versión principal de una plantilla determinada a la vez. Todas las confirmaciones que se hayan insertado mientras se estaba realizando una sincronización se agruparán por lotes. Las confirmaciones por lotes se sincronizan una vez finalizado el intento de sincronización anterior.

Sincronización de plantillas de servicio

AWS Proton puede sincronizar tanto las plantillas de entorno como las de servicio desde tu repositorio de git. Para sincronizar las plantillas de servicio, añada un archivo adicional con el nombre `.template-registration.yaml` a cada directorio de versiones principales del paquete de plantillas. Este archivo contiene los detalles adicionales AWS Proton necesarios para crear una versión de plantilla de servicio para ti tras una confirmación: entornos compatibles y fuentes de componentes compatibles.

La ruta completa del archivo es *`service-template-name`*`/`*`major-version`*`/.template-registration.yaml`. Para obtener más información, consulte the section called “Sincronización de plantillas de servicio”.

Consideraciones sobre la configuración de sincronización de plantillas

Revise las siguientes consideraciones para utilizar las configuraciones de sincronización de plantillas.

- Los repositorios no deben tener más de 250 MB.
- Para configurar la sincronización de plantillas, primero vincule el repositorio a AWS Proton. Para obtener más información, consulte the section called “Creación de un enlace a un repositorio”.
- Cuando se crea una nueva versión de la plantilla a partir de una plantilla sincronizada, tendrá el estado de `DRAFT`.
- Se creará una nueva versión secundaria de una plantilla si se cumple alguno de los siguientes requisitos:
  - El contenido del paquete de plantillas es diferente al de la última versión secundaria de la plantilla sincronizada.

  - Se eliminó la última versión secundaria de la plantilla previamente sincronizada.
- La sincronización no se puede pausar.
- Tanto las nuevas versiones secundarias como las principales se sincronizan automáticamente.
- Las configuraciones de sincronización de plantillas no pueden crear nuevas plantillas de nivel superior.
- No se puede sincronizar una plantilla desde varios repositorios con una configuración de sincronización de plantillas.
- No se pueden utilizar etiquetas en lugar de ramificaciones.
- Al crear una plantilla de servicio, se especifican plantillas de entorno compatibles.
- Puede crear una plantilla de entorno y agregarla como un entorno compatible para la plantilla de servicio en la misma confirmación.
- Las sincronizaciones con una sola versión principal de la plantilla se ejecutan de una en una. Durante una sincronización, si se detectan confirmaciones nuevas, se agruparán por lotes y se aplicarán al final de la sincronización activa. Las sincronizaciones con las diferentes versiones principales de la plantilla se realizan en paralelo.
- Si cambia la ramificación desde la que se sincronizan las plantillas, se completará primero cualquier sincronización en curso desde la ramificación anterior. A continuación, la sincronización comenzará a partir de la nueva ramificación.
- Si cambia el repositorio desde el que se sincronizan las plantillas, es posible que se produzca un error en las sincronizaciones en curso desde el repositorio anterior o que se ejecuten hasta completarse. Esto dependerá de la etapa de la sincronización en la que se encuentren.

Para obtener más información, consulta la referencia de la API AWS Proton de The Service.

Temas

Creación de una configuración de sincronización de plantillas

Aprenda a crear una configuración de sincronización de plantillas con AWS Proton.

Requisitos previos para crear una configuración de sincronización de plantillas:

- Haber vinculado un repositorio con AWS Proton.
- Que un paquete de plantillas se encuentre en el repositorio del usuario.

El enlace al repositorio consta de lo siguiente:

- Una CodeConnections conexión que da AWS Proton permiso para acceder a tu repositorio y suscribirte a sus notificaciones.
- Un rol vinculado a un servicio. Al vincular el repositorio, el rol vinculado al servicio se creará para el usuario.

Antes de crear su primera configuración de sincronización de plantillas, envíe un paquete de plantillas al repositorio, tal y como se muestra en el siguiente diseño de directorios.

```
/templates/                                          # subdirectory (optional)
/templates/my-env-template/                          # template name
/templates/my-env-template/v1/                       # template version
/templates/my-env-template/v1/infrastructure/        # template bundle
/templates/my-env-template/v1/schema/
```

Tras crear la primera configuración de sincronización de plantillas, las nuevas versiones de las plantillas se crean automáticamente al insertar una confirmación que agrega un paquete de plantillas actualizadas a una versión nueva (por ejemplo, en `/my-env-template/v2/`).

```
/templates/                                          # subdirectory (optional)
/templates/my-env-template/                          # template name
/templates/my-env-template/v1/                       # template version
/templates/my-env-template/v1/infrastructure/        # template bundle
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

Puedes incluir nuevas versiones de paquetes de plantillas para una o más plantillas configuradas para la sincronización en una sola confirmación. AWS Proton crea una nueva versión de plantilla para cada nueva versión del paquete de plantillas que se incluyó en la confirmación.

Después de crear la configuración de sincronización de plantillas, aún puede crear nuevas versiones de la plantilla manualmente en la consola o con ella AWS CLI cargando paquetes de plantillas desde un bucket de S3. La sincronización de plantillas solo funciona en una dirección: desde tu repositorio hasta. AWS Proton Las versiones de plantillas creadas manualmente no se sincronizan.

Después de configurar una configuración de sincronización de plantillas, AWS Proton escucha los cambios en tu repositorio. Cada vez que se realiza un cambio, se busca cualquier directorio que tenga el mismo nombre que la plantilla. A continuación, busca dentro de ese directorio cualquier directorio que parezca una versión principal. AWS Proton registra el paquete de plantillas en la versión principal de la plantilla correspondiente. Las nuevas versiones están siempre en el estado `DRAFT`. Puede publicar las nuevas versiones con la consola o AWS CLI.

Por ejemplo, supongamos que tiene una plantilla llamada `my-env-template` configurada para sincronizarse desde `my-repo/templates` en una ramificación `main` con el siguiente diseño.

```
/code
/code/service.go
README.md
/templates/
/templates/my-env-template/
/templates/my-env-template/v1/
/templates/my-env-template/v1/infrastructure/
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

AWS Proton sincroniza el contenido de `/templates/my-env-template/v1/` to `my-env-template:1` y el contenido de `/templates/my-env-template/v2/` tomy`-env-template:2`. Si aún no existen, se crearán estas versiones principales.

AWS Proton encontró el primer directorio que coincide con el nombre de la plantilla. Puede limitar las AWS Proton búsquedas de directorios especificando una configuración de sincronización `subdirectoryPath` al crear o editar una plantilla. Por ejemplo, puede especificar `/production-templates/` para `subdirectoryPath`.

Puede crear una configuración de sincronización de plantillas mediante la consola o la CLI.

AWS Management Console

Cree una plantilla y una configuración de sincronización de plantillas mediante la consola.

1. En la consola de AWS Proton, elija Plantillas de entorno.
2. Seleccione Crear plantilla de entorno.
3. En la página Crear plantilla de entorno, en la sección Opciones de plantilla, seleccione Crear una plantilla para aprovisionar nuevos entornos.
4. En la sección Origen del paquete de plantillas, seleccione Sincronizar plantillas de Git.
5. Consulte la sección Repositorio de código fuente:
   a. En Repositorio, seleccione el repositorio vinculado que contenga el paquete de plantillas.
   b. En Ramificación, seleccione una ramificación del repositorio desde la que realizar la sincronización.
   c. (Opcional) En Directorio de paquetes de plantillas, escriba el nombre de un directorio para limitar la búsqueda del paquete de plantillas.
6. En la sección Detalles de la plantilla.
   a. Introduzca un Nombre de plantilla.
   b. (Opcional) Introduzca un Nombre de visualización de la plantilla.
   c. (Opcional) Introduzca una Descripción de la plantilla para la plantilla de entorno.
7. (Opcional) Marque la casilla Personalización de la configuración de cifrado (avanzada) en la sección Configuración de cifrado para proporcionar su propia clave de cifrado.
8. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.
9. Seleccione Crear plantilla de entorno.

   Ahora aparecerá una nueva página que muestra el estado y los detalles de su nueva plantilla de entorno. Estos detalles incluyen una lista de etiquetas AWS administradas y administradas por el cliente. AWS Proton genera automáticamente etiquetas AWS administradas cuando crea AWS Proton recursos. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.
10. En la página de detalles de la plantilla, seleccione la pestaña Sincronizar para ver los datos detallados de la configuración de sincronización de plantillas.

11. Seleccione la pestaña Versiones de la plantilla para ver las versiones de las plantillas con detalles de estado.
12. El estado de una nueva plantilla de entorno comienza en el estado Borrador. Usted y las personas que gocen de permisos `proton:CreateEnvironment` podrán verla y acceder a ella. Siga el siguiente paso para poner la plantilla a disposición de otras personas.
13. En la sección Versiones de la plantilla, seleccione el botón de radio situado a la izquierda de la versión secundaria de la plantilla que acaba de crear (1.0). Como alternativa, puede seleccionar Publicar en la alerta de información y omitir el siguiente paso.
14. En la sección Versiones de la plantilla, seleccione Publicar.
15. El estado de la plantilla cambiará a Publicado. Es la versión más reciente y recomendada de la plantilla.
16. En el panel de navegación, seleccione Plantillas de entorno para ver una lista de las plantillas de entorno y sus detalles.

El procedimiento para crear una plantilla de servicio y la configuración de sincronización de plantillas es similar.

AWS CLI

Cree una plantilla y una configuración de sincronización de plantillas mediante la AWS CLI.

1. Cree una plantilla. En este ejemplo, se crea una plantilla de entorno.

   Ejecute el siguiente comando de la .

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

   La respuesta será como la que se muestra a continuación.

   ```
   {
       "environmentTemplate": {
           "arn": "arn:aws:proton:us-east-1:123456789012:environment-template/env-
   template",
           "createdAt": "2021-11-07T23:32:43.045000+00:00",
           "displayName": "env-template",
           "lastModifiedAt": "2021-11-07T23:32:43.045000+00:00",
           "name": "env-template",
           "status": "DRAFT",
   ```

```
        "templateName": "env-template"
    }
}
```

2. Cree la configuración de sincronización de su plantilla AWS CLI proporcionando lo siguiente:

   - La plantilla con la que desee sincronizar. Una vez que haya creado la configuración de sincronización de la plantilla, podrá seguir creando nuevas versiones a partir de ella manualmente en la consola o con la AWS CLI.
   - El nombre de la plantilla.
   - El tipo de plantilla.
   - El repositorio vinculado desde el que desee sincronizar.
   - El proveedor del repositorio vinculado.
   - La ramificación en la que se encuentra el paquete de plantillas.
   - (Opcional) La ruta al directorio que contenga el paquete de plantillas. De forma predeterminada, AWS Proton busca el primer directorio que coincida con el nombre de la plantilla.

   Ejecute el siguiente comando de la .

```
$ aws proton create-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-name "myrepos/templates" \
    --repository-provider "GITHUB" \
    --branch "main" \
    --subdirectory "env-template/"
```

   La respuesta será como la que se muestra a continuación.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
```

```
}
```

3. Para publicar la versión de la plantilla, consulte Registro y publicación de plantillas.

Sincronización de plantillas de servicio

En los siguientes ejemplos se muestra cómo se pueden sincronizar las plantillas de entorno. Las plantillas de servicio son similares. Para sincronizar las plantillas de servicio, añada un archivo adicional con el nombre de `.template-registration.yaml` a cada directorio de versiones principales del paquete de plantillas. Este archivo contiene los detalles adicionales que se AWS Proton necesitan cuando se crea una versión de la plantilla de servicio para usted tras una confirmación. Cuando se crea de forma explícita una versión de plantilla de servicio mediante la AWS Proton consola o la API, se proporcionan estos detalles como entradas y este archivo reemplaza estas entradas para la sincronización de la plantilla.

```
./templates/                                                  # subdirectory (optional)
 /templates/my-svc-template/                                  # service template name
 /templates/my-svc-template/v1/                               # service template version
 /templates/my-svc-template/v1/.template-registration.yaml    # service template version
 properties
 /templates/my-svc-template/v1/instance_infrastructure/       # template bundle
 /templates/my-svc-template/v1/schema/
```

El archivo `.template-registration.yaml` contiene los siguientes detalles:

- Entornos compatibles [obligatorio]: los entornos basados en estas plantillas de entorno y en las versiones principales son compatibles con los servicios basados en esta versión de la plantilla de servicio.
- Orígenes de componentes compatibles [opcional]: los componentes que utilizan estos orígenes son compatibles con los servicios en función de esta versión de plantilla de servicio. Si no se especifica, los componentes no se pueden conectar a estos servicios. Para obtener más información sobre los componentes, consulte Componentes.

La sintaxis YAML del archivo es la siguiente:

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
```

```
  - DIRECTLY_DEFINED
```

Especifique una o más combinaciones de plantillas de entorno o de versiones principales. Especificar `supported_component_sources` es opcional y el único valor admitido es `DIRECTLY_DEFINED`.

Example .template-registration.yaml

En este ejemplo, la versión de la plantilla de servicio es compatible con las versiones principales 1 y 2 de la plantilla de entorno `my-env-template`. También es compatible con las versiones principales 1 y 3 de la plantilla de entorno `another-env-template`. El archivo no especifica `supported_component_sources`, por lo que los componentes no se pueden conectar a los servicios basados en esta versión de la plantilla de servicio.

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
  - another-env-template:3
```

> **Note**
>
> Anteriormente, AWS Proton se definió un archivo diferente para especificar los entornos compatibles. `.compatible-envs` AWS Proton sigue siendo compatible con ese archivo y su formato por motivos de compatibilidad con versiones anteriores. No recomendamos seguir utilizándolo, ya que no es extensible y no admite características más nuevas como los componentes.

Visualización de los detalles de la configuración de sincronización de plantillas

Vea los datos detallados de la configuración de sincronización de plantillas mediante la consola o la CLI.

AWS Management Console

Utilice la consola para ver detalles sobre la configuración sincronizada de plantillas.

1. En el panel de navegación, elija Plantillas (de entorno o de servicio).

2. Para ver los datos detallados, elija el nombre de la plantilla para la que creó una configuración de sincronización de plantillas.
3. En la página de detalles de la plantilla, seleccione la pestaña Sincronización para ver los datos detallados de la configuración de sincronización de plantillas.

AWS CLI

Utilice AWS CLI para ver una plantilla sincronizada.

Ejecute el siguiente comando de la .

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

La respuesta será como la que se muestra a continuación.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
        "templateType": "SERVICE"
    }
}
```

Utilice AWS CLI para obtener el estado de sincronización de la plantilla.

Para `template-version`, introduzca la versión principal de la plantilla.

Ejecute el siguiente comando de la .

```
$ aws proton get-template-sync-status \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

Edición de la configuración de sincronización de una plantilla

Puede editar cualquiera de los parámetros de configuración de sincronización de plantillas, excepto `template-name` y `template-type`.

Obtenga información sobre cómo editar la configuración sincronizada de plantillas mediante la consola o la CLI.

AWS Management Console

Edite una ramificación de configuración de sincronización de plantillas mediante la consola.

En la lista de plantillas.

1. En la consola de AWS Proton, elija Plantillas (de entorno o de servicio).
2. En la lista de plantillas, elija el nombre de la plantilla con la configuración sincronizada de plantillas que desee editar.
3. En la página de detalles de la plantilla, seleccione la pestaña Sincronización de plantillas.
4. En la sección Detalles de la sincronización de plantillas, elija Editar.
5. En la página Editar, en la sección Repositorio de código fuente, en Ramificación, seleccione una ramificación y, a continuación, seleccione Guardar configuración.

AWS CLI

El siguiente ejemplo de comando y respuesta muestra cómo se puede editar una **`branch`** de configuración de sincronización de plantillas mediante la CLI.

Ejecute el siguiente comando de la .

```
$ aws proton update-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-provider "GITHUB" \
    --repository-name "myrepos/templates" \
    --branch "fargate" \
    --subdirectory "env-template"
```

La respuesta será como la que se muestra a continuación.

```
{
```

```
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

También puede utilizarla AWS CLI para actualizar las plantillas de servicios sincronizadas.

Eliminación de una configuración sincronizada de plantillas

Elimine una configuración sincronizada de plantillas mediante la consola o la CLI.

AWS Management Console

Elimine una configuración sincronizada de plantillas mediante la consola.

1. En la página de detalles de la plantilla, seleccione la pestaña Sincronizar.
2. En la sección de Detalles de sincronización, seleccione Desconectar.

AWS CLI

Los siguientes comandos y respuestas de ejemplo muestran cómo utilizarlos AWS CLI para eliminar las configuraciones de plantillas sincronizadas.

Ejecute el siguiente comando de la .

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

La respuesta será como la que se muestra a continuación.

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
```

```
    }
}
```

Configuraciones de sincronización de servicios

Con la sincronización de servicios, se pueden configurar e implementar servicios de AWS Proton mediante Git. Se puede utilizar la sincronización de servicios para administrar las implementaciones y actualizaciones iniciales de servicios de AWS Proton con una configuración definida en un repositorio de Git. A través de Git, puede utilizar características como el seguimiento de versiones y las solicitudes de extracción para configurar, administrar e implementar los servicios. La sincronización de servicios combina AWS Proton y Git para ayudar al usuario a aprovisionar una infraestructura estandarizada que se defina y administre mediante plantillas de AWS Proton. Además administra las definiciones de servicios en el repositorio de Git del usuario y reduce la necesidad de tener que cambiar de herramienta. En comparación con el uso exclusivo de Git, la estandarización de las plantillas y la implementación en AWS Proton ayudan al usuario a dedicar menos tiempo a administrar su infraestructura. AWS Proton también proporciona una mayor transparencia y la posibilidad de realizar auditorías tanto para los desarrolladores como para los equipos de plataformas.

Archivo OPS de AWS Proton

El archivo `proton-ops` define dónde AWS Proton debe encontrar el archivo de especificaciones que se utiliza para actualizar la instancia de servicio. También define en qué orden actualizar las instancias de servicio y cuándo promover los cambios de una instancia a otra.

El archivo `proton-ops` permite sincronizar una instancia de servicio mediante el archivo de especificaciones, o varios archivos de especificaciones, que se encuentran en el repositorio vinculado. Para ello, defina un bloque de sincronización en el archivo de `proton-ops`, como en el siguiente ejemplo.

Ejemplo ./configuration/proton-ops.yaml:

```
sync:
  services:
      frontend-svc:
          alpha:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
```

```
        beta:
            branch: dev
            spec: ./frontend-svc/test/frontend-spec.yaml
        gamma:
            branch: pre-prod
            spec: ./frontend-svc/pre-prod/frontend-spec.yaml
        prod-one:
            branch: prod
            spec: ./frontend-svc/prod/frontend-spec-second.yaml
        prod-two:
            branch: prod
            spec: ./frontend-svc/prod/frontend-spec-second.yaml
        prod-three:
            branch: prod
            spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

En el ejemplo anterior, `frontend-svc` es el nombre del servicio y `alpha`, `beta`, `gamma`, `prod-one`, `prod-two` y `prod-three` son las instancias.

El archivo `spec` puede ser todas las instancias o un subconjunto de las instancias definidas en el archivo `proton-ops`. Sin embargo, como mínimo, debe tener la instancia definida en la ramificación y la especificación desde la que se sincronice. Si las instancias no están definidas en el archivo `proton-ops`, con la ramificación específica y la ubicación del archivo `spec`, el servicio de sincronización no creará ni actualizará esas instancias.

El siguiente ejemplo muestra el posible aspecto de los archivos `spec`. Recuerde que el archivo `proton-ops` se sincroniza desde estos archivos `spec`.

Ejemplo **`./frontend-svc/test/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "alpha"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "beta"
  environment: "frontend-env"
  spec:
```

```
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Ejemplo **`./frontend-svc/pre-prod/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "gamma"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Ejemplo **`./frontend-svc/prod/frontend-spec-second.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "prod-one"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-two"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-three"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Si una instancia no se sincroniza y hay un problema persistente al intentar sincronizarla, llamar a la API `GetServiceInstanceSyncStatus` puede ayudar a resolver el problema.

> **Note**
>
> Los límites de AWS Proton siguen restringiendo a los clientes que utilizan la sincronización de servicios.

Bloqueadores

Al sincronizar el servicio de AWS Proton mediante la sincronización de servicios, el usuario puede actualizar las especificaciones del servicio, así como crear y actualizar instancias de servicio desde su repositorio de Git. Sin embargo, puede haber ocasiones en las que sea necesario actualizar un servicio o una instancia manualmente a través de la AWS Management Console o la AWS CLI.

AWS Proton ayuda a evitar sobrescribir los cambios manuales que se realicen a través de la AWS Management Console o la AWS CLI, como por ejemplo actualizar una instancia de servicio o eliminar una instancia de servicio. Para conseguirlo, AWS Proton crea automáticamente un bloqueador de sincronización de servicios al deshabilitar la sincronización de servicios cuando detecta un cambio manual.

Para obtener todos los bloqueadores asociados a un servicio, se debe hacer lo siguiente para cada `serviceInstance` asociada al servicio:

- Llamar a la API `getServiceSyncBlockerSummary` solo con el `serviceName`.
- Llamar a la API `getServiceSyncBlockerSummary` con el `serviceName` y el `serviceInstanceName`.

Esto devuelve una lista de los bloqueadores más recientes y el estado asociado a ellos. Si algún bloqueador está marcado como ACTIVO, debe resolverlo mediante una llamada a la API `UpdateServiceSyncBlocker` con el `blockerId` y `resolvedReason` para cada uno de ellos.

Si actualiza o crea una instancia de servicio manualmente, AWS Proton creará un bloqueador de sincronización de servicios en la instancia de servicio. AWS Proton continuará sincronizando todas las demás instancias de servicio, pero deshabilitará la sincronización de esta instancia de servicio hasta que se resuelva el bloqueador. Si elimina una instancia de servicio de un servicio, AWS Proton creará un bloqueador de sincronización de servicios en el servicio. Esto impide a AWS Proton que sincronice las instancias de servicio hasta que se haya resuelto el bloqueador.

Una vez que tenga todos los bloqueadores activos, deberá resolverlos mediante una llamada a la API `UpdateServiceSyncBlocker` con el `blockerId` y `resolvedReason` para cada uno de los bloqueadores activos.

Mediante la AWS Management Console, puede determinar si la sincronización de un servicio está deshabilitada; para ello, tendrá que dirigirse a AWS Proton y seleccionar la pestaña Sincronización de servicios. Si el servicio o las instancias de servicio están bloqueados, aparecerá el botón Habilitar. Para habilitar la sincronización de servicios, seleccione Habilitar.

Temas

Creación de una configuración de sincronización de servicios

Puede crear una configuración de sincronización de servicios mediante la consola o la AWS CLI.

AWS Management Console

1. En la página Elegir una plantilla de servicio, seleccione una plantilla y elija Configurar.
2. En la página Configurar servicio, en la sección Detalles del servicio, introduzca un nuevo Nombre del servicio.
3. (Opcional) Escriba una descripción para el servicio.
4. En la sección Repositorio del código fuente de la aplicación, seleccione Elegir un repositorio Git vinculado para seleccionar un repositorio al que ya se haya vinculado con AWS Proton. Si aún no tiene un repositorio vinculado, seleccione Vincular otro repositorio de Git y siga las instrucciones en Crear un enlace al repositorio.
5. En Repositorio, elija de la lista el nombre del repositorio que contenga el código fuente.
6. En Ramificación, elija de la lista el nombre de la ramificación del repositorio para el código fuente.
7. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.
8. Elija Siguiente.

9. En la página Configurar instancias de servicio, en la sección Fuente de definición del servicio, seleccione Sincronice el servicio desde Git.
10. En la sección Archivos de definición de servicio, si desea que AWS Proton cree el archivo `proton-ops`, seleccione Quiero que AWS Proton cree los archivos. Con esta opción, AWS Proton creará la `spec` y el archivo `proton-ops` en las ubicaciones que especifique el usuario. Seleccione Voy a proporcionar mis propios archivos para crear su propio archivo OPS.
11. En la sección Repositorio de definición de servicios, seleccione Elegir un repositorio Git vinculado para seleccionar un repositorio al que ya se haya vinculado con AWS Proton.
12. En Nombre del repositorio, elija de la lista el nombre del repositorio del código fuente.
13. Para la ramificación del archivo **`proton-ops`**, elija de la lista el nombre de la ramificación en la que AWS Proton colocará el OPS y el archivo de especificaciones.
14. En la sección Instancias de servicio, cada campo se rellena automáticamente en función de los valores del archivo `proton-ops`.
15. Seleccione Siguiente y revise las entradas.
16. Seleccione Crear.

AWS CLI

Creación de una configuración de sincronización de servicios mediante la AWS CLI

- Ejecute el siguiente comando.

```
$ aws proton create-service-sync-config \
    --resource "service-arn" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --proton-ops-file "./configuration/custom-proton-ops.yaml" (optional)
```

La respuesta será como la que se muestra a continuación.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
```

```
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Visualización de los detalles de configuración de una sincronización de servicios

Puede ver los datos de los detalles de configuración de una sincronización de servicios mediante la consola o la AWS CLI.

AWS Management Console

Uso de la consola para ver los detalles de configuración de una sincronización de servicios

1. En el panel de navegación, elija Servicios.
2. Para ver los datos detallados, elija el nombre de un servicio para el que haya creado una configuración de sincronización de servicios.
3. En la página de detalles del servicio, seleccione la pestaña Sincronización de servicios para ver los datos detallados de la configuración de la sincronización de servicios.

AWS CLI

Utilice la AWS CLI para obtener un servicio sincronizado.

Ejecute el siguiente comando.

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

La respuesta será como la que se muestra a continuación.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
```

```
    }
}
```

Utilice la AWS CLI para obtener el estado de sincronización de servicios.

Ejecute el siguiente comando.

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

Edición de una configuración de sincronización de servicios

Puede editar una configuración de sincronización de servicios mediante la consola de la AWS CLI.

AWS Management Console

Edite una configuración de sincronización de servicios mediante la consola.

1. En el panel de navegación, elija Servicios.
2. Para ver los datos detallados, elija el nombre de un servicio para el que haya creado una configuración de sincronización de servicios.
3. En la página de detalles del servicio, seleccione la pestaña Sincronización de servicios.
4. En la sección Usuarios, seleccione Editar.
5. En la página Editar, actualice la información que desee editar y, a continuación, seleccione Guardar.

AWS CLI

En el siguiente comando y respuesta de ejemplo se muestra cómo se puede editar la configuración de sincronización de servicios mediante la AWS CLI.

Ejecute el siguiente comando.

```
$ aws proton update-service-sync-config \
    --service-name "service name" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
```

```
    --ops-file-branch "main" \
    --ops-file "./configuration/custom-proton-ops.yaml"
```

La respuesta será como la que se muestra a continuación.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Eliminación de una configuración de sincronización de servicios

Puede eliminar una configuración de sincronización de servicios mediante la consola o la AWS CLI.

AWS Management Console

Eliminación de una configuración de sincronización de servicios mediante la consola

1. En la página de detalles del servicio, seleccione la pestaña Sincronización de servicios.
2. En la sección de Detalles de la sincronización de servicios, seleccione Desconectar para desconectar el repositorio. Una vez desconectado el repositorio, dejaremos de sincronizar el servicio desde ese repositorio.

AWS CLI

Los siguientes comandos y respuestas de ejemplo muestran cómo utilizar la AWS CLI para eliminar las configuraciones sincronizadas de servicios.

Ejecute el siguiente comando.

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

La respuesta será como la que se muestra a continuación.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> La sincronización de servicios no elimina las instancias de servicio. Solo elimina configuración.

Entornos de AWS Proton

En AWS Proton, un entorno representa el conjunto de políticas y recursos compartidos en los que se implementan los servicios de AWS Proton. Pueden contener cualquier recurso que se espere que se comparta entre las instancias de servicio de AWS Proton. Estos recursos pueden incluir VPC, clústeres y equilibradores de carga compartidos o puertas de enlace de API. Se debe crear un entorno de AWS Proton antes de poder implementar un servicio en él.

En esta sección se describe cómo administrar entornos mediante operaciones de creación, visualización, actualización y eliminación. Para obtener más información, consulte la Referencia de la API del servicio de AWS Proton.

Temas

Roles de IAM

Con AWS Proton, el usuario puede proporcionar los roles de IAM y las claves de AWS KMS para los recursos de AWS que dicho usuario posea y administre. Estos recursos se aplican posteriormente a los recursos que administren los desarrolladores y que sean de su propiedad. El usuario puede crear un rol de IAM para controlar el acceso de su equipo de desarrolladores a la API de AWS Proton.

Rol de servicio AWS Proton

Al crear un entorno nuevo, se debe proporcionar un rol de servicio de IAM relacionado. El rol contiene todos los permisos necesarios para actualizar toda la infraestructura aprovisionada definida tanto en las plantillas de entorno como en las plantillas de servicio. Para ver ejemplos de roles, consulte AWS Proton rol de servicio para el aprovisionamiento mediante AWS CloudFormation. Si

utiliza conexiones de cuentas de entorno y cuentas de entorno, cree el rol en una cuenta de entorno seleccionada. Para obtener más información, consulte Creación de un entorno en una cuenta y aprovisionamiento en otra cuenta y Conexiones de cuentas de entorno.

La forma en la que se proporcione este rol de servicio y la persona que lo asuma dependerán del método de aprovisionamiento del entorno.

- Aprovisionamiento administrado por AWS: el rol se proporciona directamente a AWS Proton, ya sea al crear un entorno o de forma indirecta a través de las conexiones de cuentas. AWS Proton asume el rol en la cuenta correspondiente para aprovisionar el entorno y la infraestructura del servicio.
- Aprovisionamiento autoadministrado: es responsabilidad del usuario configurar la automatización del aprovisionamiento para que asuma el rol adecuado con las credenciales adecuadas cuando una solicitud de extracción (PR) desencadene una acción de aprovisionamiento. Para ver un ejemplo de la acción de GitHub que asume un rol, consulte Asumir un rol en la documentación de la Acción “Configurar credenciales de AWS” para acciones de GitHub.

Para obtener más información acerca de los métodos de aprovisionamiento, consulte the section called “Métodos de aprovisionamiento”.

Creación de un entorno

Obtenga información sobre cómo crear entornos de AWS Proton.

Puede crear un entorno de AWS Proton de las dos formas siguientes:

- Cree, administre y aprovisione un entorno estándar mediante una plantilla de entorno estándar. AWS Proton aprovisiona la infraestructura para el entorno del usuario.
- Conecte AWS Proton a la infraestructura administrada por el cliente mediante una plantilla de entorno administrado por el cliente. El usuario aprovisiona sus propios recursos compartidos de AWS Proton de forma externa y, a continuación, proporciona las salidas de aprovisionamiento que AWS Proton puede utilizar.

Puede elegir uno de los diversos enfoques de aprovisionamiento al crear un entorno.

- Aprovisionamiento administrado por AWS: cree, administre y aprovisione un entorno en una sola cuenta. AWS Proton aprovisiona el entorno.

Este método solo admite plantillas de infraestructura como código (IaC) de CloudFormation.

- Aprovisionamiento administrado por AWS a otra cuenta: en una sola cuenta de administración, cree y administre un entorno que esté aprovisionado en otra cuenta con conexiones a cuentas de entorno. AWS Proton aprovisiona el entorno en la otra cuenta. Para obtener más información, consulte Creación de un entorno en una cuenta y aprovisionamiento en otra cuenta y Conexiones de cuentas de entorno.

  Este método solo es compatible con las plantillas de IaC de CloudFormation.

- Aprovisionamiento autoadministrado: AWS Proton envía las solicitudes de extracción de aprovisionamiento a un repositorio vinculado con la propia infraestructura de aprovisionamiento del usuario.

  Este método solo es compatible con las plantillas de IaC de Terraform.

- Aprovisionamiento de CodeBuild: AWS Proton utiliza AWS CodeBuild para ejecutar los comandos del intérprete de comandos que proporcione el usuario. Los comandos pueden leer las entradas que proporciona AWS Proton, y son responsables de aprovisionar o desaprovisionar la infraestructura y generar valores de salida. Un paquete de plantillas para este método incluye los comandos en un archivo de manifiesto y todos los programas, scripts u otros archivos que estos comandos puedan necesitar.

  Como ejemplo del uso del aprovisionamiento de CodeBuild, puede incluir código que utilice el AWS Cloud Development Kit (AWS CDK) para aprovisionar recursos de AWS y un manifiesto que instale el CDK y ejecute su código de CDK.

  Para obtener más información, consulte the section called “CodeBuild paquete”.

  > **Note**
  >
  > Puede utilizar el aprovisionamiento de CodeBuild con entornos y servicios. En este momento, no puede aprovisionar componentes de esta forma.

Con el aprovisionamiento administrado por AWS (tanto en la misma cuenta como en otra cuenta), AWS Proton realiza llamadas directas para aprovisionar los recursos del usuario.

Con el aprovisionamiento autoadministrado, AWS Proton realiza solicitudes de extracción para proporcionar archivos de IaC compilados que el motor de IaC utiliza para aprovisionar recursos.

Para obtener más información, consulte the section called “Métodos de aprovisionamiento”, the section called “Paquetes de plantillas” y the section called “Requisitos del esquema del entorno”.

Temas

- Creación y aprovisionamiento de un entorno estándar en la misma cuenta
- Creación de un entorno en una cuenta y aprovisionamiento en otra cuenta
- Creación y aprovisionamiento de un entorno mediante el aprovisionamiento autoadministrado

Creación y aprovisionamiento de un entorno estándar en la misma cuenta

Utilice la consola o la AWS CLI para crear y aprovisionar un entorno en una sola cuenta. AWS administra el aprovisionamiento.

AWS Management Console

Uso de la consola para crear y aprovisionar un entorno en una sola cuenta

1. En la consola de AWS Proton, elija Entornos.
2. Seleccione Crear entorno.
3. En la página Elegir una plantilla de entorno, seleccione una plantilla y elija Configurar.
4. En la página Configurar entorno, en la sección Aprovisionamiento, elija Aprovisionamiento administrado por AWS.
5. En la sección Cuenta de implementación, elija Esta Cuenta de AWS.
6. En la página Configurar entorno, en la sección Configuración del entorno, introduzca un Nombre del entorno.
7. (Opcional) Escriba una descripción para el entorno.
8. En la sección Roles de entorno, seleccione el rol de servicio de AWS Proton que se creó como parte de Configuración de funciones de AWS Proton servicio.
9. (Opcional) En la sección Rol de componente, seleccione un rol de servicio que permita que los componentes definidos directamente se ejecuten en el entorno y reduzca los recursos que pueden aprovisionar. Para obtener más información, consulte Componentes.
10. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.
11. Elija Siguiente.

12. En la página Configurar ajustes personalizados del entorno, debe introducir valores para los parámetros `required`. Puede introducir valores para los parámetros `optional` o bien utilizar los valores predeterminados si los hubiera.
13. Seleccione Siguiente y revise las entradas.
14. Seleccione Crear.

    Consulte los detalles y el estado del entorno, así como las etiquetas administradas por AWS y las etiquetas administradas por el cliente para dicho entorno.
15. En el panel de navegación, elija Entornos.

    Una nueva página mostrará una lista de sus entornos junto con el estado y otros detalles del entorno.

AWS CLI

Utilice la AWS CLI para crear y aprovisionar un entorno en una sola cuenta.

Para crear un entorno, especifique el ARN del rol de servicio de AWS Proton, la ruta al archivo de especificaciones, el nombre del entorno, el ARN de la plantilla de entorno, las versiones principal y secundaria y la descripción (opcional).

Los siguientes ejemplos muestran un archivo de especificaciones con formato YAML que especifica los valores de dos entradas en función de lo que se haya definido en el archivo de esquema de la plantilla del entorno. Puede utilizar el comando `get-environment-template-minor-version` para ver el esquema de la plantilla de entorno.

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

Cree un entorno mediante la ejecución del siguiente comando.

```
$ 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"
```

Respuesta:

```
{
    "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"
    }
}
```

Tras crear un nuevo entorno, se podrá ver una lista de etiquetas administradas por el cliente y por AWS, tal y como se muestra en el siguiente comando de ejemplo. AWS Proton genera automáticamente las etiquetas administradas por AWS para el usuario. También puede modificar y crear etiquetas administradas por el cliente mediante la AWS CLI. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

Comando:

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

Creación de un entorno en una cuenta y aprovisionamiento en otra cuenta

Utilice la consola o la AWS CLI cree un entorno estándar en una cuenta de administración que aprovisione la infraestructura del entorno en otra cuenta. AWS administra el aprovisionamiento.

Antes de utilizar la consola o la CLI, complete los siguientes pasos.

1. Identifique los ID de las Cuenta de AWS para la cuenta de administración y de entorno y cópielos para utilizarlos más adelante.
2. En la cuenta de entorno, cree un rol de servicio AWS Proton con los permisos mínimos que desee crear en el entorno. Para obtener más información, consulte AWS Proton rol de servicio para el aprovisionamiento mediante AWS CloudFormation.

AWS Management Console

Utilice la consola para crear un entorno en una cuenta y aprovisionar dicho entorno en otra.

1. En la cuenta de entorno, cree una conexión de cuenta de entorno y utilícela para enviar una solicitud de conexión a la cuenta de administración.

   a. En la consola de AWS Proton, seleccione Conexiones de cuentas de entorno en el panel de navegación.

   b. En la página Conexiones de cuentas de entorno, seleccione Solicitud de conexión.

   > **Note**
   >
   > Compruebe que el ID de la cuenta que aparece en el encabezado de la página Conexión de cuenta del entorno coincida con el ID de la cuenta del entorno previamente identificado.

   c. En la página Solicitud de conexión, en la sección Rol de entorno, seleccione el Rol de servicio existente y el nombre del rol de servicio que se creó para el entorno.

   d. En la sección Conectarse a una cuenta de administración, introduzca el ID de la cuenta de administración y un Nombre del entorno para el entorno de AWS Proton. Copie el nombre para utilizarlo más adelante.

   e. Seleccione Solicitud de conexión en la esquina inferior derecha de la página.

   f. Su solicitud aparecerá como pendiente en la tabla de conexiones del entorno enviadas a una cuenta de administración y un modal mostrará cómo aceptar la solicitud de la cuenta de administración.

2. En la cuenta de administración, acepte una solicitud de conexión desde la cuenta de entorno.

   a. Inicie sesión en la cuenta de administración y seleccione Conexiones de cuentas de entorno en la consola de AWS Proton.

   b. En la página Conexiones de cuentas de entorno, en la tabla Solicitudes de conexión de cuentas de entorno, seleccione la conexión de la cuenta de entorno con el ID de la cuenta de entorno que coincida con el ID de la cuenta de entorno previamente identificado.

> **Note**
>
> Compruebe que el ID de la cuenta que aparece en el encabezado de la página Conexión de cuenta del entorno coincida con el ID de la cuenta de administración previamente identificado.

c. Elija Aceptar. El estado cambia de PENDIENTE a CONECTADO.

3. En la cuenta de administración, cree un entorno.

a. En el panel de navegación, elija Plantillas de entorno.

b. En la página de Plantillas de entorno, elija Crear plantilla de entorno.

c. En la página Elegir una plantilla de entorno, elija una plantilla de entorno.

d. En la página Configurar entorno, en la sección Aprovisionamiento, elija Aprovisionamiento administrado por AWS.

e. En la sección Cuenta de implementación, elija Otra cuenta de AWS.

f. En la sección Detalles del entorno, seleccione la Conexión de cuenta de entorno y el Nombre del entorno.

g. Elija Siguiente.

h. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisar y crear.

i. Revise y elija Crear entorno.

AWS CLI

Utilice la AWS CLI para crear un entorno en una cuenta y aprovisionar dicho entorno en otra.

En la cuenta de entorno, cree una conexión de cuenta de entorno y solicite la conexión; para ello, ejecute el siguiente comando.

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

En la cuenta de administración, acepte la solicitud de conexión a la cuenta de entorno; para ello, ejecute el siguiente comando.

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Ejecute el siguiente comando para ver la conexión de cuenta de entorno.

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

En la cuenta de administración, ejecute el siguiente comando para crear un entorno.

```
$ 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"
```

Respuesta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-
connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
```

```
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
        "templateName": "simple-env-template"
    }
}
```

Creación y aprovisionamiento de un entorno mediante el aprovisionamiento autoadministrado

Al utilizar el aprovisionamiento autoadministrado, AWS Proton envía las solicitudes de extracción de aprovisionamiento a un repositorio vinculado con la propia infraestructura de aprovisionamiento del usuario. Las solicitudes de extracción inician el flujo de trabajo propio del usuario, que consiste en llamar a los servicios de AWS para aprovisionar la infraestructura.

Consideraciones sobre el aprovisionamiento autoadministrado:

- Antes de crear un entorno, configure un directorio de recursos del repositorio para el aprovisionamiento autoadministrado. Para obtener más información, consulte AWS Proton infraestructura como archivos de código.
- Tras crear el entorno, AWS Proton espera a recibir notificaciones asincrónicas sobre el estado del aprovisionamiento de la infraestructura. El código de aprovisionamiento debe utilizar la API de AWS Proton `NotifyResourceStateChange` para enviar estas notificaciones asíncronas a AWS Proton.

Puede utilizar el aprovisionamiento autoadministrado en la consola o con la AWS CLI. En los siguientes ejemplos se muestra cómo se puede utilizar el aprovisionamiento autoadministrado con Terraform.

AWS Management Console

Utilice la consola para crear un entorno de Terraform mediante el aprovisionamiento autoadministrado.

1. En la consola de AWS Proton, elija Entornos.
2. Seleccione Crear entorno.
3. En la página Elegir una plantilla de entorno, seleccione una plantilla de Terraform y elija Configurar.

4. En la página Configurar entorno, en la sección Aprovisionamiento, elija Aprovisionamiento autoadministrado.
5. En la sección Detalles del repositorio de aprovisionamiento:

   a. Si aún no ha vinculado su repositorio de aprovisionamiento a AWS Proton, seleccione Nuevo repositorio, elija uno de los proveedores de repositorios y, a continuación, para la Conexión de Codestar, elija una de las conexiones.

   > **Note**
   >
   > Si aún no tiene ninguna conexión a la cuenta del proveedor de repositorios correspondiente, seleccione Añadir una nueva conexión de CodeStar. A continuación, cree una conexión y después seleccione el botón de actualización situado junto al menú Conexión de CodeStar. Ahora debería poder elegir su nueva conexión en el menú.

   Si ya ha vinculado su repositorio a AWS Proton, seleccione Repositorio existente.

   b. En Nombre del repositorio, elija un repositorio. El menú desplegable muestra los repositorios vinculados para el Repositorio existente o la lista de repositorios de la cuenta del proveedor para el Nuevo repositorio.

   c. En Nombre de la ramificación, elija alguna de las ramificaciones del repositorio.
6. En la sección Configuración del entorno, introduzca un Nombre del entorno.
7. (Opcional) Escriba una descripción para el entorno.
8. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.
9. Elija Siguiente.
10. En la página Configurar ajustes personalizados del entorno, debe introducir valores para los parámetros `required`. Puede introducir valores para los parámetros `optional` o bien utilizar los valores predeterminados si los hubiera.
11. Seleccione Siguiente y revise las entradas.
12. Seleccione Crear para enviar una solicitud de extracción.

    - Si aprueba la solicitud de extracción, la implementación estará en curso.
    - Si rechaza la solicitud de extracción, se cancelará la creación del entorno.

- Si se agota el tiempo de espera de la solicitud de extracción, la creación del entorno no se completará.

13. Consulte los detalles y el estado del entorno, así como las etiquetas administradas por AWS y las etiquetas administradas por el cliente para dicho entorno.
14. En el panel de navegación, elija Entornos.

    Una nueva página mostrará una lista de sus entornos junto con el estado y otros detalles del entorno.

AWS CLI

Al crear un entorno mediante el aprovisionamiento autoadministrado, se añade el parámetro `provisioningRepository` y se omiten los parámetros `ProtonServiceRoleArn` y `environmentAccountConnectionId`.

Utilice la AWS CLI para crear un entorno de Terraform mediante el aprovisionamiento autoadministrado.

1. Cree un entorno y envíe una solicitud de extracción al repositorio para su revisión y aprobación.

   Los siguientes ejemplos muestran un archivo de especificaciones con formato YAML que define los valores de dos entradas en función del archivo de esquema de la plantilla del entorno. Puede utilizar el comando `get-environment-template-minor-version` para ver el esquema de la plantilla de entorno.

   Especificación:

   ```
   proton: EnvironmentSpec
   spec:
     ssm_parameter_value: "test"
   ```

   Cree un entorno mediante la ejecución del siguiente comando.

   ```
   $ aws proton create-environment \
       --name "pr-environment" \
       --template-name "pr-env-template" \
       --template-major-version "1" \
   ```

```
    --provisioning-repository="branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec.yaml"
```

Respuesta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T17:06:58.679000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T17:06:58.679000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateName": "pr-env-template"
    }
```

2. Revise la solicitud.

   - Si aprueba la solicitud, el aprovisionamiento pasará a estar en curso.
   - Si rechaza la solicitud, se cancelará la creación del entorno.
   - Si se agota el tiempo de espera de la solicitud de extracción, la creación del entorno no se completará.

3. Proporcione el estado de aprovisionamiento de forma asíncrona a AWS Proton. En el siguiente ejemplo, se notifica a AWS Proton que el aprovisionamiento se ha realizado correctamente.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

Visualización de datos del entorno

Puede ver los datos detallados del entorno mediante la consola de AWS Proton o la AWS CLI.

AWS Management Console

Puede ver listas de entornos con detalles y entornos individuales con datos detallados mediante la consola de AWS Proton.

1. Para ver una lista de los entornos, seleccione Entornos en el panel de navegación.
2. Para ver los datos detallados, seleccione el nombre de algún entorno.

   Vea los datos detallados del entorno seleccionado.

AWS CLI

Utilice la AWS CLI para obtener o enumerar los detalles del entorno.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Actualización de un entorno

Si el entorno de AWS Proton está asociado a una conexión de cuenta de entorno, no actualice ni incluya el parámetro `protonServiceRoleArn` para actualizar o conectarse a una conexión de cuenta de entorno.

Solo puede actualizar a una nueva conexión de cuenta de entorno si se cumplen las dos condiciones siguientes:

- La conexión de cuenta de entorno se creó en la misma cuenta de entorno en la que se creó la conexión de cuenta de entorno actual.
- La conexión de cuenta de entorno está asociada al entorno actual.

Si el entorno no está asociado a una conexión de cuenta de entorno, no actualice ni incluya el parámetro `environmentAccountConnectionId`.

Puede actualizar el parámetro y el valor `environmentAccountConnectionId` o `protonServiceRoleArn`. No puede actualizar ambos.

Si su entorno utiliza un aprovisionamiento autoadministrado, no actualice el parámetro `provisioning-repository` y omita los parámetros `environmentAccountConnectionId` y `protonServiceRoleArn`.

Existen cuatro modos de actualizar un entorno, tal y como se describe en la siguiente lista. Al utilizar la AWS CLI, el campo `deployment-type` define el modo. Al utilizar la consola, estos modos se asignan a las acciones Editar, Actualizar, Actualización secundaria y Actualización principal que aparecen en el menú desplegable Acciones.

`NONE`

En este modo, no se produce ninguna implementación. Solo se actualizan los parámetros de metadatos solicitados.

`CURRENT_VERSION`

En este modo, el entorno se implementa y actualiza con las nuevas especificaciones que el usuario proporcione. Solo se actualizan los parámetros solicitados. No incluya parámetros de versiones principales o secundarias cuando utilice este `deployment-type`.

`MINOR_VERSION`

En este modo, el entorno se implementa y actualiza con la versión secundaria recomendada (más reciente) publicada de la versión principal actual que se utiliza de forma predeterminada. También puede especificar una versión secundaria diferente de la versión principal que se está utilizando actualmente.

`MAJOR_VERSION`

En este modo, el entorno se implementa y actualiza con las versiones principal y secundaria recomendadas (más recientes) publicadas de la plantilla actual que se utiliza de forma predeterminada. También puede especificar una versión principal diferente que sea superior a la versión principal en uso y una versión secundaria (opcional).

Temas

Actualización de un entorno de aprovisionamiento administrado por AWS

El aprovisionamiento estándar solo es compatible con los entornos que se aprovisionan con AWS CloudFormation.

Utilice la consola o la AWS CLI para actualizar el entorno.

AWS Management Console

Actualice un entorno mediante la consola como se muestra en los pasos siguientes.

1. Elija uno de los dos pasos siguientes.

    a. En la lista de entornos.

        i. En la consola de AWS Proton, elija Entornos.

        ii. En la lista de entornos, elija el botón de radio a la izquierda del entorno que desee actualizar.

b. En la página de detalles del entorno de la consola.

i. En la consola de AWS Proton, elija Entornos.

ii. En la lista de entornos, elija el nombre del entorno que desee actualizar.

2. Elija uno de los siguientes cuatro pasos para actualizar el entorno.

a. Llevar a cabo alguna edición que no requiera la implementación del entorno.

i. Por ejemplo, cambiar una descripción.

Elija Editar.

ii. Rellene el formulario y seleccione Siguiente.

iii. Revise la edición y seleccione Actualizar.

b. Actualizar únicamente las entradas de metadatos.

i. Elija Acciones y, a continuación, Actualizar.

ii. Rellene el formulario y seleccione Editar.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

c. Actualizar a una nueva versión secundaria de su plantilla de entorno.

i. Elija Acciones y, a continuación, Actualización secundaria.

ii. Rellene el formulario y seleccione Siguiente.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

d. Actualizar a una nueva versión principal de su plantilla de entorno.

i. Elija Acciones y, a continuación, Actualización principal.

ii. Rellene el formulario y seleccione Siguiente.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

AWS CLI

Utilice la AWS Proton AWS CLI para actualizar un entorno a una nueva versión secundaria.

Ejecute el siguiente comando para actualizar el entorno:

```
$ 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"
```

Respuesta:

```
{
    "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"
    }
}
```

Ejecute el siguiente comando para obtener y confirmar el estado:

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

Respuesta:

```
{
    "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"
    }
}
```

Actualización de entornos de aprovisionamiento autoadministrados

El aprovisionamiento autoadministrado solo es compatible con los entornos que se aprovisionan con Terraform.

Utilice la consola o la AWS CLI para actualizar el entorno.

AWS Management Console

Actualice un entorno mediante la consola como se muestra en los pasos siguientes.

1. Elija uno de los dos pasos siguientes.
   a. En la lista de entornos.
      i. En la consola de AWS Proton, elija Entornos.
      ii. En la lista de entornos, elija el botón de radio a la izquierda de la plantilla de entorno que desee actualizar.
   b. En la página de detalles del entorno de la consola.
      i. En la consola de AWS Proton, elija Entornos.
      ii. En la lista de entornos, elija el nombre del entorno que desee actualizar.
2. Elija uno de los siguientes cuatro pasos para actualizar el entorno.
   a. Llevar a cabo alguna edición que no requiera la implementación del entorno.
      i. Por ejemplo, cambiar una descripción.

Elija Editar.

ii. Rellene el formulario y seleccione Siguiente.

iii. Revise la edición y seleccione Actualizar.

b. Actualizar únicamente las entradas de metadatos.

i. Elija Acciones y, a continuación, Actualizar.

ii. Rellene el formulario y seleccione Editar.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

c. Actualizar a una nueva versión secundaria de su plantilla de entorno.

i. Elija Acciones y, a continuación, Actualización secundaria.

ii. Rellene el formulario y seleccione Siguiente.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

d. Actualizar a una nueva versión principal de su plantilla de entorno.

i. Elija Acciones y, a continuación, Actualización principal.

ii. Rellene el formulario y seleccione Siguiente.

iii. Rellene los formularios y seleccione Siguiente hasta llegar a la página Revisión.

iv. Revise las actualizaciones y seleccione Actualizar.

AWS CLI

Utilice la AWS CLI para actualizar un entorno de Terraform a una nueva versión secundaria con aprovisionamiento autoadministrado.

1. Ejecute el siguiente comando para actualizar el entorno:

```
$ aws proton update-environment \
    --name "pr-environment" \
    --deployment-type "MINOR_VERSION" \
    --template-major-version "1" \
    --template-minor-version "1" \
```

```
    --provisioning-repository "branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec-mod.yaml"
```

Respuesta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:09:15.745000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "pr-env-template"
    }
}
```

2. Ejecute el siguiente comando para obtener y confirmar el estado:

```
$ aws proton get-environment \
    --name "pr-environment"
```

Respuesta:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
```

```
        "lastDeploymentSucceededAt": "2021-11-18T21:25:41.998000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "spec": "proton: EnvironmentSpec\nspec:\n   ssm_parameter_value: \"test
\"\n ssm_another_parameter_value: \"update\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "pr-env-template"
    }
}
```

3. Revise la solicitud de extracción enviada por AWS Proton.

   - Si aprueba la solicitud, el aprovisionamiento pasará a estar en curso.
   - Si rechaza la solicitud, se cancelará la creación del entorno.
   - Si se agota el tiempo de espera de la solicitud de extracción, la creación del entorno no se completará.

4. Proporcione el estado de aprovisionamiento a AWS Proton.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

Cancelación de la implementación de un entorno en curso

Puede intentar cancelar la implementación de la actualización de un entorno si el `deploymentStatus` está `IN_PROGRESS`. AWS Proton intentará cancelar la implementación. No se garantiza que la cancelación se realice correctamente.

Cuando se cancela la implementación de una actualización, AWS Proton intenta cancelar la implementación tal y como se indica en los pasos siguientes.

Con el aprovisionamiento administrado por AWS, AWS Proton hace lo siguiente:

- Establece el estado de la implementación en `CANCELLING`.
- Detiene la implementación en curso y elimina cualquier recurso nuevo que haya creado dicha implementación cuando está establecida en `IN_PROGRESS`.
- Establece el estado de la implementación en `CANCELLED`.
- Revierte el estado del recurso al que tenía antes de que se iniciara la implementación.

Con el aprovisionamiento autoadministrado, AWS Proton hace lo siguiente:

- Intenta cerrar la solicitud de extracción para evitar que se fusionen los cambios en el repositorio.
- Establece el estado de la implementación en `CANCELLED` si la solicitud de extracción se cerró correctamente.

Para obtener instrucciones sobre cómo cancelar la implementación de un entorno, consulte CancelEnvironmentDeployment en la Referencia de la API de AWS Proton.

Puede utilizar la consola o la CLI para cancelar los entornos que estén en curso.

AWS Management Console

Utilice la consola para cancelar la implementación de una actualización del entorno, tal y como se muestra en los pasos siguientes.

1. En la consola de AWS Proton, seleccione Entornos en el panel de navegación.
2. En la lista de entornos, elija el nombre del entorno con la actualización de implementación que desee cancelar.
3. Si el estado de la implementación de la actualización está establecido en En curso, en la página de detalles del entorno, seleccione Acciones y, a continuación, Cancelar implementación.
4. Se le pedirá que confirme que desea cancelar la implementación. Seleccione Cancelar implementación.
5. El estado de la implementación de la actualización se establece en Cancelando y, a continuación, en Cancelado para completar la cancelación.

AWS CLI

Utilice la AWS Proton AWS CLI para cancelar la implementación de una actualización del entorno en estado IN_PROGRESS a una nueva versión secundaria 2.

En la plantilla utilizada en este ejemplo se incluye una condición de espera para que la cancelación comience antes de que la implementación de la actualización se complete correctamente.

Ejecute el siguiente comando para cancelar la actualización:

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

Respuesta:

```
{
    "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"
    }
}
```

Ejecute el siguiente comando para obtener y confirmar el estado:

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

Respuesta:

```
{
```

```
    "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"
    }
}
```

Eliminación de un entorno

Puede eliminar un entorno de AWS Proton mediante la consola de AWS Proton o la AWS CLI.

> **Note**
>
> No puede eliminar un entorno que tenga algún componente asociado. Para eliminar un entorno de este tipo, primero debe eliminar todos los componentes que se ejecuten en el entorno. Para obtener más información sobre los componentes, consulte Componentes.

AWS Management Console

Elimine un entorno mediante la consola tal y como se describe en las dos opciones siguientes.

En la lista de entornos.

1. En la consola de AWS Proton, elija Entornos.
2. En la lista de entornos, seleccione el botón de radio a la izquierda del entorno que desee eliminar.
3. Elija Acciones y, a continuación, elija Eliminar.
4. Un modal le pedirá que confirme la acción de eliminación.

5. Siga las instrucciones y seleccione Sí, eliminar.

En la página de detalles del entorno.

1. En la consola de AWS Proton, elija Entornos.
2. En la lista de entornos, elija el nombre del entorno que desee eliminar.
3. En la página de detalles del entorno, elija Acciones y, a continuación, Eliminar.
4. Un modal le pedirá que confirme que desea continuar con la eliminación.
5. Siga las instrucciones y seleccione Sí, eliminar.

AWS CLI

Utilice la AWS CLI para eliminar un entorno.

No elimine un entorno si los servicios o las instancias de servicio están implementados en el entorno.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "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"
    }
}
```

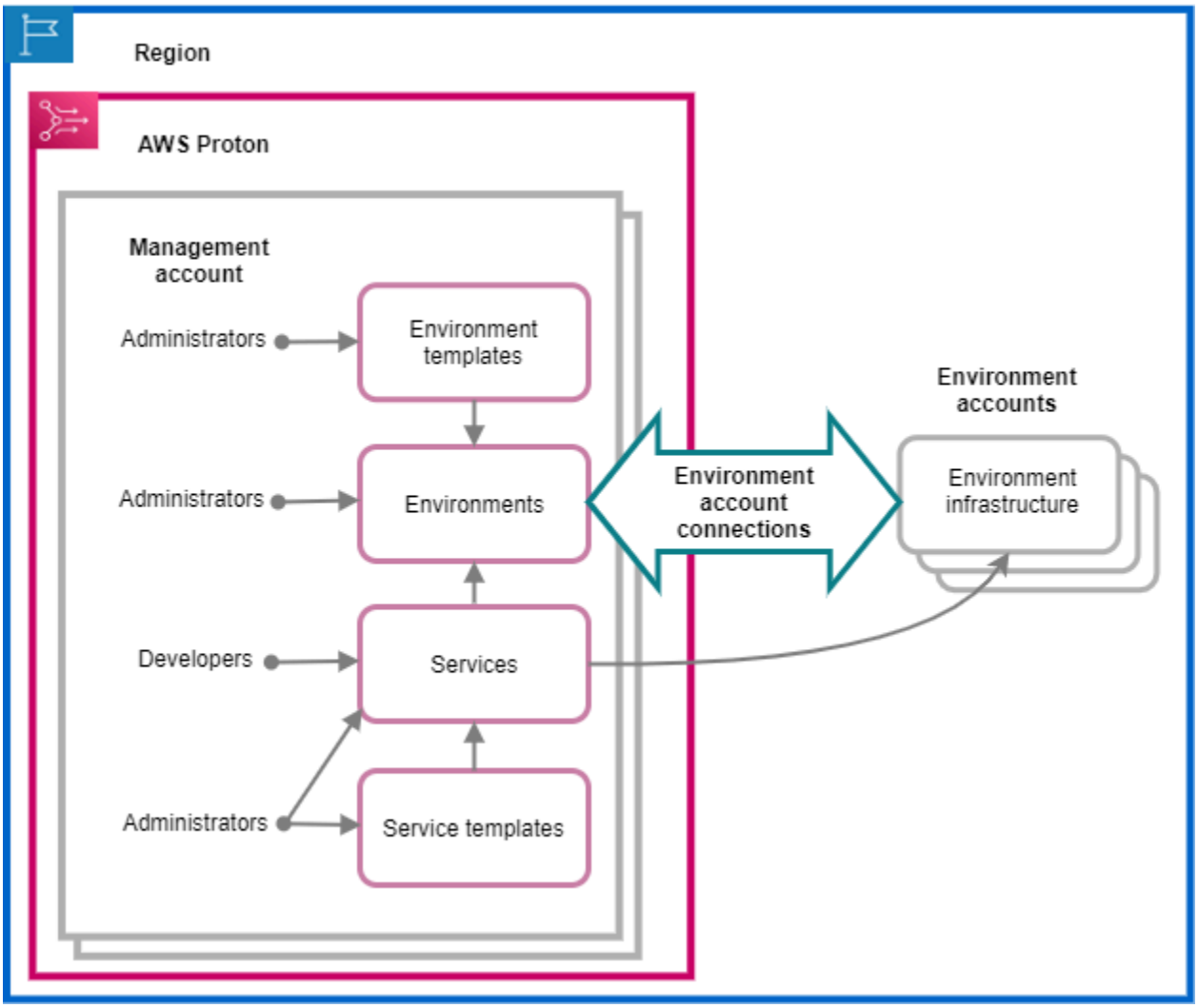
Conexiones de cuentas de entorno

Información general

Obtenga información sobre cómo crear y administrar un entorno de AWS Proton en una cuenta y aprovisionar sus recursos de infraestructura en otra cuenta. Esto puede ayudar a mejorar la visibilidad y la eficiencia a gran escala. Las conexiones de cuentas de entorno solo admiten el aprovisionamiento estándar con la infraestructura como código (IaC) de AWS CloudFormation.

> **Note**
>
> La información de este tema es relevante para los entornos que estén configurados con aprovisionamiento administrado por AWS. En el caso de los entornos configurados con aprovisionamiento autoadministrado, AWS Proton no aprovisiona directamente la infraestructura. En su lugar, envía solicitudes de extracción (PR) al repositorio del usuario para llevar a cabo el aprovisionamiento. Es responsabilidad del usuario asegurarse de que su código de automatización asuma la identidad y el rol correctos.
> Para obtener más información acerca de los métodos de aprovisionamiento, consulte the section called “Métodos de aprovisionamiento”.

Terminología



Con las conexiones de cuentas de entorno de AWS Proton, se puede crear un entorno de AWS Proton a partir de una cuenta y aprovisionar su infraestructura en otra cuenta.

Cuenta de administración

La única cuenta en la que el usuario, como administrador, crea un entorno de AWS Proton que aprovisiona recursos de infraestructura en otra cuenta de entorno.

Cuenta de entorno

Una cuenta en la que se aprovisiona la infraestructura del entorno cuando se crea un entorno de AWS Proton en otra cuenta.

Conexión de cuenta de entorno

> Una conexión bidireccional segura entre una cuenta de administración y una cuenta de entorno. Mantiene la autorización y los permisos como se describe más adelante en las siguientes secciones.

Al crear una conexión de cuenta de entorno en una cuenta de entorno de una región específica, solo las cuentas de administración de la misma región pueden ver y utilizar la conexión de cuenta de entorno. Esto significa que el entorno de AWS Proton creado en la cuenta de administración y la infraestructura de entorno aprovisionada en la cuenta de entorno deben estar en la misma región.

Consideraciones sobre la conexión de cuenta de entorno

- El usuario necesita una conexión de cuenta de entorno para cada entorno que desee aprovisionar en una cuenta de entorno.
- Para obtener información sobre las cuotas de conexión de cuentas de entorno, consulte Cuotas de AWS Proton.

Etiquetado

En la cuenta de entorno, utilice la consola o la AWS CLI para ver y administrar las etiquetas de conexión de cuenta de entorno administradas por el cliente. Las etiquetas administradas por AWS no se generan para las conexiones de cuentas de entorno. Para obtener más información, consulte Etiquetado.

Creación de un entorno en una cuenta y aprovisionamiento de su infraestructura en otra cuenta

Para crear y aprovisionar un entorno desde una sola cuenta de administración, configure una cuenta de entorno para el entorno que vaya a crear.

Comience en la cuenta de entorno y cree la conexión.

En la cuenta de entorno, cree un rol de servicio de AWS Proton que se limite únicamente a los permisos necesarios para aprovisionar los recursos de infraestructura del entorno. Para obtener más información, consulte AWS Proton rol de servicio para el aprovisionamiento mediante AWS CloudFormation.

A continuación, cree y envíe una solicitud de conexión de una cuenta de entorno a su cuenta de administración. Cuando se acepta la solicitud, AWS Proton puede utilizar el rol de IAM asociado que permite el aprovisionamiento de recursos del entorno en la cuenta de entorno asociada.

En la cuenta de administración, acepte o rechace la conexión de la cuenta de entorno.

En la cuenta de administración, acepte o rechace la solicitud de conexión de cuenta de entorno. No se puede eliminar la conexión de una cuenta de entorno de la cuenta de administración.

Si acepta la solicitud, AWS Proton puede utilizar el rol de IAM asociado que permite el aprovisionamiento de recursos en la cuenta de entorno asociada.

Los recursos de la infraestructura del entorno se aprovisionan en la cuenta de entorno asociada. Solo se pueden utilizar las API de AWS Proton para acceder y administrar tanto el entorno como sus recursos de infraestructura desde la cuenta de administración. Para obtener más información, consulte Creación de un entorno en una cuenta y aprovisionamiento en otra cuenta y Actualización de un entorno.

Tras rechazar una solicitud, no se podrá aceptar ni utilizar la conexión de cuenta de entorno rechazada.

> **Note**
>
> No se puede rechazar una conexión de cuenta de entorno que esté conectada a un entorno. Para rechazar la conexión de cuenta de entorno, primero hay que eliminar el entorno asociado.

En la cuenta de entorno, acceda a los recursos de infraestructura aprovisionados.

En la cuenta de entorno, puede ver los recursos de infraestructura aprovisionados y acceder a ellos. Por ejemplo, puede utilizar las acciones de la API de CloudFormation para monitorizar y limpiar las pilas si es necesario. No puede utilizar las acciones de la API de AWS Proton para acceder o administrar el entorno de AWS Proton que se utilizó para aprovisionar los recursos de infraestructura.

En la cuenta de entorno, puede eliminar las conexiones de cuenta de entorno que haya creado en la cuenta de entorno. No puede aceptarlas ni rechazarlas. Si elimina una conexión de cuenta de entorno que esté utilizando un entorno de AWS Proton, AWS Proton no podrá administrar los recursos de infraestructura del entorno hasta que se acepte una nueva conexión de entorno para

la cuenta de entorno y el entorno designado. El usuario es responsable de limpiar los recursos aprovisionados que permanezcan sin ninguna conexión al entorno.

Uso de la consola o de la CLI para administrar las conexiones de cuentas de entorno

Puede utilizar la CLI para crear una conexión de cuenta de entorno

AWS Management Console

Utilice la consola para crear una conexión de cuenta de entorno y enviar una solicitud a la cuenta de administración, tal y como se muestra en los pasos siguientes.

1. Elija un nombre para el entorno que planea crear en su cuenta de administración o elija el nombre de un entorno existente que requiera una conexión a una cuenta de entorno.
2. En una cuenta de entorno, en la consola de AWS Proton, seleccione Conexiones de cuentas de entorno en el panel de navegación.
3. En la página Conexiones de cuentas de entorno, seleccione Solicitud de conexión.

> **Note**
>
> Verifique que el ID de cuenta aparezca en el encabezado de la página Conexión de cuenta de entorno. Asegúrese de que coincida con el ID de cuenta de la cuenta de entorno en la que desee aprovisionar el entorno designado.

4. En la página Solicitud de conexión:
   a. En la sección Conectarse a una cuenta de administración, introduzca el ID de la cuenta de administración y el Nombre del entorno que introdujo en el paso 1.
   b. En la sección Rol de entorno, elija Nuevo rol de servicio y AWS Proton creará automáticamente un nuevo rol. O bien, seleccione el Rol de servicio existente y el nombre del rol de servicio que creó anteriormente.

> **Note**
>
> El rol que AWS Proton crea automáticamente para el usuario tiene amplios permisos. Le recomendamos que limite el rol a los permisos necesarios para aprovisionar los recursos de infraestructura del entorno. Para obtener más

información, consulte AWS Proton rol de servicio para el aprovisionamiento mediante AWS CloudFormation.

c. (Opcional) En la sección Etiquetas, elija Añadir nueva etiqueta para crear una etiqueta administrada por el cliente para la conexión de la cuenta de entorno.

d. Seleccione Solicitud de conexión.

5. La solicitud aparecerá como pendiente en la tabla de conexiones del entorno enviadas a una cuenta de administración y un modal indicará al usuario cómo aceptar la solicitud de la cuenta de administración.

Acepte o rechace la solicitud de conexión de cuenta de entorno.

1. En una cuenta de administración, en la consola de AWS Proton, seleccione Conexiones de cuentas de entorno en el panel de navegación.
2. En la página Conexiones de cuentas de entorno, en la tabla Solicitudes de conexión de cuentas de entorno, seleccione la solicitud de conexión de entorno que desee aceptar o rechazar.

> **Note**
>
> Verifique que el ID de cuenta aparezca en el encabezado de la página Conexión de cuenta de entorno. Asegúrese de que coincida con el ID de cuenta de la cuenta de administración asociada a la conexión de la cuenta de entorno que se va a rechazar. Tras rechazar esta conexión de cuenta de entorno, no podrá aceptar ni utilizar la conexión de cuenta de entorno rechazada.

3. Elija Rechazar o Aceptar.
   - Si ha seleccionado Rechazar, el estado cambiará de pendiente a rechazado.
   - Si ha seleccionado Aceptar, el estado cambiará de pendiente a conectado.

Elimine una conexión de cuenta de entorno.

1. En una cuenta de entorno, en la consola de AWS Proton, seleccione Conexiones de cuentas de entorno en el panel de navegación.

> **Note**
>
> Verifique que el ID de cuenta aparezca en el encabezado de la página Conexión de cuenta de entorno. Asegúrese de que coincida con el ID de cuenta de la cuenta de administración asociada a la conexión de la cuenta de entorno que se va a rechazar. Después de eliminar la conexión de esta cuenta de entorno, AWS Proton no podrá administrar los recursos de infraestructura del entorno en la cuenta de entorno. Solo se podrá administrar después de que la cuenta de administración acepte una nueva conexión de cuenta de entorno para la cuenta de entorno y el entorno designado.

2. En la página Conexiones de cuentas de entorno, en la sección Solicitudes enviadas para conectarse a la cuenta de administración, seleccione Eliminar.
3. Un modal le pedirá que confirme que desea continuar con la eliminación. Elija Eliminar.

AWS CLI

Elija un nombre para el entorno que planea crear en su cuenta de administración o elija el nombre de un entorno existente que requiera una conexión a una cuenta de entorno.

Cree una conexión de cuenta de entorno en una cuenta de entorno.

Ejecute el siguiente comando:

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role" \
    --management-account-id "111111111111"
```

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
```

```
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

Acepte o rechace la conexión de una cuenta de entorno en una cuenta de administración, tal y como se muestra en el siguiente comando y respuesta.

> **Note**
>
> Si rechaza esta conexión de cuenta de entorno, no podrá aceptar ni utilizar la conexión de cuenta de entorno rechazada.

Si especifica Rechazar, el estado cambiará de pendiente a rechazado.

Si especifica Aceptar, el estado cambiará de pendiente a conectado.

Ejecute el siguiente comando para aceptar la conexión de cuenta de entorno:

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
```

```
}
```

Ejecute el siguiente comando para rechazar la conexión de cuenta de entorno:

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-reject",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role"
    }
}
```

Vea las conexiones de una cuenta de entorno. Puede obtener o enumerar las conexiones de cuentas de entorno.

Ejecute el siguiente comando “get”:

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Elimine una conexión de cuenta de entorno en una cuenta de entorno.

> **Note**
>
> Si elimina esta conexión de cuenta de entorno, AWS Proton no podrá administrar los recursos de infraestructura del entorno en la cuenta de entorno hasta que se haya aceptado una nueva conexión de entorno para la cuenta de entorno y el entorno designado. El usuario es responsable de limpiar los recursos aprovisionados que permanezcan sin ninguna conexión al entorno.

Ejecute el siguiente comando:

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

Respuesta:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
```

```
}
```

Entornos administrados por el cliente

Con los entornos administrados por el cliente, puede utilizar la infraestructura existente, como una VPC, que ya haya implementado como entorno de AWS Proton. Al utilizar entornos administrados por el cliente, puede aprovisionar sus propios recursos compartidos fuera de AWS Proton. Sin embargo, aún puede permitir que AWS Proton consuma las salidas de aprovisionamiento relevantes como entradas para servicios de AWS Proton cuando se implementen. Si las salidas pueden cambiar, AWS Proton puede aceptar actualizaciones. Sin embargo, AWS Proton no puede cambiar el entorno directamente, ya que el aprovisionamiento se administra fuera de AWS Proton.

Una vez creado el entorno, el usuario es responsable de proporcionar las mismas salidas a AWS Proton que se habrían creado si AWS Proton hubiera creado el entorno, como los nombres de los clústeres de Amazon ECS o los ID de las VPC de Amazon.

Con esta funcionalidad, puede implementar y actualizar los recursos de servicio de AWS Proton desde una plantilla de servicio de AWS Proton en este entorno. Sin embargo, el entorno en sí no se modifica mediante las actualizaciones de la plantilla en AWS Proton. El usuario es responsable de ejecutar las actualizaciones en el entorno y de actualizar esas salidas en AWS Proton.

Puede tener varios entornos en una sola cuenta que sean una combinación de entornos administrados por AWS Proton y administrados por el cliente. También puede vincular una segunda cuenta y utilizar una plantilla de AWS Proton en la cuenta principal para ejecutar las implementaciones y actualizaciones de los entornos y servicios de esa segunda cuenta vinculada.

Cómo utilizar los entornos administrados por el cliente

Lo primero que deben hacer los administradores es registrar una plantilla de entorno importada y administrada por el cliente. No incluya manifiestos ni archivos de infraestructura en el paquete de plantillas. Proporcione únicamente el esquema.

El siguiente esquema describe una lista de salidas que utilizan el formato de API abierta y replica las salidas desde una plantilla de AWS CloudFormation.

> **Important**
>
> Solo se permiten entradas de cadena para las salidas.

El siguiente ejemplo es un fragmento de las secciones de salida de una plantilla de AWS CloudFormation para una plantilla de Fargate correspondiente.

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

El esquema del entorno de AWS Proton importado correspondiente es similar al siguiente. No proporciones valores predeterminados en el esquema.

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentOutput"
  types:
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:
          type: string
          description: "The ARN of the ECS role"
        VpcId:
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

En el momento de registrar la plantilla, el usuario indica que la plantilla es importada y proporciona la ubicación del bucket de Amazon S3 para el paquete. AWS Proton valida que el esquema solo contenga `environment_input_type` y no parámetros de plantillas de AWS CloudFormation antes de poner la plantilla en estado borrador.

Debe proporcionar lo siguiente para crear un entorno importado.

- Un rol de IAM para poder utilizarlo cuando se realicen implementaciones.
- Una especificación con los valores de las salidas requeridas.

Puede proporcionar ambas opciones mediante la consola o la AWS CLI con un proceso similar al de la implementación de un entorno normal.

Creación de roles de aprovisionamiento de CodeBuild

Las herramientas de infraestructura como código (IaC), como AWS CloudFormation y Terraform, requieren permisos para los diferentes tipos de recursos de AWS. Por ejemplo, si una plantilla de IaC declara un bucket de Amazon S3, necesitará permisos para crear, leer, actualizar y eliminar buckets de Amazon S3. Se considera una buena práctica de seguridad limitar los roles a los permisos mínimos necesarios. Dada la variedad de recursos de AWS, resulta difícil crear políticas de privilegios mínimos para las plantillas de IaC, especialmente cuando los recursos que administran esas plantillas pueden cambiar más adelante. Por ejemplo, en las últimas modificaciones de una plantilla administrada por AWS Proton, el usuario añade un recurso de base de datos de RDS.

La configuración de los permisos correctos ayuda a facilitar las implementaciones de la IaC de AWS Proton. El aprovisionamiento de CodeBuild ejecuta comandos de la CLI arbitrarios que proporciona el cliente en un proyecto de CodeBuild ubicado en la cuenta de cliente. Por lo general, estos comandos crean y eliminan la infraestructura mediante una herramienta de infraestructura como código (IaC), como AWS CDK. Cuando se implementa un recurso de AWS cuya plantilla utiliza el aprovisionamiento de CodeBuild, AWS iniciará una compilación en un proyecto de CodeBuild administrado por AWS. Se pasa un rol a CodeBuild, que CodeBuild asumirá para ejecutar comandos. Este rol, denominado Rol de aprovisionamiento de CodeBuild, lo proporciona el cliente y contiene los permisos necesarios para aprovisionar la infraestructura. Dicho rol está destinado a que solo lo asuma CodeBuild y ni siquiera AWS Proton puede asumirlo.

Creación del rol

El Rol de aprovisionamiento de CodeBuild se puede crear en la consola de IAM o en la AWS CLI. Para crearlo en la AWS CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AWSProtonCodeBuildProvisioningBasicAccess
```

Esto también asocia el `AWSProtonCodeBuildProvisioningBasicAccess`, que contiene los permisos mínimos que necesita el servicio de CodeBuild para ejecutar una compilación.

Si prefiere utilizar la consola, asegúrese de lo siguiente al crear el rol:

1. Para la entidad de confianza, seleccione el servicio de AWS y, a continuación, seleccione CodeBuild.
2. En el paso Añadir permisos, seleccione `AWSProtonCodeBuildProvisioningBasicAccess` y cualquier otra política que desee asociar.

Acceso de administrador

Si asocia la política `AdministratorAccess` al Rol de aprovisionamiento de CodeBuild, se garantizará que en ninguna plantilla de IaC se produzcan errores por falta de permisos. También significa que cualquier persona que pueda crear una plantilla de entorno o una plantilla de servicio pueda realizar acciones a nivel de administrador, incluso si ese usuario no es administrador. AWS Proton no recomienda utilizar `AdministatorAccess` con el Rol de aprovisionamiento de CodeBuild. Si decide utilizar `AdministratorAccess` con el Rol de aprovisionamiento de CodeBuild, hágalo en un entorno aislado.

Para crear un rol con `AdministratorAccess`, puede utilizar la consola de IAM o bien ejecutar este comando:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

Creación de un rol con un alcance mínimo

Si desea crear un rol con permisos mínimos, hay disponibles varios enfoques:

- Implemente el rol con permisos de administrador y, a continuación, limite los permisos del rol. Recomendamos utilizar el Analizador de acceso de IAM.
- Utilice políticas administradas para dar acceso a los servicios que planee utilizar.

AWS CDK

Si utiliza el AWS CDK con AWS Proton y has ejecutado `cdk bootstrap` en cada cuenta o región del entorno, entonces ya existe un rol para `cdk deploy`. En este caso, asocie la siguiente política al Rol de aprovisionamiento de CodeBuild:

```
{
    "Action": "sts:AssumeRole",
    "Resource": [
        "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
        "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
    ],
    "Effect": "Allow"
}
```

VPC personalizada

Si el usuario decide ejecutar CodeBuild en una VPC personalizada, necesitará los siguientes permisos en su rol de CodeBuild:

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:network-interface/*",
        "arn:aws:ec2:region:account-id:subnet/*",
        "arn:aws:ec2:region:account-id:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:*/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
```

```
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:AuthorizedService": "codebuild.amazonaws.com"
        }
    }
}
```

También puede utilizar la política administrada `AmazonEC2FullAccess`, aunque incluye permisos que quizás no necesite. Para asociar la política administrada mediante la CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

Servicios de AWS Proton

Un servicio de AWS Proton es una instancia de una plantilla de servicio, que normalmente incluye varias instancias de servicio y una canalización. Una instancia de servicio de AWS Proton es una instancia de una plantilla de servicio en un entorno específico. Una plantilla de servicio es una definición completa de la infraestructura y la canalización de servicios opcional para un servicio de AWS Proton.

Después de implementar las instancias de servicio, para actualizarlas puede insertar el código fuente que solicite la canalización de CI/CD o bien puede actualizar el servicio a las nuevas versiones de la plantilla de servicio. AWS Proton le avisa cuando hay nuevas versiones de la plantilla de servicio disponibles para que pueda actualizar los servicios a una nueva versión. Cuando se actualice el servicio, AWS Proton vuelve a implementar el servicio y las instancias de servicio.

En este capítulo se muestra cómo administrar los servicios mediante las operaciones de creación, visualización, actualización y eliminación. Para obtener más información, consulte la Referencia de la API del servicio de AWS Proton.

Temas

Crear un servicio

Para implementar una aplicación con AWS Proton, como desarrollador, debe crear un servicio y proporcionar las siguientes entradas.

1. El nombre de una plantilla de servicio de AWS Proton publicada por el equipo de la plataforma.
2. Un nombre para el servicio.

3. El número de instancias de servicio que desee implementar.
4. Una selección de entornos que desee utilizar.
5. Una conexión al repositorio de código si se utiliza una plantilla de servicio que incluya una canalización de servicios (opcional).

¿Qué hay en un servicio?

Cuando se crea un servicio de AWS Proton, se puede elegir entre dos tipos diferentes de plantillas de servicio:

- Una plantilla de servicio que incluye una canalización de servicios (predeterminada).
- Una plantilla de servicio que no incluye una canalización de servicios.

Debe crear al menos una instancia de servicio al crear el servicio.

Una instancia de servicio y una canalización opcional están asociadas a un servicio. Solo puede crear o eliminar una canalización en el contexto de las acciones de creación y eliminación de servicios. Para obtener información sobre cómo agregar y eliminar instancias de un servicio, consulte Edición de un servicio.

> **Note**
>
> El entorno del usuario está configurado para un aprovisionamiento autoadministrado o administrado por AWS. AWS Proton aprovisiona servicios en un entorno mediante el mismo método de aprovisionamiento que utiliza el entorno. El desarrollador que crea o actualiza las instancias de servicio no verá ninguna diferencia y su experiencia será la misma en ambos casos.
> Para obtener más información acerca de los métodos de aprovisionamiento, consulte the section called “Métodos de aprovisionamiento”.

Plantillas de servicio

Están disponibles las versiones principales y secundarias de las plantillas de servicio. Al utilizar la consola, se selecciona la versión principal y secundaria `Recommended` más reciente de la plantilla de servicio. Al utilizar la AWS CLI y especificar solo la versión principal de la plantilla de servicio, se especifica implícitamente su última versión secundaria `Recommended`.

A continuación se describe la diferencia entre las versiones principales y secundarias de la plantilla y su uso.

- Las nuevas versiones de una plantilla pasarán a ser `Recommended` en cuanto un miembro del equipo de la plataforma las aprueba. Esto significa que los nuevos servicios se crean con esa versión y se pide al usuario que actualice los servicios existentes a la nueva versión.
- A través de AWS Proton, el equipo de la plataforma puede actualizar automáticamente las instancias de servicio a una nueva versión secundaria de una plantilla de servicio. Las versiones secundarias deben ser compatibles con versiones anteriores.
- Como las versiones principales requieren que se proporcionen nuevas entradas como parte del proceso de actualización, el usuario debe actualizar el servicio a una versión principal de la plantilla de servicio. Las versiones principales no son compatibles con versiones anteriores.

Crear un servicio

Los siguientes procedimientos muestran cómo utilizar la consola de AWS Proton o la AWS CLI para crear un servicio con o sin una canalización de servicios.

AWS Management Console

Cree un servicio como se muestra en los siguientes pasos de la consola.

1. En la consola de AWS Proton, seleccione Configuración.
2. Elija Create service.
3. En la página Elegir una plantilla de servicio, seleccione una plantilla y elija Configurar.

   Si no quiere utilizar una canalización habilitada, elija una plantilla marcada con Excluye la canalización para el servicio.
4. En la página Configurar servicio, en la sección Configuración del servicio, introduzca un Nombre del servicio.
5. (Opcional) Escriba una descripción para el servicio.
6. En la sección Configuración del repositorio de servicios:

   a. Para Conexión de CodeStar, elija su conexión de la lista.

   b. En ID del repositorio, elija de la lista el nombre del repositorio que contenga el código fuente.

c. En Nombre de la ramificación, elija de la lista el nombre de la ramificación del repositorio de código fuente.

7. (Opcional) En la sección Etiquetas, seleccione Agregar etiqueta e introduzca una clave y un valor para crear una etiqueta administrada por el cliente.
8. Elija Siguiente.
9. En la página Configurar ajustes personalizados, en la sección Instancias de servicio, en la sección Nueva instancia. Debe introducir valores para los parámetros `required`. Puede introducir valores para los parámetros `optional` o bien utilizar los valores predeterminados si los hubiera.
10. En la sección de Entradas de canalización, debe introducir valores para los parámetros `required`. Puede introducir valores para los parámetros `optional` o bien utilizar los valores predeterminados si los hubiera.
11. Seleccione Siguiente y revise las entradas.
12. Seleccione Crear.

    Vea los detalles y el estado del servicio, así como las etiquetas administradas por AWS y las etiquetas administradas por el cliente para su servicio.
13. En el panel de navegación, elija Servicios.

    Una nueva página muestra una lista de los servicios del usuario junto con el estado y otros detalles de los servicios.

AWS CLI

Al utilizar la AWS CLI, el usuario debe especificar las entradas del servicio en un archivo de `spec` con formato YAML, `.aws-proton/service.yaml`, ubicado en el directorio de código fuente.

Puede utilizar el comando `get-service-template-minor-version` de la CLI para ver los parámetros obligatorios y opcionales del esquema para los que se proporcionan valores en el archivo de especificaciones.

Si desea utilizar una plantilla de servicio que tenga `pipelineProvisioning: "CUSTOMER_MANAGED"`, no incluya la sección `pipeline:` en la especificación ni incluya los parámetros `-repository-connection-arn`, `-repository-id` y `-branch-name` en el comando `create-service`.

Cree un servicio con una canalización de servicios, como se muestra en los siguientes pasos de la CLI.

1. Configure el rol de servicio para la canalización como se muestra en el siguiente comando de ejemplo de la CLI.

   Comando:

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

2. En la siguiente lista se muestra un ejemplo de especificación, basado en el esquema de la plantilla de servicio, que incluye la canalización de servicios y las entradas de la instancia.

   Especificación:

```
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"
```

   Cree un servicio con una canalización, como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

   Comando:

```
$ aws proton create-service \
        --name "MySimpleService" \
        --branch-name "mainline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --repository-connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
```

```
        --repository-id "myorg/myapp" \
        --spec "file://spec.yaml"
```

Respuesta:

```
{
    "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"
    }
}
```

Cree un servicio sin una canalización de servicios, como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

A continuación, se muestra un ejemplo de especificación que no incluye las entradas de la canalización de servicios.

Especificación:

```
proton: ServiceSpec

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

Para crear un servicio sin una canalización de servicios aprovisionada, debe proporcionar la ruta a un archivo **`spec.yaml`** y no incluir los parámetros del repositorio, como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

Comando:

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

Respuesta:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/
MySimpleServiceNoPipeline",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleServiceNoPipeline",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service-no-pipeline"
    }
}
```

Visualización de datos de servicio

Puede ver y enumerar los datos detallados del servicio mediante la consola de AWS Proton o la AWS CLI.

AWS Management Console

Enumere y consulte los detalles del servicio mediante la consola de AWS Proton, tal y como se muestra en los pasos siguientes.

1. Para ver una lista de los servicios, elija Servicios en el panel de navegación.
2. Para ver los datos detallados, elija el nombre de un servicio.

   Consulte los datos de detalle del servicio.

AWS CLI

Vea los detalles de un servicio con una canalización de servicios, como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

Comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Vea los detalles de un servicio sin una canalización de servicios, como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

Comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Edición de un servicio

Puede realizar las siguientes modificaciones en un servicio de AWS Proton.

- Edite la descripción del servicio.
- Para editar un servicio, agregue y quite las instancias de servicio.

Edición de la descripción del servicio

Puede utilizar la consola o la AWS CLI para editar la descripción de un servicio.

AWS Management Console

Edite un servicio mediante la consola tal y como se describe en los pasos siguientes.

En la lista de servicios.

1. En la consola de AWS Proton, seleccione Configuración.
2. En la lista de servicios, seleccione el botón de radio situado a la izquierda del servicio que desee actualizar.
3. Elija Editar.
4. En la página Configurar servicio, rellene el formulario y seleccione Siguiente.
5. En la página Configurar ajustes personalizados, seleccione Siguiente.
6. Revise sus cambios y seleccione Guardar cambios.

En la página de detalles del servicio.

1. En la consola de AWS Proton, seleccione Configuración.
2. En la lista de servicios, elija el nombre del servicio que desee editar.
3. En la página de detalles del servicio, seleccione Editar.
4. En la página Configurar servicio, rellene el formulario y seleccione Siguiente.
5. En la página Configurar ajustes personalizados, rellene el formulario y seleccione Siguiente.
6. Revise sus cambios y seleccione Guardar cambios.

AWS CLI

Edite una descripción como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

Comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Edite un servicio para agregar o eliminar instancias de servicio

En el caso de un servicio de AWS Proton, puede agregar o eliminar instancias de servicio; para ello, debe enviar una especificación editada. Para que una solicitud se realice correctamente, se deben cumplir las siguientes condiciones:

- Cuando se envía la solicitud de edición, el servicio y la canalización aún no se editan ni se eliminan.
- La especificación editada no incluye ni las ediciones que modifican a la canalización del servicio ni las ediciones de las instancias de servicio existentes que no sean para eliminarse.
- La especificación editada no elimina ninguna instancia de servicio existente que tenga un componente conectado. Para eliminar una instancia de servicio de este tipo, primero debe actualizar el componente para desconectarlo de su instancia de servicio. Para obtener más información sobre los componentes, consulte Componentes.

Las instancias en las que se ha producido un error de eliminación son instancias de servicio en el estado `DELETE_FAILED`. Cuando se solicita editar un servicio, AWS Proton intenta eliminar automáticamente las instancias en las que se haya producido un error de eliminación como parte del proceso de edición. Si alguna de las instancias de servicio no se pudo eliminar, es posible que aún haya recursos asociados a dichas instancias, aunque no estén visibles desde la consola o la AWS CLI. Compruebe los recursos de la infraestructura de las instancias que no se pudieron eliminar y límpialos para que AWS Proton pueda eliminarlos.

Para conocer la cuota de instancias de servicio de un servicio, consulte Cuotas de AWS Proton. También debe mantener al menos una instancia de servicio para el servicio después de crearlo. Durante el proceso de actualización, AWS Proton hace un recuento de las instancias de servicio

existentes y de las instancias que se van a añadir o eliminar. Las instancias que no se hayan podido eliminar se incluyen en este recuento y el usuario debe contabilizarlas al editar su `spec`.

Uso de la consola o la AWS CLI para agregar o eliminar instancias de servicio

AWS Management Console

Edite el servicio para agregar o eliminar instancias de servicio mediante la consola.

En la consola de AWS Proton

1. En el panel de navegación, elija Servicios.
2. Seleccione el servicio que desee editar.
3. Elija Editar.
4. (Opcional) En la página Configurar servicio, edite el nombre o la descripción del servicio y, a continuación, seleccione Siguiente.
5. En la página Configurar ajustes personalizados, elija Eliminar para eliminar una instancia de servicio y elija Agregar nueva instancia para agregar una instancia de servicio y, por último, complete el formulario.
6. Elija Siguiente.
7. Revise las actualizaciones y seleccione Guardar cambios.
8. Un modal le pedirá al usuario que verifique la eliminación de las instancias de servicio. Siga las instrucciones y seleccione Sí, eliminar.
9. En la página de detalles del servicio, consulte los detalles del estado de su servicio.

AWS CLI

Agregue y elimine instancias de servicio con una **`spec`** editada, como se muestra en el siguiente ejemplo de comandos y respuestas de la AWS CLI.

Al utilizar la CLI, `spec` debe excluir las instancias de servicio que desee eliminar e incluir tanto las instancias de servicio que desee agregar como las instancias de servicio existentes que no se hayan marcado para su eliminación.

En la siguiente lista se muestra el ejemplo de `spec` anterior a la edición y una lista de las instancias de servicio implementadas según la especificación. Esta especificación se utilizó en el ejemplo anterior para editar la descripción de un servicio.

Especificación:

```
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"
```

El siguiente ejemplo de comando y respuesta de la CLI, `list-service-instances`, muestra las instancias activas antes de agregar o eliminar una instancia de servicio.

Comando:

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

Respuesta:

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

En la siguiente lista se muestra el ejemplo editado que `spec` utilizó para eliminar y agregar una instancia. Se elimina la instancia existente nombrada como `my-instance` y se agrega una nueva instancia nombrada como `yet-another-instance`.

Especificación:

```
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"
```

Puede utilizar `"${Proton::CURRENT_VAL}"` para indicar qué valores de parámetros desea conservar de la `spec` original, si los valores existen en la `spec`. Utilice `get-service` para ver la `spec` original de un servicio, tal y como se describe en Visualización de datos de servicio.

La siguiente lista muestra cómo puede utilizar `"${Proton::CURRENT_VAL}"` para asegurarse de que `spec` no incluya cambios en los valores de los parámetros para que las instancias de servicios existentes permanezcan.

Especificación:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

La siguiente lista muestra el comando y la respuesta de la CLI para editar el servicio.

Comando:

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

Respuesta:

```
{
    "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"
    }
}
```

El siguiente comando y respuesta de `list-service-instances` confirman que la instancia existente nombrada como `my-instance` se elimina y se agrega una nueva instancia nombrada como `yet-another-instance`.

Comando:

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

Respuesta:

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


Qué ocurre cuando se agregan o se eliminan instancias de servicio

Tras enviar una edición de servicio para eliminar y agregar instancias de servicio, AWS Proton realiza las siguientes acciones.

- Establece el servicio en `UPDATE_IN_PROGRESS`.
- Si el servicio tiene una canalización, establece su estado en `IN_PROGRESS` y bloquea las acciones de la canalización.
- Establece las instancias de servicio que se van a eliminar como `DELETE_IN_PROGRESS`.
- Bloquea las acciones del servicio.
- Bloquea las acciones en instancias de servicio que estén marcadas para eliminarse.
- Crea nuevas instancias de servicio.
- Elimina las instancias que se hayan indicado para su eliminación.
- Intenta eliminar las instancias cuya eliminación haya producido errores.
- Una vez se agreguen o eliminen instancias por completo, vuelve a aprovisionar la canalización de servicios (si la hubiera), establece el servicio en `ACTIVE` y habilita las acciones del servicio y de la canalización.

AWS Proton intenta solucionar los modos de error de la siguiente manera.

- Si no se pudieron crear una o más instancias de servicio, AWS Proton intenta desaprovisionar todas las instancias de servicio recién creadas y devuelve la `spec` al estado anterior. No elimina ninguna instancia de servicio ni modifica la canalización de ninguna manera.

- Si no se pudieron eliminar una o más instancias de servicio, AWS Proton vuelve a aprovisionar la canalización sin las instancias eliminadas. La `spec` se actualiza para incluir las instancias agregadas y excluir las instancias que se marcaron para su eliminación.
- Si la canalización no se aprovisiona correctamente, no se intentará revertirla y tanto el servicio como la canalización reflejarán un estado de actualización con errores.

Modificaciones de etiquetado y servicio

Al añadir instancias de servicio como parte de la edición del servicio, las etiquetas administradas por AWS se propagan a las nuevas instancias y los recursos aprovisionados y se crean automáticamente para ellos. Si crea etiquetas nuevas, esas etiquetas solo se aplicarán a las nuevas instancias. Las etiquetas administradas por el cliente del servicio existente también se propagan a las nuevas instancias. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

Eliminación de un servicio

Se puede eliminar un servicio de AWS Proton, con sus instancias y canalización, mediante la consola de AWS Proton o la AWS CLI.

No se puede eliminar un servicio que tenga una instancia de servicio con un componente conectado. Para eliminar un servicio de este tipo, primero debe actualizar todos los componentes conectados para desconectarlos de sus instancias de servicio. Para obtener más información sobre los componentes, consulte Componentes.

AWS Management Console

Elimine un servicio mediante la consola tal y como se describe en los pasos siguientes.

En la página de detalles del servicio.

1. En la consola de AWS Proton, seleccione Configuración.
2. En la lista de servicios, elija el nombre del servicio que desee eliminar.
3. En la página de detalles del servicio, elija Acciones y, a continuación, Eliminar.
4. Un modal le pedirá que confirme la acción de eliminación.
5. Siga las instrucciones y seleccione Sí, eliminar.

AWS CLI

Elimine un servicio como se muestra en el siguiente ejemplo de comando y respuesta de la CLI.

Comando:

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

Respuesta:

```
{
    "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"
    }
}
```

Visualización de los datos de instancias de servicio

Obtenga información sobre cómo ver los datos detallados de las instancias de servicio de AWS Proton. Puede utilizar la consola o la AWS CLI.

Una instancia de servicio pertenece a un servicio. Solo puede crear o eliminar una instancia en el contexto de las acciones de edición, creación y eliminación de servicios. Para obtener información sobre cómo agregar y eliminar instancias de un servicio, consulte Edición de un servicio.

AWS Management Console

Enumere y consulte los detalles de la instancia de servicio mediante la consola de AWS Proton, tal y como se muestra en los pasos siguientes.

1. Para ver una lista de las instancias de servicio, seleccione Instancias de servicio en el panel de navegación.
2. Para ver los datos detallados, elija el nombre de una instancia de servicio.

   Vea los datos detallados de la instancia de servicio.

AWS CLI

Enumere y vea los detalles de la instancia de servicio como se muestra en los siguientes comandos y respuestas de ejemplo de la CLI.

Comando:

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

Respuesta:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
service-instance/instance-one",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentArn": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "name": "instance-one",
            "serviceName": "simple-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Comando:

```
$ aws proton get-service-instance \
    --name "instance-one" \
```

```
    --service-name "simple-svc"
```

Respuesta:

```
{
    "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"
    }
}
```

Actualización de una instancia de servicio

Obtenga información sobre cómo actualizar una instancia de servicio de AWS Proton y sobre cómo cancelar la actualización.

Una instancia de servicio pertenece a un servicio. Solo puede crear o eliminar una instancia en el contexto de las acciones de edición, creación y eliminación de servicios. Para obtener información sobre cómo agregar y eliminar instancias de un servicio, consulte Edición de un servicio.

Existen cuatro modos de actualizar una instancia de servicio, tal y como se describe en la siguiente lista. Al utilizar la AWS CLI, el campo `deployment-type` define el modo. Al utilizar la consola, estos modos se asignan a las acciones Editar y Actualizar a la última versión secundaria y Actualizar a la última versión principal, que aparecen en el menú Acciones de la página de detalles de la instancia de servicio.

`NONE`

En este modo, no se produce ninguna implementación. Solo se actualizan los parámetros de metadatos solicitados.

`CURRENT_VERSION`

En este modo, la instancia de servicio se implementa y actualiza con la nueva especificación que se proporcione. Solo se actualizan los parámetros solicitados. No incluya parámetros de versiones principales o secundarias cuando utilice este `deployment-type`.

`MINOR_VERSION`

En este modo, la instancia de servicio se implementa y actualiza con la versión secundaria recomendada (más reciente) publicada de la versión principal actual que se utiliza de forma predeterminada. También puede especificar una versión secundaria diferente de la versión principal que se está utilizando actualmente.

`MAJOR_VERSION`

En este modo, la instancia de servicio se implementa y actualiza con las versiones principal y secundaria recomendadas (más recientes) publicadas de la plantilla actual que se utiliza de forma predeterminada. También puede especificar una versión principal diferente que sea superior a la versión principal en uso y una versión secundaria (opcional).

Puede intentar cancelar la implementación de la actualización de una instancia de servicio si el `deploymentStatus` está `IN_PROGRESS`. AWS Proton intentará cancelar la implementación. No se garantiza que la cancelación se realice correctamente.

Cuando se cancela la implementación de una actualización, AWS Proton intenta cancelar la implementación tal y como se indica en los pasos siguientes.

- Establece el estado de la implementación en `CANCELLING`.
- Detiene la implementación en curso y elimina cualquier recurso nuevo que haya creado dicha implementación cuando está `IN_PROGRESS`.
- Establece el estado de la implementación en `CANCELLED`.
- Revierte el estado del recurso al que tenía antes de que se iniciara la implementación.

Para obtener más información sobre cómo cancelar la implementación de una instancia de servicio, consulte CancelServiceInstanceDeployment en la Referencia de la API de AWS Proton.

Utilice la consola o la AWS CLI para realizar actualizaciones o cancelar las implementaciones de actualizaciones.

AWS Management Console

Actualice una instancia de servicio mediante la consola siguiendo estos pasos.

1. En la consola de AWS Proton, seleccione Instancias de servicio en el panel de navegación.
2. En la lista de instancias de servicio, elija el nombre de la instancia de servicio que desee actualizar.
3. Seleccione Acciones y, a continuación, elija una de las opciones de actualización, Editar para actualizar las especificaciones o Acciones y, a continuación, Actualizar a la última versión secundaria o Actualizar a la última versión principal.
4. Rellene cada formulario y seleccione Siguiente hasta llegar a la página de Revisión.
5. Revisa las modificaciones y seleccione Actualizar.

AWS CLI

Actualice una instancia de servicio a una nueva versión secundaria, como se muestra en los comandos y respuestas de ejemplo de la CLI.

Al actualizar la instancia de servicio con una `spec` modificada, puede utilizar `"${Proton::CURRENT_VAL}"` para indicar qué valores de parámetros desea conservar de la `spec` original, siempre y cuando dichos valores existan en la `spec`. Utilice `get-service` para ver la `spec` original de una instancia de servicio, tal y como se describe en Visualización de datos de servicio.

El siguiente ejemplo muestra cómo puede utilizar `"${Proton::CURRENT_VAL}"` en una `spec`.

Especificación:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"
```

```
instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Comando: actualizar

```
$ 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"
```

Respuesta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentName": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Comando: obtener y confirmar el estado

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

Respuesta:

```
{
    "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"
    }
}
```

AWS Management Console

Cancele la implementación de una instancia de servicio mediante la consola, tal y como se muestra en los pasos siguientes.

1. En la consola de AWS Proton, seleccione Instancias de servicio en el panel de navegación.
2. En la lista de instancias de servicio, elija el nombre de la instancia de servicio con la actualización de implementación que desee cancelar.
3. Si el estado de la implementación de la actualización está En curso, en la página de detalles de la instancia de servicio, seleccione Acciones y, a continuación, Cancelar implementación.

4. Un modal le pedirá que confirme la cancelación. Seleccione Cancelar implementación.
5. El estado de la implementación de la actualización se establece en Cancelando y, a continuación, en Cancelado para completar la cancelación.

AWS CLI

Cancele la actualización de implementación de una instancia de servicio con estado IN_PROGRESS a la nueva versión secundaria 2, como se muestra en los siguientes comandos y respuestas de ejemplo de la CLI.

En la plantilla utilizada en este ejemplo se incluye una condición de espera para que la cancelación comience antes de que la implementación de la actualización se complete correctamente.

Comando: cancelar

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

Respuesta:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: abc\n  my_sample_pipeline_required_input:
 '123'\ninstances:\n- name: my-instance\n  environment: MySimpleEnv
\n  spec:\n    my_sample_service_instance_optional_input: def\n
 my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
 environment: MySimpleEnv\n  spec:\n    my_sample_service_instance_required_input:
 '789'\n",
```

```
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Comando: obtener y confirmar el estado

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

Respuesta:

```
{
    "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"
    }
}
```

Actualización de una canalización de servicios

Obtenga información sobre cómo actualizar una canalización de servicios de AWS Proton y sobre cómo cancelar la actualización.

Una canalización de servicios pertenece a un servicio. Solo puede crear o eliminar una canalización en el contexto de las acciones de creación y eliminación de servicios.

Existen cuatro modos de actualizar una canalización de servicios, tal y como se describe en la siguiente lista. Al utilizar la AWS CLI, el campo `deployment-type` define el modo. Cuando se utiliza la consola, estos modos se asignan a las opciones Editar canalización y Actualizar a la versión recomendada.

`NONE`

En este modo, no se produce ninguna implementación. Solo se actualizan los parámetros de metadatos solicitados.

`CURRENT_VERSION`

En este modo, la canalización de servicio se implementa y actualiza con la nueva especificación que se proporcione. Solo se actualizan los parámetros solicitados. No incluya parámetros de versiones principales o secundarias cuando utilice este `deployment-type`.

`MINOR_VERSION`

En este modo, la canalización de servicio se implementa y actualiza con la versión secundaria recomendada (más reciente) publicada de la versión principal actual que se utiliza de forma predeterminada. También puede especificar una versión secundaria diferente de la versión principal que se está utilizando actualmente.

`MAJOR_VERSION`

En este modo, la canalización de servicios se implementa y actualiza con la versión principal y secundaria publicada y recomendada (más reciente) de la plantilla actual de forma predeterminada. También puede especificar una versión principal diferente que sea superior a la versión principal en uso y una versión secundaria (opcional).

Puede intentar cancelar la implementación de la actualización de una canalización de servicio si el `deploymentStatus` está `IN_PROGRESS`. AWS Proton intentará cancelar la implementación. No se garantiza que la cancelación se realice correctamente.

Cuando se cancela la implementación de una actualización, AWS Proton intenta cancelar la implementación tal y como se indica en los pasos siguientes.

- Establece el estado de la implementación en `CANCELLING`.
- Detiene la implementación en curso y elimina cualquier recurso nuevo que haya creado dicha implementación cuando está `IN_PROGRESS`.
- Establece el estado de la implementación en `CANCELLED`.
- Revierte el estado del recurso al que tenía antes de que se iniciara la implementación.

Para obtener más información sobre cómo cancelar la implementación de una canalización de servicio, consulte CancelServicePipelineDeployment en la Referencia de la API de AWS Proton.

Utilice la consola o la AWS CLI para realizar actualizaciones o cancelar las implementaciones de actualizaciones.

AWS Management Console

Actualice una canalización de servicio mediante la consola tal y como se describe en los pasos siguientes.

1. En la consola de AWS Proton, seleccione Configuración.
2. En la lista de servicios, elija el nombre del servicio cuya canalización desee actualizar.
3. Hay dos pestañas en la página de detalles del servicio: Información general y Canalización. Elija Canalización.
4. Si desea actualizar las especificaciones, seleccione Editar canalización, rellene cada formulario y seleccione Siguiente hasta completar el formulario final y, a continuación, seleccione Actualizar canalización.

   Si desea actualizar a una nueva versión y hay un Icono de información que indica que hay una nueva versión disponible en la Plantilla de canalización, elija el nombre de la nueva versión de la plantilla.

   a. Seleccione Actualizar a la versión recomendada.

b. Rellene cada formulario y elija Siguiente hasta que complete el formulario final y seleccione Actualizar.

AWS CLI

Actualice una canalización de servicio a una nueva versión secundaria, como se muestra en los siguientes comandos y respuestas de ejemplo de la CLI.

Al actualizar la canalización de servicio con una `spec` modificada, puede utilizar `"${Proton::CURRENT_VAL}"` para indicar qué valores de parámetros desea conservar de la `spec` original, siempre y cuando dichos valores existan en la `spec`. Utilice `get-service` para ver la `spec` original de una canalización de servicio, tal y como se describe en Visualización de datos de servicio.

El siguiente ejemplo muestra cómo puede utilizar `"${Proton::CURRENT_VAL}"` en una `spec`.

Especificación:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Comando: actualizar

```
$ 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"
```

Respuesta:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"my-instance\"\n    environment: \"MySimpleEnv
\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"MySimpleEnv\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Comando: obtener y confirmar el estado

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

Respuesta:

```
{
    "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"
    }
}
```

AWS Management Console

Cancele la implementación de una canalización de servicio mediante la consola, tal y como se muestra en los pasos siguientes.

1. En la consola de AWS Proton, elija Servicios en el panel de navegación.
2. En la lista de servicios, elija el nombre del servicio que contenga la canalización con la actualización de implementación que desee cancelar.
3. En la página de detalles del servicio, elija la pestaña Canalización.

4. Si el estado de la implementación de la actualización está En curso, en la página de detalles de la canalización de servicios, seleccione Cancelar implementación.
5. Un modal le pedirá que confirme la cancelación. Seleccione Cancelar implementación.
6. El estado de la implementación de la actualización se establece en Cancelando y, a continuación, en Cancelado para completar la cancelación.

AWS CLI

Cancele la actualización de implementación de una canalización de servicio con estado IN_PROGRESS a la versión secundaria 2, como se muestra en los siguientes comandos y respuestas de ejemplo de la CLI.

En la plantilla utilizada en este ejemplo se incluye una condición de espera para que la cancelación comience antes de que la implementación de la actualización se complete correctamente.

Comando: cancelar

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

Respuesta:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Comando: obtener y confirmar el estado

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

Respuesta:


```
{
    "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"
    }
}
```

Componentes de AWS Proton

Los componentes son un tipo de recurso de AWS Proton. Añaden flexibilidad a las plantillas de servicio. Los componentes proporcionan a los equipos de plataformas un mecanismo para ampliar los patrones de infraestructura principales y definir medidas de seguridad que permiten a los desarrolladores administrar aspectos de su infraestructura de aplicaciones.

En AWS Proton, los administradores definen la infraestructura estándar que se utiliza en todos los equipos de desarrollo y en las aplicaciones. Sin embargo, es posible que los equipos de desarrollo necesiten incluir recursos adicionales para sus casos de uso específicos, como las colas de Amazon Simple Queue Service (Amazon SQS) o las tablas de Amazon DynamoDB. Estos recursos específicos de la aplicación pueden cambiar con frecuencia, especialmente durante las primeras etapas del desarrollo de una aplicación. Mantener estos cambios frecuentes en las plantillas creadas por los administradores puede resultar difícil de administrar y escalar; los administradores tendrían que mantener muchas más plantillas sin un verdadero valor añadido para el administrador. La alternativa (permitir a los desarrolladores de aplicaciones crear plantillas para sus aplicaciones) tampoco es la ideal, ya que reduce la capacidad de los administradores de estandarizar los principales componentes de la arquitectura, como las tareas de AWS Fargate. Aquí es donde entran en juego los componentes.

Con un componente, un desarrollador puede agregar recursos adicionales a su aplicación, más allá de lo que los administradores definieron en las plantillas de entorno y servicio. A continuación, el desarrollador adjunta el componente a una instancia de servicio. AWS Proton aprovisiona los recursos de infraestructura definidos por el componente del mismo modo que aprovisiona los recursos para los entornos y las instancias de servicio.

Un componente puede leer las entradas de la instancia de servicio y proporcionar salidas a la instancia de servicio, para una experiencia totalmente integrada. Por ejemplo, si el componente agrega un bucket de Amazon Simple Storage Service (Amazon S3) para que lo utilice una instancia de servicio, la plantilla del componente puede tener en cuenta los nombres de las instancias de servicio y de entorno para darle un nombre al bucket. Cuando AWS Proton representa la plantilla de servicio para aprovisionar una instancia de servicio, la instancia de servicio puede hacer referencia al bucket y utilizarlo.

Los componentes que actualmente admite AWS Proton son los componentes definidos directamente. El archivo de infraestructura como código (IaC) que define la infraestructura del componente se transfiere directamente a la API de AWS Proton o a la consola. Esto es diferente a un entorno o

servicio, en el que se define la IaC en un paquete de plantillas y se registra el paquete como un recurso de plantilla y, a continuación, se utiliza un recurso de plantilla para crear el entorno o el servicio.

> **Note**
>
> Los componentes definidos directamente permiten a los desarrolladores definir una infraestructura adicional y aprovisionarla. AWS Proton aprovisiona todos los componentes definidos directamente que se ejecutan en el mismo entorno y utilizan el mismo rol de AWS Identity and Access Management (IAM).

Un administrador puede controlar lo que los desarrolladores pueden hacer con los componentes de dos maneras:

- Orígenes de componentes compatibles: un administrador puede permitir que se asocien componentes a las instancias de servicio en función de una propiedad de las versiones de las plantillas de servicio de AWS Proton. De forma predeterminada, los desarrolladores no pueden conectar componentes a las instancias de servicio.

  Para obtener más información sobre esta propiedad, consulte el parámetro supportedComponentSources de la acción de la API CreateServiceTemplateVersion en la Referencia de la API de AWS Proton.

  > **Note**
  >
  > Cuando utiliza la sincronización de plantillas, AWS Proton crea versiones de plantillas de servicio de forma implícita al confirmar los cambios a un paquete de plantillas de servicio en un repositorio. En este caso, en lugar de especificar los orígenes de componentes compatibles durante la creación de la versión de la plantilla de servicio, el usuario debe especificar esta propiedad en un archivo asociado a cada versión principal de la plantilla de servicio. Para obtener más información, consulte the section called “Sincronización de plantillas de servicio”.

- Roles de componentes: un administrador puede asignar un rol de componente a un entorno. AWS Proton asume este rol al aprovisionar la infraestructura que define un componente definido directamente en el entorno. Por lo tanto, el rol de componente abarca la infraestructura que los desarrolladores pueden añadir mediante componentes definidos directamente en el entorno.

Al no existir el rol de componente, los desarrolladores no pueden crear componentes definidos directamente en el entorno.

Para obtener más información sobre la asignación de un rol de componente, consulte el parámetro componentRoleArn de la acción de la API CreateEnvironment en la Referencia de la API de AWS Proton.

> **Note**
>
> Los roles de los componentes no se utilizan en los entornos de Aprovisionamiento autoadministrado.

Temas

¿Cómo se comparan los componentes con otros recursos de AWS Proton?

En muchos sentidos, los componentes son similares a otros recursos de AWS Proton. Su infraestructura se define en un archivo de plantilla de IaC, creado en formato YAML de AWS CloudFormation o Terraform HCL. AWS Proton puede aprovisionar la infraestructura de componentes mediante el aprovisionamiento administrado por AWS o el aprovisionamiento autoadministrado.

Sin embargo, los componentes se diferencian de otros recursos de AWS Proton en varios aspectos:

- Estado desconectado: los componentes están diseñados para conectarse a las instancias de servicio y ampliar su infraestructura, pero también pueden estar en un estado desconectado, en

el que no están conectados a ninguna instancia de servicio. Para obtener más información sobre componentes del sistema, consulte the section called “Estados de los componentes”.

- No hay ningún esquema: los componentes no tienen ningún esquema asociado a diferencia de los paquetes de plantillas. Un servicio define las entradas de los componentes. Un componente puede consumir entradas cuando está conectado a una instancia de servicio.
- No hay ningún componente administrado por el cliente: AWS Proton siempre aprovisiona al usuario la infraestructura de componentes. No hay ninguna versión de componentes del tipo “traiga sus propios recursos”. Para obtener más información sobre entornos administrados por el cliente, consulte the section called “Create”.
- No hay recursos de plantillas: los componentes definidos directamente no tienen ningún recurso de plantilla asociado similar a las plantillas de entorno y servicio. Debe proporcionar un archivo de plantilla de IaC directamente al componente. Del mismo modo, el usuario proporciona directamente un manifiesto que define el lenguaje de la plantilla y el motor de representación para aprovisionar la infraestructura del componente. El archivo de plantilla y el manifiesto se crean de forma similar a la creación de un paquete de plantillas. Sin embargo, con los componentes definidos directamente, no es necesario almacenar los archivos de IaC como paquetes en ubicaciones determinadas, ni se crea un recurso de plantilla en AWS Proton partir de archivos de IaC.
- Sin aprovisionamiento basado en CodeBuild: no puede aprovisionar componentes definidos directamente mediante su propio script de aprovisionamiento personalizado, conocido como aprovisionamiento basado en CodeBuild. Para obtener más información, consulte the section called “Aprovisionamiento de CodeBuild”.

Componentes de la consola de AWS Proton

Utilice la consola de AWS Proton para crear, actualizar, ver y utilizar componentes de AWS Proton.

Las siguientes páginas de la consola están relacionadas con los componentes. Incluimos enlaces directos a las páginas de consolas de nivel superior.

- Componentes: consulte la lista de componentes de su cuenta de AWS. Puede crear nuevos componentes y actualizar o eliminar los componentes existentes. Seleccione de la lista el nombre de algún componente para ver su página de detalles.

  También existen listas similares en las páginas Detalles del entorno y Detalles de la instancia de servicio. Estas listas muestran solo los componentes asociados al recurso que se está viendo.

Al crear un componente a partir de una de estas listas, AWS Proton preselecciona el entorno asociado en la página Crear componente.

- Detalles del componente: para ver la página de detalles del componente, elija el nombre de algún componente en la lista Componentes.

  En la página de detalles, consulte los detalles y el estado del componente y actualice o elimine el componente. Vea y administre las listas de salidas (por ejemplo, los ARN de recursos aprovisionados), las pilas de AWS CloudFormation aprovisionadas y las etiquetas asignadas.
- Crear componente: cree un componente. Introduzca el nombre y la descripción del componente, elija los recursos asociados, especifique el archivo de IaC de origen del componente y asigne etiquetas.
- Actualizar componente: para actualizar un componente, seleccione el componente en la lista Componentes y, a continuación, en el menú Acciones, elija Actualizar componente. Como alternativa, en las páginas de Detalles del componente, seleccione Actualizar.

  Puede actualizar la mayoría de los detalles del componente. No puede actualizar el nombre del componente. Además, puede elegir si desea volver a implementar o no el componente después de una actualización correcta.
- Configurar el entorno: al crear o actualizar un entorno, puede especificar un rol de componente. Este rol controla la capacidad de ejecutar componentes definidos directamente en el entorno y proporciona permisos para aprovisionarlos.
- Crear una nueva versión de la plantilla de servicio: al crear una versión de la plantilla de servicio, puede especificar los Orígenes de componentes compatibles para la versión de la plantilla. Esto controla la capacidad de asociar componentes a las instancias de servicio de los servicios basados en esta versión de plantilla.

Componentes de la API de AWS Proton y la AWS CLI

Utilice la API de AWS Proton o la AWS CLI para crear, actualizar, ver y usar componentes de AWS Proton.

Las siguientes acciones de la API administran directamente los recursos de los componentes de AWS Proton.

- CreateComponent: para crear un componente de AWS Proton.
- DeleteComponent: para eliminar un componente de AWS Proton.

- GetComponent: para obtener datos detallados de un componente.
- ListComponentOutputs: para obtener una lista de las salidas de los componentes de infraestructura como código (IaC).
- ListComponentProvisionedResources: para enumerar los recursos aprovisionados para un componente con detalles.
- ListComponents: lista de los componentes con datos resumidos. Puede filtrar la lista de resultados por entorno, por servicio o por una sola instancia de servicio.

Las siguientes acciones de API de otros recursos de AWS Proton tienen algunas funciones relacionadas con los componentes.

- CreateEnvironment, UpdateEnvironment: utilice `componentRoleArn` para especificar el nombre de recurso de Amazon (ARN) del rol de servicio de IAM que utiliza AWS Proton al aprovisionar componentes directamente definidos en este entorno. Determina el alcance de la infraestructura que puede aprovisionar un componente definido directamente.
- CreateServiceTemplateVersion: utilice `supportedComponentSources` para especificar los orígenes de componentes compatibles. Los componentes con orígenes compatibles se pueden asociar a las instancias de servicio en función de esta versión de plantilla de servicio.

Preguntas frecuentes sobre los componentes

¿Cuál es el ciclo de vida de un componente?

Los componentes pueden estar conectados o desconectados. Están diseñados para conectarse a una instancia de servicio y mejorar su infraestructura la mayor parte del tiempo. Los componentes desconectados se encuentran en un estado de transición que permite eliminar un componente o conectarlo a otra instancia de servicio de forma controlada y segura. Para obtener más información, consulte the section called “Estados de los componentes”.

¿Por qué no puedo eliminar los componentes conectados?

Solución: para eliminar un componente conectado, actualice el componente para desconectarlo de la instancia de servicio, valide la estabilidad de la instancia de servicio y, a continuación, elimine el componente.

¿Por qué es necesario? Los componentes conectados proporcionan la infraestructura adicional que la aplicación necesita para realizar sus funciones de tiempo de ejecución. La instancia de servicio

puede estar utilizando las salidas de los componentes para detectar y utilizar los recursos de esta infraestructura. Eliminar el componente y, por lo tanto, eliminar sus recursos de infraestructura, podría interrumpir la instancia de servicio conectada.

Como medida de seguridad adicional, AWS Proton requiere que actualice el componente y lo desconecte de su instancia de servicio antes de poder eliminarlo. A continuación, podrá validar la instancia de servicio para asegurarse de que sigue implementándose y funcionando correctamente. Si detecta algún problema, puede volver a conectar rápidamente el componente a la instancia de servicio y, a continuación, intentar solucionar el problema. Cuando tenga la certeza de que su instancia de servicio haya dejado de depender del componente, podrá eliminarlo de forma segura.

¿Por qué no puedo cambiar directamente la instancia de servicio conectada a un componente?

Solución: para cambiar la conexión, actualice el componente para desconectarlo de la instancia de servicio, valide la estabilidad del componente y de la instancia de servicio y, a continuación, conecte el componente a la nueva instancia de servicio.

¿Por qué es necesario? Un componente está diseñado para conectarse a una instancia de servicio. El componente puede utilizar las entradas de la instancia de servicio para la asignación de nombres y la configuración de los recursos de infraestructura. Cambiar la instancia de servicio conectada podría afectar negativamente al componente (además de provocar una posible interrupción en la instancia de servicio), tal y como se describe en la pregunta frecuente anterior: ¿Por qué no puedo eliminar los componentes conectados?). Por ejemplo, podría provocar el cambio de nombre, y posiblemente incluso la sustitución, de los recursos definidos en la plantilla de la IaC del componente.

Como medida de seguridad adicional, AWS Proton requiere que actualice el componente y lo desconecte de su instancia de servicio antes de poder conectarlo a otra instancia de servicio. A continuación, podrá validar la estabilidad tanto del componente como de la instancia de servicio antes de conectar el componente a la nueva instancia de servicio.

Estados de los componentes

Los componentes de AWS Proton pueden estar en dos estados fundamentalmente diferentes:

- Conectado: el componente está conectado a una instancia de servicio. Define la infraestructura que admite la funcionalidad de tiempo de ejecución de la instancia de servicio. El componente amplía la infraestructura definida en las plantillas de entorno y servicio con una infraestructura definida por el desarrollador.

Un componente típico está conectado durante la mayor parte de su ciclo de vida útil.

- Desconectado: el componente está asociado a un entorno de AWS Proton y no está conectado a ninguna instancia de servicio del entorno.

  Se trata de un estado de transición para prolongar la vida útil de un componente más allá de una única instancia de servicio.

La siguiente tabla proporciona una comparación general de los distintos estados de los componentes.

	Conectado	Desconectado
Objetivo principal del estado	Ampliar la infraestructura de una instancia de servicio.	Para mantener la infraestructura del componente entre conexiones de instancias de servicio.
Asociado a	Una instancia de servicio y un entorno	Un entorno
Propiedades específicas clave	• Nombre del servicio • Nombre de la instancia de servicio • Especificación	• Nombre del entorno
Se puede eliminar	✗ No	✓ Sí
Se puede actualizar a otra instancia de servicio	✗ No	✓ Sí
Puede leer entradas	✓ Sí	✗ No

El objetivo principal de un componente es conectarse a una instancia de servicio y ampliar su infraestructura con recursos adicionales. Un componente conectado puede leer las entradas de la

instancia de servicio según las especificaciones. No puede eliminar directamente el componente ni conectarlo a una instancia de servicio diferente. Tampoco puede eliminar su instancia de servicio ni el servicio y el entorno relacionados. Para hacer cualquiera de estas cosas, actualice primero el componente para separarlo de su instancia de servicio.

Para mantener la infraestructura del componente más allá de la vida útil de una sola instancia de servicio, debe actualizar el componente y desconectarlo de su instancia de servicio; para ello, elimine los nombres del servicio y de la instancia de servicio. Este estado desconectado es un estado de transición. El componente no tiene entradas. Su infraestructura permanecerá aprovisionada y podrá actualizarla. Puede eliminar los recursos a los que estaba asociado el componente cuando se conectó (instancia de servicio, servicio). Puede eliminar el componente o actualizarlo para volver a conectarlo a una instancia de servicio.

Infraestructura de componentes en forma de archivos de código

Los archivos de infraestructura como código (IaC) de los componentes son similares a los de otros recursos de AWS Proton. Aquí encontrará información sobre algunos detalles específicos de los componentes. Para obtener información completa sobre la creación de archivos de IaC para AWS Proton, consulte Creación de plantillas y paquetes.

Uso de parámetros con componentes

El espacio de nombres de parámetros de AWS Proton incluye algunos parámetros a los que puede hacer referencia un archivo de IaC de un servicio para obtener el nombre y las salidas de un componente asociado. El espacio de nombres también incluye parámetros a los que puede hacer referencia un archivo de IaC de un componente para obtener entradas, salidas y valores de recursos del entorno, el servicio y la instancia de servicio a los que está asociado el componente.

Un componente no tiene entradas propias, sino que las obtiene de la instancia de servicio a la que esté conectado. Un componente también puede leer las salidas del entorno.

Para obtener más información sobre el uso de parámetros en los archivos de IaC de los componentes y servicios asociados, consulte the section called “Parámetros del componente CloudFormation IaC”. Para obtener información general sobre los parámetros de AWS Proton y una referencia completa del espacio de nombres de los parámetros, consulte the section called “Parámetros”.

Creación de archivos de IaC robustos

Como administrador, al crear una versión de plantilla de servicio, puede decidir si desea permitir que las instancias de servicio creadas a partir de la versión de plantilla tengan componentes conectados. Consulte el parámetro supportedComponentSources de la acción de la API CreateServiceTemplateVersion en la Referencia de la API de AWS Proton. Sin embargo, para cualquier instancia de servicio futura, la persona que crea la instancia decide si conecta o no un componente en ella y (en el caso de los componentes definidos directamente) quien crea la IaC del componente suele ser una persona diferente: un desarrollador que utiliza su plantilla de servicio. Por lo tanto, el usuario no puede garantizar que un componente se conecte a una instancia de servicio. Tampoco puede garantizar la existencia de nombres de salida de componentes específicos ni la validez y seguridad de los valores de estas salidas.

AWS Proton y la sintaxis de Jinja le ayudan a solucionar estos problemas y a crear plantillas de servicio robustas que se representan sin errores de las siguientes maneras:

- Filtros de parámetros de AWS Proton: cuando hace referencia a las propiedades de salida de los componentes, puede utilizar filtros de parámetros, es decir, modificadores que validan y filtran los valores de los parámetros y además les dan formato. Para obtener más información y ejemplos, consulte the section called “CloudFormation filtros de parámetros”.
- Propiedad predeterminada única: cuando el usuario hace referencia a un único recurso o propiedad de salida de un componente, puede garantizar que no se producirán errores en la representación de la plantilla de servicio mediante el uso del filtro `default`, con o sin un valor predeterminado. Si el componente o un parámetro de salida específico al que el usuario hace referencia no existe, se representa en su lugar el valor predeterminado (o una cadena vacía, si no se ha especificado ningún valor predeterminado) y dicha representación se realiza correctamente. Para obtener más información, consulte the section called “Provisión de valores predeterminados”.

  Ejemplos:

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

> **Note**
>
> No hay que confundir la parte `.default` del espacio de nombres, que designa los componentes definidos directamente, con el filtro `default`, que proporciona un valor predeterminado cuando la propiedad a la que se hace referencia no existe.

- Referencia de objeto completo: cuando se hace referencia a todo el componente o al conjunto de salidas de un componente, AWS Proton devuelve un objeto vacío, `{}`, y, por lo tanto, garantiza que no se producirá ningún error al representar la plantilla de servicio. No es necesario que utilice ningún filtro. Asegúrese de hacer la referencia en un contexto que pueda incluir un objeto vacío o utilice una condición `{{ if .. }}` para comprobar si hay un objeto vacío.

  Ejemplos:

  - `{{ service_instance.components.default }}`
  - `{{ service_instance.components.default.outputs }}`

Ejemplo de componentes de AWS CloudFormation

A continuación, se muestra un ejemplo completo de un componente definido directamente de AWS Proton y de cómo se puede utilizar en un servicio de AWS Proton. El componente aprovisiona un bucket de Amazon Simple Storage Service (Amazon S3) y una política de acceso relacionada. La instancia de servicio puede hacer referencia a este bucket y utilizarlo. El nombre del bucket se basa en los nombres del entorno, el servicio, la instancia de servicio y el componente, lo que significa que el bucket está asociado a una instancia específica de la plantilla del componente que amplía una instancia de servicio específica. Los desarrolladores pueden crear varios componentes en función de esta plantilla de componentes para aprovisionar buckets de Amazon S3 para distintas instancias de servicio y necesidades funcionales.

El ejemplo abarca la creación de varios archivos obligatorios de infraestructuras como código (IaC) de AWS CloudFormation y la creación de un rol de AWS Identity and Access Management (IAM) obligatorio. El ejemplo agrupa los pasos según los roles de las personas propietarias.

Pasos de administrador

Para permitir a los desarrolladores utilizar componentes con un servicio

1. Cree un rol de AWS Identity and Access Management (IAM) que reduzca los recursos que pueden aprovisionar los componentes definidos directamente que se ejecutan en el entorno. AWS Proton asume este rol más adelante para aprovisionar componentes directamente definidos en el entorno.

   Para este ejemplo, utilice la siguiente política:

   Example Rol de componente definido directamente

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
```

```
        "s3:GetBucket",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

2. Proporcione el rol que creó en el paso anterior al crear o actualizar el entorno. En la consola de AWS Proton, especifique un Rol de componente en la página Configurar entorno. Si utiliza la API de AWS Proton o la AWS CLI, especifique el `componentRoleArn` de las acciones de la API CreateEnvironment o UpdateEnvironment.
3. Cree una plantilla de servicio que haga referencia a un componente definido directamente conectado a la instancia de servicio.

   El ejemplo muestra cómo escribir una plantilla de servicio sólida que no se rompa si un componente no estuviera conectado a la instancia de servicio.

   Example archivo de IaC de servicio de CloudFormation mediante un componente

```
# service/instance_infrastructure/cloudformation.yaml

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

```
                proton_cfn_ecs_task_definition_formatted_env_vars }}
        {% endif %}

# ...

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

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

4. Cree una nueva versión secundaria de la plantilla de servicio que declare compatibles los componentes definidos directamente.

   - Paquete de plantillas en Amazon S3: en la consola de AWS Proton, al crear una versión de plantilla de servicio, en Orígenes de componentes compatibles, elija Definición directa. Si utiliza la API de AWS Proton o la AWS CLI, especifique `DIRECTLY_DEFINED` en el parámetro `supportedComponentSources` de las acciones de la API CreateServiceTemplateVersion o UpdateServiceTemplateVersion.
   - Sincronización de plantillas: confirme un cambio en el repositorio del paquete de plantillas de servicio, donde especifique `DIRECTLY_DEFINED` como elemento de `supported_component_sources:` en el archivo `.template-registration.yaml` del directorio de versiones principales. Para obtener más información acerca de este archivo, consulte the section called “Sincronización de plantillas de servicio”.

5. Publique la nueva versión secundaria de la plantilla de servicio. Para obtener más información, consulte the section called “Publicación”.

6. Asegúrese de permitir el permiso `proton:CreateComponent` en el rol de IAM de los desarrolladores que utilicen esta plantilla de servicio.

Pasos para desarrolladores

Para utilizar un componente definido directamente con una instancia de servicio

1. Cree un servicio que utilice la versión de la plantilla de servicio que el administrador creó con la ayuda de los componentes. Como alternativa, actualice una de sus instancias de servicio existentes para utilizar la última versión de la plantilla.
2. Escriba un archivo de plantilla de IaC de componentes que aprovisione un bucket de Amazon S3 y una política de acceso relacionada y exponga estos recursos como salidas.

   Example archivo de IaC de componente de CloudFormation

```
# cloudformation.yaml

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

3. Si utiliza la API de AWS Proton o la AWS CLI, escriba un archivo de manifiesto para el componente.

Example manifiesto de componentes definidos directamente

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

4. Cree un componente definido directamente. AWS Proton asume el rol de componente que el administrador definió para aprovisionar el componente.

   En la consola de AWS Proton, en la página Componentes, elija Crear componente. En Configuración de componentes, introduzca un Nombre del componente y una Descripción del componente opcional. En Conexión de componentes, elija Conectar el componente a una instancia de servicio. Seleccione el entorno, el servicio y la instancia de servicio. En Origen del componente, elija AWS CloudFormation y, a continuación, elija el archivo de IaC del componente.

   > **Note**
   >
   > No es necesario proporcionar un manifiesto, ya que la consola lo crea automáticamente.

   Si utiliza la API de AWS Proton o la AWS CLI, utilice la acción de la API CreateComponent. Establezca el `name` del componente y una `description` opcional. Establezca `environmentName`, `serviceName` y `serviceInstanceName`. Establezca `templateSource` y `manifest` para las rutas de los archivos que ha creado.

   > **Note**
   >
   > La especificación de un nombre de entorno es opcional cuando se especifican los nombres de los servicios y de las instancias de servicio. La combinación de estos dos elementos es única en su cuenta de AWS, y AWS Proton puede determinar el entorno a partir de la instancia de servicio.

5. Actualice la instancia de servicio para volver a implementarla. AWS Proton utiliza las salidas del componente en la plantilla de instancia de servicio representada para permitir que la aplicación utilice el bucket de Amazon S3 que el componente aprovisionó.

Uso de repositorios de Git con AWS Proton

AWS Proton utiliza los repositorios de Git para una variedad de propósitos. La siguiente lista clasifica los tipos de repositorios asociados a los recursos de AWS Proton. En el caso de los roles de AWS Proton que se conectan repetidamente al repositorio del usuario para enviar contenido a él o extraerlo de él, hay que registrar un enlace al repositorio con AWS Proton en la cuenta de AWS del usuario. Un enlace a un repositorio es un conjunto de propiedades que AWS Proton puede utilizar cuando se conecta a un repositorio. AWS Proton actualmente es compatible con GitHub, GitHub Enterprise y BitBucket.

Repositorios para desarrolladores

Repositorio de código: un repositorio que los desarrolladores utilizan para almacenar el código de las aplicaciones. Se utiliza para la implementación de código. AWS Proton no interactúa directamente con este repositorio. Cuando un desarrollador aprovisiona un servicio que incluye una canalización, proporciona el nombre del repositorio y la ramificación desde la que leer el código de la aplicación. AWS Proton transfiere esta información a la canalización que aprovisiona.

Para obtener más información, consulte the section called “Create”.

Repositorios de administradores

Repositorio de plantillas: un repositorio en el que los administradores almacenan paquetes de plantillas de AWS Proton. Se utiliza para la sincronización de plantillas. Cuando un administrador crea una plantilla en AWS Proton, puede apuntar a un repositorio de plantillas y AWS Proton mantiene la nueva plantilla sincronizada con dicho repositorio. Cuando el administrador actualiza el paquete de plantillas en el repositorio, AWS Proton crea automáticamente una nueva versión de la plantilla. Vincule un repositorio de plantillas AWS Proton antes de poder utilizarlo para la sincronización.

Para obtener más información, consulte the section called “Configuraciones de sincronización de plantillas”.

> **Note**
>
> No es necesario disponer de un repositorio de plantillas si el usuario sigue cargando sus plantillas en Amazon Simple Storage Service (Amazon S3) y llamando a las API de administración de plantillas de AWS Proton para crear nuevas plantillas o versiones de plantillas.

Repositorios de aprovisionamiento autoadministrado

Repositorio de infraestructura: repositorio que aloja plantillas de infraestructura representadas. Se utiliza para el aprovisionamiento autoadministrado de la infraestructura de recursos. Cuando un administrador crea un entorno para el aprovisionamiento autoadministrado, proporciona un repositorio. AWS Proton envía las solicitudes de extracción (PR) a este repositorio para crear la infraestructura del entorno y de cualquier instancia de servicio implementada en el entorno. Vincule un repositorio de infraestructura a AWS Proton antes de poder utilizarlo para el aprovisionamiento de infraestructura autoadministrado.

Repositorio de canalizaciones: repositorio que se utiliza para crear canalizaciones. Se utiliza para el aprovisionamiento autoadministrado de canalizaciones. El uso de un repositorio adicional para aprovisionar las canalizaciones permite a AWS Proton almacenar las configuraciones de las canalizaciones de forma independiente de cualquier entorno o servicio individual. Solo necesita proporcionar un repositorio de canalización único para todos los servicios de aprovisionamiento autoadministrados. Vincule un repositorio de canalizaciones a AWS Proton antes de poder utilizarlo para el aprovisionamiento de canalizaciones autoadministrado.

Para obtener más información, consulte the section called “Aprovisionamiento administrado por AWS”.

Temas

Creación de un enlace para el repositorio del usuario

El usuario puede crear un enlace a su repositorio mediante la consola o la CLI. Cuando se crea un enlace a un repositorio, AWS Proton crea un rol vinculado a un servicio automáticamente.

AWS Management Console

Cree un enlace a su repositorio como se muestra en los siguientes pasos de la consola.

1. En la consola de AWS Proton, seleccione Repositorios.
2. Elija Create repository.

3. En la página Vincular nuevo repositorio, en la sección de Detalles del repositorio:
   a. Elija el proveedor de repositorios.
   b. Elija una de las conexiones existentes. Si no tiene ninguna, seleccione Añadir una nueva conexión de CodeStar para crear una conexión y, a continuación, vuelva la consola de AWS Proton, actualice la lista de conexiones y elija la nueva conexión.
   c. Elija alguno de los repositorios de código fuente conectados.
4. [opcional] En la sección Etiquetas, seleccione Añadir nueva etiqueta una o más veces e introduzca los pares de clave y valor.
5. Elija Create repository.
6. Consulte los datos detallados del repositorio vinculado.

AWS CLI

Cree y registre un enlace a su repositorio.

Ejecute el siguiente comando:

```
$ aws proton create-repository \
    --name myrepos/environments \
    --connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
    --provider "GITHUB" \
    --encryption-key "arn:aws:kms:region-id:123456789012:key/bPxRfiCYEXAMPLEKEY" \
    --tags key=mytag1,value=value1 key=mytag2,value=value2
```

Los últimos dos parámetros, --encryption-key y --tags, son opcionales.

Respuesta:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
        "connectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/2ad03b28-a7c4-EXAMPLE11111",
        "encryptionKey": "arn:aws:kms:region-id:123456789012:key/
bPxRfiCYEXAMPLEKEY",
        "name": "myrepos/environments",
        "provider": "GITHUB"
```

```
    }
}
```

Tras crear un enlace al repositorio, se podrá ver una lista de etiquetas administradas por el cliente y por AWS, tal y como se muestra en el siguiente comando de ejemplo. AWS Proton genera automáticamente las etiquetas administradas por AWS para el usuario. También puede modificar y crear etiquetas administradas por el cliente mediante la AWS CLI. Para obtener más información, consulte Recursos y etiquetado de AWS Proton.

Comando:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/environments"
```

Visualización de los datos del repositorio vinculado

Puede enumerar y ver los detalles del repositorio vinculado mediante la consola o la AWS CLI. En el caso de los enlaces a los repositorios que se utilizan para sincronizar los repositorios de Git con AWS Proton, se puede recuperar la definición y el estado de la sincronización del repositorio mediante la AWS CLI.

AWS Management Console

Enumere y consulte los detalles de los repositorios vinculados mediante la consola de AWS Proton.

1. Para ver una lista de los repositorios vinculados, seleccione Repositorios en el panel de navegación.
2. Para ver los datos detallados, seleccione el nombre de algún repositorio.

AWS CLI

Enumere los repositorios vinculados.

Ejecute el siguiente comando:

```
$ aws proton list-repositories
```

Respuesta:

```
{
    "repositories": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
            "name": "myrepos/templates",
            "provider": "GITHUB"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
            "name": "myrepos/environments",
            "provider": "GITHUB"
        }
    ]
}
```

Consulte los detalles de un repositorio vinculado.

Ejecute el siguiente comando:

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

Respuesta:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Enumere los repositorios sincronizados.

En el siguiente ejemplo, se enumeran los repositorios que el usuario configuró para la sincronización de plantillas.

Ejecute el siguiente comando:

```
$ aws proton list-repository-sync-definitions \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Vea el estado de sincronización del repositorio.

En el siguiente ejemplo se recupera el estado de sincronización de un repositorio de sincronización de plantillas.

Ejecute el siguiente comando:

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Respuesta:

```
{
    "latestSync": {
        "events": [
            {
                "event": "Clone started",
                "time": "2021-11-21T00:26:35.883000+00:00",
                "type": "CLONE_STARTED"
            },
            {
                "event": "Updated configuration",
                "time": "2021-11-21T00:26:41.894000+00:00",
                "type": "CONFIG_UPDATED"
            },
            {
                "event": "Starting syncs for commit 62c03ff86eEXAMPLE1111111",
                "externalId": "62c03ff86eEXAMPLE1111111",
                "time": "2021-11-21T00:26:44.861000+00:00",
                "type": "STARTING_SYNC"
            }
```

```
        ],
        "startedAt": "2021-11-21T00:26:29.728000+00:00",
        "status": "SUCCEEDED"
    }
}
```

Eliminación de un enlace a un repositorio

Puede eliminar un enlace de repositorio mediante la consola o la AWS CLI.

> **Note**
>
> Al eliminar un enlace al repositorio, solo se elimina el enlace registrado que AWS Proton tiene en la cuenta de AWS del usuario. No elimina ninguna información del repositorio del usuario.

AWS Management Console

Elimine un enlace al repositorio mediante la consola.

En la página de detalles del repositorio.

1. En la consola de AWS Proton, seleccione Repositorios.
2. En la lista de repositorios, elija el botón de radio a la izquierda del repositorio que desee eliminar.
3. Elija Eliminar.
4. Un modal le pedirá que confirme la acción de eliminación.
5. Siga las instrucciones y seleccione Sí, eliminar.

AWS CLI

Elimine un enlace a un repositorio.

Ejecute el siguiente comando:

```
$ aws proton delete-repository \
    --name myrepos/templates \
```

```
    --provider"GITHUB"
```

Respuesta:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Supervisión AWS Proton

El monitoreo es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de AWS Proton sus AWS soluciones. En la siguiente sección se describen las herramientas de supervisión con las que puede utilizar AWS Proton.

Automatice AWS Proton con EventBridge

Puedes monitorizar AWS Proton los eventos en Amazon EventBridge. EventBridge ofrece un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones software-as-a-service (SaaS) y. Servicios de AWS Puede configurar los eventos para que respondan a los cambios en el estado AWS de los recursos. EventBridge luego, dirige estos datos a los servicios de destino, como AWS Lambda Amazon Simple Notification Service. Estos eventos son los mismos que aparecen en Amazon CloudWatch Events. CloudWatch Events ofrece un flujo casi en tiempo real de los eventos del sistema que describen los cambios en AWS los recursos. Para obtener más información, consulta ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon.

Se utiliza EventBridge para recibir notificaciones de los cambios de estado en los flujos de trabajo de AWS Proton aprovisionamiento.

Tipos de eventos

Los eventos se componen de reglas que incluyen un patrón de eventos y destinos. Para configurar una regla, elija el patrón de eventos y los objetos de destino:

Patrón de eventos

> Cada regla se expresa como un patrón de eventos con el código fuente y el tipo de eventos que se van a monitorizar, así como los destinos del evento. Para supervisar los eventos, debe crear una regla con el servicio que esté monitoreando como origen del evento. Por ejemplo, puede crear una regla con un patrón de eventos que utilice AWS Proton como origen de eventos para activar una regla cuando se produzcan cambios en el estado de una implementación.

Destinos

> La nueva regla recibe un servicio seleccionado como destino de eventos. Puede configurar un servicio de destino para enviar notificaciones, capturar información de estado, tomar medidas correctivas, iniciar eventos o adoptar otras medidas.

Los objetos de eventos contienen campos estándar como ID, cuenta Región de AWS, tipo de detalle, fuente, versión, recurso y hora (opcional). El campo de detalle es un objeto anidado que contiene campos personalizados para el evento.

AWS Proton los eventos se emiten en función del mejor esfuerzo posible. La entrega al máximo esfuerzo significa que el servicio intenta enviar todos los eventos a EventBridge, pero en algunos casos excepcionales es posible que un evento no se entregue.

Para cada AWS Proton recurso que puede emitir eventos, la siguiente tabla muestra el valor del tipo de detalle, los campos de detalle y (si está disponible) una referencia a una lista de valores para los `status` campos de `previousStatus` detalle. Cuando se elimina un recurso, el valor del campo de `status` detalle es `DELETED`.

Recurso	Valor de tipo de detalle	Campos de detalle

`EnvironmentTemplate`	AWS Proton Cambio de estado de la plantilla de entorno	`name`
		
		`status`
		
		`previousStatus`

`EnvironmentTemplateVersion`	AWS Proton Cambio de estado de la versión de la plantilla de entorno	`name`
		
		`majorVersion`
		
		`minorVersion`
		
		`status`
		
		`previousStatus`
		
		valores de estado

Recurso	Valor de tipo de detalle	Campos de detalle

`ServiceTemplate`	AWS Proton Cambio de estado de la plantilla de servicio	`name`
		`status`
		`previousStatus`

`ServiceTemplateVersion`	AWS Proton Cambio de estado de la versión de la plantilla de servicio	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		valores de estado

`Environment`	AWS Proton Cambio de estado del entorno	`name`
		`status`
		`previousStatus`

Recurso	Valor de tipo de detalle	Campos de detalle

`Service`	AWS Proton Cambio de estado del servicio	`name`
		`status`
		`previousStatus`
		valores de estado

`ServiceInstance`	AWS Proton Cambio de estado de la instancia de servicio	`name`
		`serviceName`
		`status`
		`previousStatus`

`ServicePipeline`	AWS Proton Cambio de estado de Service Pipeline	`serviceName`
		`status`
		`previousStatus`

`EnvironmentAccountConnection`	AWS Proton Cambio de estado de conexión de la cuenta de entorno	`id`
		`status`
		`previousStatus`
		valores de estado

Recurso	Valor de tipo de detalle	Campos de detalle

`Component`	AWS Proton Cambio de estado del componente	`name`
		
		`status`
		
		`previousStatus`

AWS Proton ejemplos de eventos

Los siguientes ejemplos muestran las formas en que AWS Proton se pueden enviar eventos a EventBridge.

Plantilla de servicio

```
{
    "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"
    }
}
```

Versión de la plantilla de servicio

```
{
    "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"
    }
}
```

Entorno

```
{
    "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: Envía alertas de Amazon Simple Notification Service sobre cambios en el estado del AWS Proton servicio

En este tutorial, utilizará una regla de eventos AWS Proton preconfigurada que captura los cambios de estado de su AWS Proton servicio. EventBridgeenvía los cambios de estado a un tema de Amazon SNS. Te suscribes al tema y Amazon SNS te envía correos electrónicos de cambio de estado para tu AWS Proton servicio.

Requisitos previos

Tiene un AWS Proton servicio existente con un `Active` estado. Como parte de este tutorial, podrá añadir instancias de servicio a este servicio y, a continuación, eliminarlas.

Si necesita crear un AWS Proton servicio, consulteIntroducción. Para obtener más información, consulte Cuotas de AWS Proton y the section called “Edición”.

Paso 1: Crear y suscribirse a un tema de Amazon SNS

Cree un tema de Amazon SNS para que sirva como destino de eventos para la regla de eventos que cree en el paso 2.

Crear un tema de Amazon SNS

1. Inicie sesión y abra la consola de Amazon SNS.
2. En el panel de navegación, elija Temas y, a continuación, seleccione Crear un tema.
3. En la página Crear un tema:
   a. Seleccione Tipo Estándar.
   b. Para Nombre, escriba **`tutorial-service-status-change`** y, a continuación, elija Crear tema.
4. En la página de tutorial-service-status-changedetalles, selecciona Crear suscripción.
5. En la página Crear suscripción:
   a. En Protocolo, elija Correo electrónico.
   b. En Punto de conexión, ingrese una dirección de email a la que actualmente tenga acceso y elija Crear suscripción.
6. Consulte su cuenta de email y espere para recibir un mensaje de correo electrónico de confirmación de la suscripción. Cuando lo reciba, ábralo y seleccione Confirmar suscripción.

Paso 2: Registrar una regla de eventos

Registra una regla de eventos que capture los cambios de estado de tu AWS Proton servicio. Para obtener más información, consulte Requisitos previos.

Cree una regla de eventos.

1. Abre la EventBridge consola de Amazon.
2. En el panel de navegación, elija Eventos, Reglas.
3. En la página Reglas, en la sección Reglas, seleccione Crear regla.
4. En la página Crear regla:
   a. En la sección Nombre y descripción, en Nombre, introduzca **`tutorial-rule`**.
   b. En la sección Definir patrón, elija Patrón de eventos.

i. En Evento coincidente con patrón, elija Predefinido por servicio.

ii. En Proveedor de servicios, elija AWS.

iii. En Nombre de servicio, seleccione AWS Proton.

iv. Para Tipo de evento, seleccione Cambio de estado de servicio de AWS Proton .

El Patrón de eventos aparece en un editor de texto.

v. Abra la consola de AWS Proton.

vi. En el panel de navegación, elija Servicios.

vii. En la página de servicios, elige el nombre de tu AWS Proton servicio.

viii. En la página Detalles del servicio, copie el Nombre de recurso de Amazon (ARN) del servicio.

ix. Vuelve a la EventBridge consola y a la regla del tutorial y selecciona Editar en el editor de texto.

x. En el editor de texto, para `"resources":`, introduzca el ARN de servicio que copió en el paso viii.

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

xi. Guarde el patrón de eventos.

c. En la sección Seleccionar destinos:

i. En Destino, elija Tema de SNS.

ii. En Tema, selecciona tutorial-service-status-change.

d. Seleccione Crear.

Paso 3: Comprobación de la regla de eventos

Comprueba que la regla de eventos funciona añadiendo una instancia a tu AWS Proton servicio.

1. Vuelva a la consola de AWS Proton.

2. En el panel de navegación, elija Servicios.
3. En la página Servicios, elija el nombre del servicio.
4. En la página de Detalles del servicio, seleccione Editar.
5. En la página Configurar servicio, seleccione Siguiente.
6. En la página Configurar ajustes personalizados, en la sección Instancias de servicio, elija Agregar nueva instancia.
7. Complete el formulario para la Nueva instancia:
   a. Escriba un Nombre para la nueva instancia.
   b. Seleccione los mismos entornos compatibles que eligió para las instancias existentes.
   c. Introduzca valores para las entradas requeridas.
   d. Elija Siguiente.
8. Revise los datos introducidos y seleccione Actualizar.
9. Una vez que el estado del servicio sea`Active`, comprueba tu correo electrónico para comprobar que has recibido AWS notificaciones con actualizaciones de estado.

```
{
    "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"
    }
}
```

Paso 4: Limpiar

Elimine el tema y la suscripción de Amazon SNS y elimine la regla. EventBridge

Elimine el tema y la suscripción de Amazon SNS.

1. Vaya a la consola de Amazon SNS.
2. En el panel de navegación, seleccione Suscripciones.
3. En la página de suscripciones, elija la suscripción que realizó al tema denominado `tutorial-service-status-change` y, a continuación, seleccione Eliminar.
4. En el panel de navegación, elija Temas.
5. En la página Temas, elija el tema denominado `tutorial-service-status-change` y, a continuación, seleccione Eliminar.
6. Un modal le pedirá que verifique la eliminación. Siga las instrucciones y seleccione Eliminar.

Elimine su EventBridge regla.

1. Ve a la EventBridge consola de Amazon.
2. En el panel de navegación, elija Eventos, Reglas.
3. En la página Reglas, elija la regla denominada `tutorial-rule` y, a continuación, seleccione Eliminar.
4. Un modal le pedirá que verifique la eliminación. Elija Eliminar.

Elimine la instancia de servicio agregada.

1. Vaya a la consola de AWS Proton.

2. En el panel de navegación, elija Servicios.
3. En la página Servicios, elija el nombre del servicio.
4. En la página de detalles del Servicio, seleccione Editar y, a continuación, Siguiente.
5. En la página Configurar ajustes personalizados, en la sección Instancias de servicio, elija Eliminar para la instancia de servicio que creó como parte de este tutorial y, a continuación, elija Siguiente.
6. Revise los datos introducidos y seleccione Actualizar.
7. Un modal le pedirá que verifique la eliminación. Siga las instrucciones y seleccione Sí, eliminar.

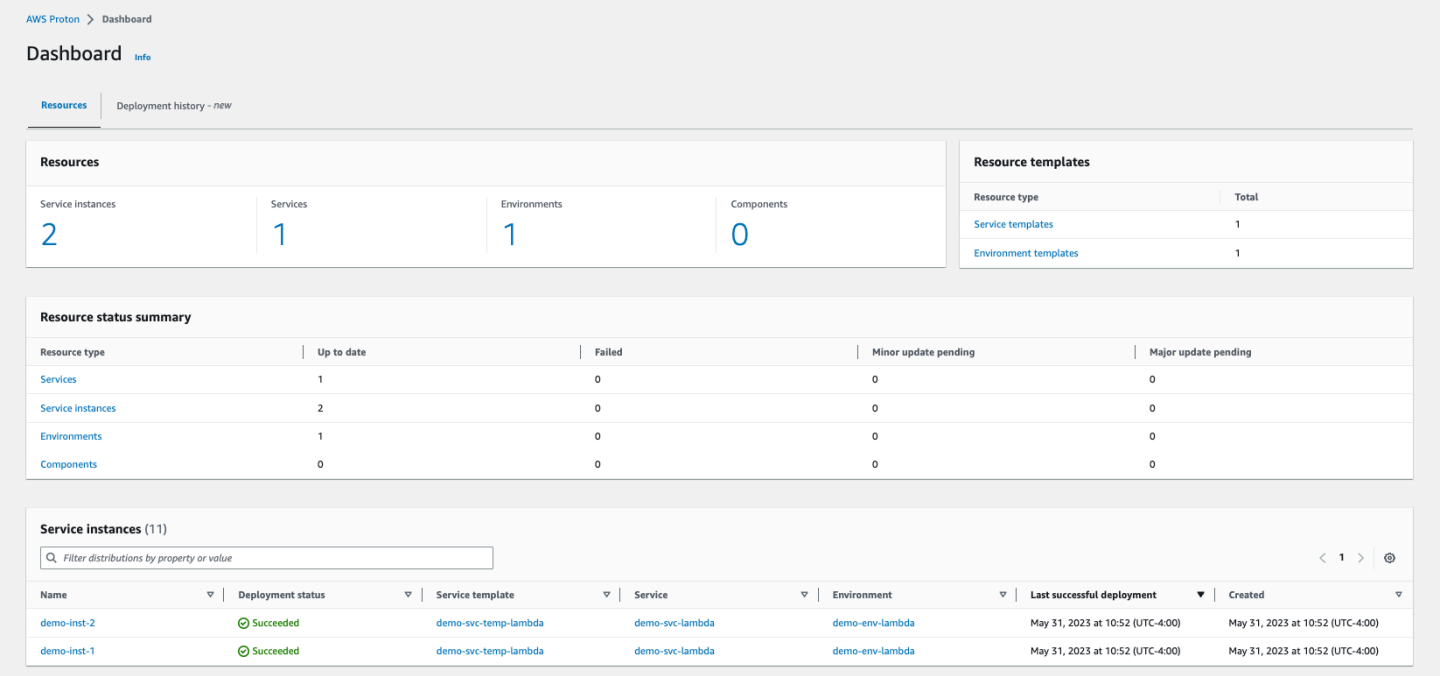
Mantén la infraestructura actualizada con el AWS Proton panel de control

El AWS Proton panel proporciona un resumen de AWS Proton los recursos de su AWS cuenta, con especial atención a la obsolescencia (es decir, el grado de actualización de los recursos desplegados). Un recurso implementado estará actualizado cuando utilice la versión recomendada de su plantilla asociada. Es posible que un recurso out-of-date implementado necesite una actualización mayor o menor de la versión de la plantilla.

Vea el panel de control en la AWS Proton consola

Para ver el AWS Proton panel de control, abra la AWS Proton consola y, a continuación, en el panel de navegación, seleccione Panel de control.

Recursos

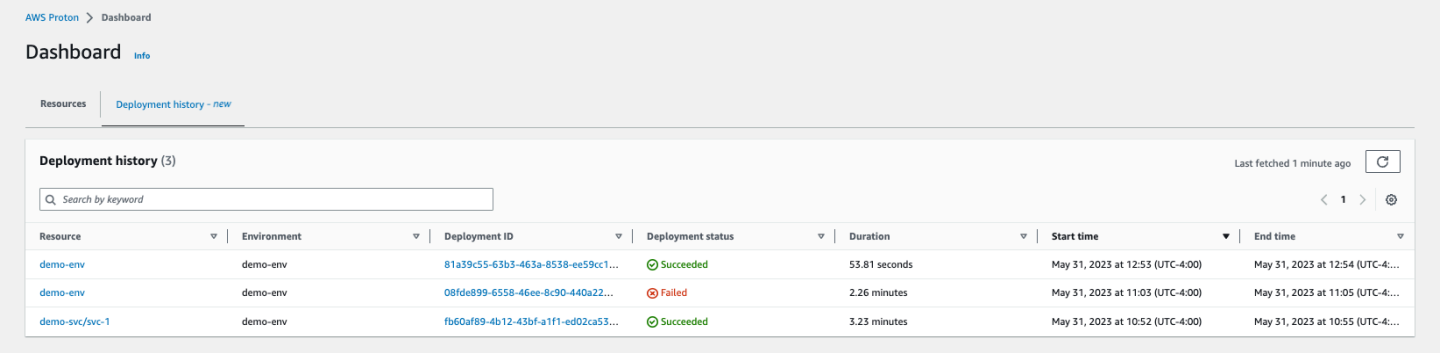


La primera pestaña del panel muestra los recuentos de todos los recursos de la cuenta del usuario. La pestaña de recursos muestra el número de sus instancias de servicio, servicios, entornos y componentes, así como sus plantillas de recursos. También desglosa los recuentos de recursos de cada tipo de recurso implementado según el estado de los recursos del tipo en cuestión. Una tabla de instancias de servicio muestra los detalles de cada instancia de servicio: su estado de implementación, los AWS Proton recursos a los que está asociada, las actualizaciones que están disponibles y algunas marcas de tiempo.

Puede filtrar la lista de instancias de servicio por cualquier propiedad de la tabla. Por ejemplo, puede utilizar filtros para ver tanto las instancias de servicio con implementaciones dentro de un intervalo de tiempo específico como las instancias de servicio que estén desactualizadas en relación con las recomendaciones de versiones principales o secundarias.

Elija el nombre de una instancia de servicio para ir a la página de detalles de la instancia de servicio, donde podrá realizar las actualizaciones de versión adecuadas. Elija cualquier otro nombre de AWS Proton recurso para ir a su página de detalles o elija un tipo de recurso para ir a la lista de recursos correspondiente.

Historial de implementaciones



La pestaña del historial de implementaciones permite al usuario ver los detalles de sus implementaciones. En la tabla del historial de implementaciones, puede realizar un seguimiento del estado de la implementación, así como del entorno y del ID de la implementación. Puede elegir el nombre del recurso o el ID de la implementación para ver aún más detalles, como un mensaje sobre el estado de la implementación y las salidas de recursos. La tabla también le permite filtrar por cualquier propiedad de la tabla.

Seguridad en AWS Proton

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y de centros de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta Servicios de AWS en Nube de AWS. Además, AWS proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWS Programas de conformidad de . Para obtener información sobre los programas de conformidad que se aplican a AWS Proton, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el Servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza AWS Proton. En los siguientes temas, se le mostrará cómo configurar AWS Proton para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros Servicios de AWS que le ayudarán a supervisar y a proteger los recursos de AWS Proton.

Temas

Identity and Access Management para AWS Proton

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. AWS Proton La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice. AWS Proton

Usuario del servicio: si utiliza el AWS Proton servicio para realizar su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más AWS Proton funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en AWS Proton, consulte Solución de problemas de AWS Proton identidad y acceso.

Administrador de servicios: si estás a cargo de AWS Proton los recursos de tu empresa, probablemente tengas acceso total a ellos AWS Proton. Su trabajo consiste en determinar a qué AWS Proton funciones y recursos deben acceder los usuarios del servicio. Luego, debe enviar

solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar la IAM AWS Proton, consulte¿Cómo AWS Proton funciona con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS. Para ver ejemplos de políticas AWS Proton basadas en la identidad que puede utilizar en IAM, consulte. Ejemplos de políticas basadas en la identidad para AWS Proton

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener

información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una

política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites

de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.

- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations .
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

¿Cómo AWS Proton funciona con IAM

Antes de utilizar IAM para gestionar el acceso AWS Proton, infórmese sobre las funciones de IAM disponibles para su uso. AWS Proton

Funciones de IAM que puede utilizar con AWS Proton

Característica de IAM	AWS Proton soporte
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No

Característica de IAM	AWS Proton soporte
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	Sí
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	Sí

Para obtener una visión general de cómo AWS Proton y otras funciones Servicios de AWS funcionan con la mayoría de las funciones de IAM, consulte Servicios de AWS la Guía del usuario de IAM sobre cómo funcionan con IAM.

Políticas basadas en la identidad para AWS Proton

Compatibilidad con las políticas basadas en identidad	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica

al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en la identidad para AWS Proton

Para ver ejemplos de políticas AWS Proton basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Proton

Políticas basadas en recursos incluidas AWS Proton

Compatibilidad con las políticas basadas en recursos	No

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para más información, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Acciones políticas para AWS Proton

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de AWS Proton acciones, consulta las acciones definidas AWS Proton en la Referencia de autorización del servicio.

Las acciones políticas AWS Proton utilizan el siguiente prefijo antes de la acción:

```
proton
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

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

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `List`, incluya la siguiente acción:

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

Para ver ejemplos de políticas AWS Proton basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Proton

Recursos de políticas para AWS Proton

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

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

Para ver una lista de los tipos de AWS Proton recursos y sus ARN, consulte los recursos definidos AWS Proton en la Referencia de autorización de servicios. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por AWS Proton.

Para ver ejemplos de políticas AWS Proton basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Proton

Claves de condición de la política para AWS Proton

Admite claves de condición de políticas específicas del servicio	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de claves de AWS Proton condición, consulte las claves de condición AWS Proton en la Referencia de autorización de servicio. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS Proton.

Para ver un ejemplo condition-key-based de política para limitar el acceso a un recurso, consulteEjemplos de políticas basadas en condiciones clave para AWS Proton.

Listas de control de acceso (ACL) en AWS Proton

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Las listas de control de acceso (ACL) son listas de beneficiarios que se pueden adjuntar a los recursos. Conceden a las cuentas permisos de acceso al recurso al que están adjuntadas.

Control de acceso basado en atributos (ABAC) con AWS Proton

Admite ABAC (etiquetas en las políticas)	Sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Para obtener más información sobre el etiquetado de AWS Proton recursos, consulte. Recursos y etiquetado de AWS Proton

Uso de credenciales temporales con AWS Proton

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluida la información sobre cuáles Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes

AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Permisos principales entre servicios para AWS Proton

Admite Forward access sessions (FAS)	Sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en él AWS, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Funciones de servicio para AWS Proton

Compatible con roles de servicio	Sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

Para obtener más información, consulte AWS Proton Ejemplos de políticas de funciones de servicio de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir AWS Proton la funcionalidad. Edite las funciones de servicio solo cuando se AWS Proton proporcionen instrucciones para hacerlo.

Funciones vinculadas al servicio para AWS Proton

Compatible con roles vinculados al servicio	Sí

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para obtener más información acerca de cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas para AWS Proton

Encontrará ejemplos de políticas de AWS Proton IAM en las siguientes secciones.

Temas

- Ejemplos de políticas basadas en la identidad para AWS Proton
- AWS Proton Ejemplos de políticas de funciones de servicio de IAM
- Ejemplos de políticas basadas en condiciones clave para AWS Proton

Ejemplos de políticas basadas en la identidad para AWS Proton

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de AWS Proton . Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS la API. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos AWS Proton, incluido el formato de los ARN para cada uno de los tipos de recursos, consulte las claves de condición, recursos y acciones de la Referencia de autorización de servicios. AWS Proton

Temas

- Prácticas recomendadas sobre las políticas
- Enlaces a ejemplos de políticas basadas en la identidad para AWS Proton

Prácticas recomendadas sobre las políticas

Las políticas basadas en la identidad determinan si alguien puede crear AWS Proton recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse

utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Enlaces a ejemplos de políticas basadas en la identidad para AWS Proton

Enlaces a ejemplos de políticas basadas en la identidad para AWS Proton

- AWS políticas gestionadas para AWS Proton
- AWS Proton Ejemplos de políticas de funciones de servicio de IAM
- Ejemplos de políticas basadas en condiciones clave para AWS Proton

AWS Proton Ejemplos de políticas de funciones de servicio de IAM

Los administradores son propietarios y administran los recursos que AWS Proton crean, tal como se definen en las plantillas de servicio y entorno. Adjuntan funciones de servicio de IAM a su cuenta que les AWS Proton permiten crear recursos en su nombre. Los administradores proporcionan las funciones y AWS Key Management Service claves de IAM para los recursos que, posteriormente, son propiedad de los desarrolladores y los administran cuando AWS Proton despliegan su aplicación como un AWS Proton servicio en un entorno. AWS Proton Para obtener más información sobre el cifrado de datos AWS KMS y el cifrado de datos, consulte. Protección de los datos en AWS Proton

Un rol de servicio es un rol de Amazon Web Services (IAM) que permite AWS Proton realizar llamadas a los recursos en su nombre. Si especifica un rol de servicio, AWS Proton utiliza las credenciales de ese rol. Utilice un rol de servicio para especificar de forma explícita las acciones que se AWS Proton pueden realizar.

Puede crear el rol de servicio y su política de permisos con el servicio de IAM. Para obtener más información sobre la creación de un rol de servicio, consulte Crear un rol para delegar permisos a un AWS servicio en la Guía del usuario de IAM.

AWS Proton rol de servicio para el aprovisionamiento mediante AWS CloudFormation

Como miembro del equipo de la plataforma, como administrador, puedes crear una función de AWS Proton servicio y asignársela AWS Proton cuando crees un entorno como función de CloudFormation servicio del entorno (el `protonServiceRoleArn` parámetro de la acción de la CreateEnvironmentAPI). Esta función le permite realizar llamadas AWS Proton a la API a otros servicios en su nombre cuando el entorno o cualquiera de las instancias de servicio que se ejecutan en él utilizan el aprovisionamiento AWS administrado y el aprovisionamiento de AWS CloudFormation la infraestructura.

Le recomendamos que utilice el siguiente rol de IAM y la política de confianza para su AWS Proton rol de servicio. Cuando utiliza la AWS Proton consola para crear un entorno y decide crear un nuevo rol, esta es la política que se AWS Proton suma al rol de servicio que crea para usted. Al determinar el alcance de los permisos de esta política, tenga en cuenta que se produce un AWS Proton error cuando se `Access Denied` producen errores.

> **Important**
>
> Tenga en cuenta que las políticas que se muestran en los siguientes ejemplos otorgan privilegios de administrador a cualquier persona que pueda registrar una plantilla en su cuenta. Como no sabemos qué recursos definirás en tus AWS Proton plantillas, estas políticas tienen permisos amplios. Le recomendamos que restrinja los permisos a los recursos específicos que se implementarán en los entornos.

AWS Proton ejemplo de política de rol de servicio para AWS CloudFormation

*123456789012*Sustitúyalo por tu Cuenta de AWS 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"
          ]
        }
      }
    }
  ]
}
```

AWS Proton política de confianza en el servicio

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Política de funciones de servicio de AWS aprovisionamiento gestionado con un alcance limitado

El siguiente es un ejemplo de una política de funciones de AWS Proton servicio con un ámbito reducido que puede utilizar si solo necesita AWS Proton servicios para aprovisionar recursos de S3.

```
{
  "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 función de servicio para el aprovisionamiento CodeBuild

Como miembro del equipo de la plataforma, como administrador, puedes crear una función de AWS Proton servicio y asignársela AWS Proton cuando crees un entorno como función de CodeBuild servicio del entorno (el `codebuildRoleArn` parámetro de la acción de la CreateEnvironmentAPI).

Esta función le permite realizar llamadas AWS Proton a la API a otros servicios en su nombre cuando el entorno o alguna de las instancias de servicio que se ejecutan en él utilizan el CodeBuild aprovisionamiento para aprovisionar la infraestructura.

Cuando utiliza la AWS Proton consola para crear un entorno y decide crear un rol nuevo, AWS Proton agrega una política con privilegios de administrador al rol de servicio que crea para usted. Al crear su propio rol y reducir los permisos, tenga en cuenta que se produce un AWS Proton error cuando `Access Denied` hay errores.

> **Important**
>
> Ten en cuenta que las políticas AWS Proton asociadas a los roles que crea para ti otorgan privilegios de administrador a cualquier persona que pueda registrar una plantilla en tu cuenta. Como no sabemos qué recursos definirás en tus AWS Proton plantillas, estas políticas tienen amplios permisos. Le recomendamos que restrinja los permisos a los recursos específicos que se implementarán en los entornos.

AWS Proton ejemplo de política de rol de servicio para CodeBuild

El siguiente ejemplo proporciona permisos CodeBuild para aprovisionar recursos mediante AWS Cloud Development Kit (AWS CDK).

*123456789012*Sustitúyalo por tu Cuenta de AWS ID.

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Action": [
       "logs:CreateLogStream",
       "logs:CreateLogGroup",
       "logs:PutLogEvents"
     ],
     "Resource": [
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
Shell-*",
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
Shell-*:*"
     ],
     "Effect": "Allow"
```

```
    },
    {
      "Action": "proton:NotifyResourceDeploymentStatusChange",
      "Resource": "arn:aws:proton:us-east-1:123456789012:*",
      "Effect": "Allow"
    },
    {
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::123456789012:role/cdk-*-deploy-role-*",
        "arn:aws:iam::123456789012:role/cdk-*-file-publishing-role-*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS Proton CodeBuild política de confianza

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "CodeBuildTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS Proton funciones de servicio de canalización

Para aprovisionar canalizaciones de servicios, AWS Proton necesita permisos para realizar llamadas a la API a otros servicios. Los roles de servicio requeridos son similares a los roles de servicio que se

proporcionan al crear entornos. Sin embargo, las funciones para crear canalizaciones se comparten entre todos los servicios de tu AWS cuenta y tú las proporcionas como configuración de la cuenta en la consola o mediante la acción de la UpdateAccountSettingsAPI.

Cuando utilizas la AWS Proton consola para actualizar la configuración de la cuenta y decides crear un nuevo rol para el rol AWS CloudFormation o el de CodeBuild servicio, las políticas que se AWS Proton añaden a los roles de servicio que crea para ti son las mismas que las políticas descritas en las secciones anteriores, Rol de aprovisionamiento administrado por AWS yCodeBuild función de aprovisionamiento. Al determinar el alcance de los permisos de esta política, tenga en cuenta que `Access Denied` los errores AWS Proton fallan.

> **Important**
>
> Tenga en cuenta que las políticas de ejemplo de las secciones anteriores otorgan privilegios de administrador a cualquier persona que pueda registrar una plantilla en su cuenta. Como no sabemos qué recursos definirás en tus AWS Proton plantillas, estas políticas tienen permisos amplios. Le recomendamos que restrinja los permisos a los recursos específicos que se implementarán en las canalizaciones.

AWS Proton rol de componente

Como miembro del equipo de la plataforma, como administrador, puedes crear un rol de AWS Proton servicio y asignárselo AWS Proton al crear un entorno como rol CloudFormation componente del entorno (el `componentRoleArn` parámetro de la acción de la CreateEnvironmentAPI). Este rol restringe la infraestructura que pueden aprovisionar los componentes definidos directamente. Para obtener más información sobre los componentes, consulte Componentes.

El siguiente ejemplo de política admite la creación de un componente definido directamente que aprovisione un bucket de Amazon Simple Storage Service (Amazon S3) y una política de acceso relacionada.

AWS Proton ejemplo de política de rol de componente

*123456789012*Sustitúyalo por tu Cuenta de AWS ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS Proton política de confianza de componentes

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Ejemplos de políticas basadas en condiciones clave para AWS Proton

El siguiente ejemplo de política de IAM deniega el acceso a AWS Proton las acciones que coincidan con las plantillas especificadas en el bloque. `Condition` Tenga en cuenta que estas claves de condición solo son compatibles con las acciones enumeradas en Acciones, recursos y claves de condición de AWS Proton. Para administrar los permisos de otras acciones, como por ejemplo `DeleteEnvironmentTemplate`, debe utilizar el control de acceso a nivel de recursos.

Ejemplo de política que deniega las acciones de una AWS Proton plantilla en una plantilla específica:

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

En el siguiente ejemplo de política, la primera sentencia a nivel de recurso deniega el acceso a las acciones de la AWS Proton plantilla que `ListServiceTemplates` no coincidan con la plantilla de servicio que aparece en el `Resource` bloque. La segunda sentencia deniega el acceso a AWS Proton las acciones que coinciden con la plantilla que aparece en el `Condition` bloque.

Ejemplo de política que deniega AWS Proton las acciones que coinciden con una plantilla específica:

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

El último ejemplo de política permite al desarrollador AWS Proton realizar acciones que coincidan con la plantilla de servicio específica que aparece en el `Condition` bloque.

Ejemplo de política que permite a los AWS Proton desarrolladores realizar acciones que coincidan con una plantilla específica:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListServices",
                "proton:ListServiceInstances",
                "proton:ListEnvironments",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetEnvironment",
                "proton:CreateService",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "proton:DeleteService",
                "codestar-connections:ListConnections"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 "arn:aws:proton:region_id:123456789012:service-template/my-service-template"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        }

    ]
}
```

AWS políticas gestionadas para AWS Proton

Para añadir permisos a usuarios, grupos y roles, es más fácil usar políticas AWS administradas que escribirlas tú mismo. Se necesita tiempo y experiencia para crear políticas administradas por el cliente de IAM que proporcionen a su equipo solo los permisos necesarios. Para empezar rápidamente, puedes usar nuestras políticas AWS gestionadas. Estas políticas cubren casos de uso comunes y están disponibles en su Cuenta de AWS. Para obtener más información sobre las políticas AWS administradas, consulte las políticas AWS administradas en la Guía del usuario de IAM.

Servicios de AWS mantener y actualizar las políticas AWS gestionadas. No puede cambiar los permisos en las políticas AWS gestionadas. En ocasiones, los servicios agregan permisos adicionales a una política administrada por AWS para admitir características nuevas. Este tipo de actualización afecta a todas las identidades (usuarios, grupos y roles) donde se asocia la política. Es más probable que los servicios actualicen una política administrada por AWS cuando se lanza una nueva característica o cuando se ponen a disposición nuevas operaciones. Los servicios no eliminan los permisos de una política AWS administrada, por lo que las actualizaciones de la política no afectarán a los permisos existentes.

Además, AWS admite políticas administradas para funciones laborales que abarcan varios servicios. Por ejemplo, la política ReadOnlyAccess AWS gestionada proporciona acceso de solo lectura a todos Servicios de AWS los recursos. Cuando un servicio lanza una nueva característica, AWS agrega permisos de solo lectura para las operaciones y los recursos nuevos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte Políticas administradas de AWS para funciones de trabajo en la Guía del usuario de IAM.

AWS Proton proporciona políticas de IAM gestionadas y relaciones de confianza que puede asociar a usuarios, grupos o funciones que permiten distintos niveles de control sobre los recursos y las operaciones de la API. Puede aplicar estas políticas directamente o puede usarlas como punto de partida para crear las suyas propias.

La siguiente relación de confianza se utiliza para cada una de las políticas AWS Proton gestionadas.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ExampleTrustRelationshipWithProtonConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS política gestionada: AWSProtonFullAccess

Puede adjuntarla AWSProtonFullAccess a sus entidades de IAM. AWS Proton también vincula esta política a un rol de servicio que le permite AWS Proton realizar acciones en su nombre.

Esta política otorga permisos administrativos que permiten el acceso total a AWS Proton las acciones y el acceso limitado a otras acciones de AWS servicio que AWS Proton dependan de ellas.

La política incluye los siguientes espacios de nombres de acciones clave:

- `proton`: permite a los administradores el acceso completo a las API de AWS Proton .
- `iam`: permite a los administradores transferir roles a AWS Proton. Esto es necesario para AWS Proton poder realizar llamadas a la API a otros servicios en nombre del administrador.
- `kms`: permite a los administradores agregar una concesión a una clave administrada por el cliente.
- `codeconnections`— Permite a los administradores enumerar y transferir conexiones de código para que puedan utilizarlas. AWS Proton

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ProtonPermissions",
            "Effect": "Allow",
            "Action": [
                "proton:*",
                "codestar-connections:ListConnections",
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateGrantPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "proton.*.amazonaws.com"
                }
            }
        },
        {
```

```
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CreateServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
sync.proton.amazonaws.com/AWSServiceRoleForProtonSync",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "sync.proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeStarConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "codeconnections:PassConnection"
            ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codeconnections:PassedToService": "proton.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS política gestionada: AWSProtonDeveloperAccess

Puede adjuntarla AWSProtonDeveloperAccess a sus entidades de IAM. AWS Proton también vincula esta política a un rol de servicio que le permite AWS Proton realizar acciones en su nombre.

Esta política concede permisos que permiten un acceso limitado a AWS Proton las acciones y a otras AWS acciones que AWS Proton dependen de ellas. El alcance de estos permisos está diseñado para respaldar la función de un desarrollador que crea e implementa AWS Proton servicios.

Esta política no proporciona acceso a las API de creación, eliminación y actualización de AWS Proton plantillas y entornos. Si los desarrolladores necesitan permisos aún más limitados que los que ofrece esta política, recomendamos crear una política personalizada que tenga un alcance reducido para conceder el privilegio mínimo.

La política incluye los siguientes espacios de nombres de acciones clave:

- `proton`: permite a los colaboradores acceder a un conjunto limitado de API de AWS Proton .
- `codeconnections`— Permite a los colaboradores enumerar y pasar conexiones de código para que puedan ser utilizadas por AWS Proton ellos.

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ProtonPermissions",
            "Effect": "Allow",
            "Action": [
                "codecommit:ListRepositories",
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineExecution",
                "codepipeline:GetPipelineState",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListPipelines",
                "codestar-connections:ListConnections",
                "codestar-connections:UseConnection",
                "proton:CancelServiceInstanceDeployment",
                "proton:CancelServicePipelineDeployment",
                "proton:CreateService",
                "proton:DeleteService",
                "proton:GetAccountRoles",
                "proton:GetAccountSettings",
                "proton:GetEnvironment",
                "proton:GetEnvironmentAccountConnection",
                "proton:GetEnvironmentTemplate",
                "proton:GetEnvironmentTemplateMajorVersion",
                "proton:GetEnvironmentTemplateMinorVersion",
                "proton:GetEnvironmentTemplateVersion",
                "proton:GetRepository",
                "proton:GetRepositorySyncStatus",
                "proton:GetResourcesSummary",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateMajorVersion",
                "proton:GetServiceTemplateMinorVersion",
                "proton:GetServiceTemplateVersion",
                "proton:GetTemplateSyncConfig",
                "proton:GetTemplateSyncStatus",
                "proton:ListEnvironmentAccountConnections",
                "proton:ListEnvironmentOutputs",
                "proton:ListEnvironmentProvisionedResources",
                "proton:ListEnvironments",
                "proton:ListEnvironmentTemplateMajorVersions",
                "proton:ListEnvironmentTemplateMinorVersions",
                "proton:ListEnvironmentTemplates",
                "proton:ListEnvironmentTemplateVersions",
```

```
                "proton:ListRepositories",
                "proton:ListRepositorySyncDefinitions",
                "proton:ListServiceInstanceOutputs",
                "proton:ListServiceInstanceProvisionedResources",
                "proton:ListServiceInstances",
                "proton:ListServicePipelineOutputs",
                "proton:ListServicePipelineProvisionedResources",
                "proton:ListServices",
                "proton:ListServiceTemplateMajorVersions",
                "proton:ListServiceTemplateMinorVersions",
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListTagsForResource",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "s3:ListAllMyBuckets",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsPermissions",
            "Effect": "Allow",
            "Action": "codestar-connections:PassConnection",
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
            "Action": "codeconnections:PassConnection",
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "codeconnections:PassedToService": "proton.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS política gestionada: AWSProtonReadOnlyAccess

Puede adjuntarla AWSProtonReadOnlyAccess a sus entidades de IAM. AWS Proton también vincula esta política a un rol de servicio que le permite AWS Proton realizar acciones en su nombre.

Esta política concede permisos que permiten el acceso de solo lectura a AWS Proton las acciones y un acceso limitado de solo lectura a otras acciones del AWS servicio que dependan de ellas. AWS Proton

La política incluye los siguientes espacios de nombres de acciones clave:

- `proton`: permite a los colaboradores acceso de solo lectura a las API de AWS Proton .

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListPipelines",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "proton:GetAccountRoles",
        "proton:GetAccountSettings",
        "proton:GetEnvironment",
        "proton:GetEnvironmentAccountConnection",
        "proton:GetEnvironmentTemplate",
        "proton:GetEnvironmentTemplateMajorVersion",
```

```
        "proton:GetEnvironmentTemplateMinorVersion",
        "proton:GetEnvironmentTemplateVersion",
        "proton:GetRepository",
        "proton:GetRepositorySyncStatus",
        "proton:GetResourcesSummary",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetServiceTemplateVersion",
        "proton:GetTemplateSyncConfig",
        "proton:GetTemplateSyncStatus",
        "proton:ListEnvironmentAccountConnections",
        "proton:ListEnvironmentOutputs",
        "proton:ListEnvironmentProvisionedResources",
        "proton:ListEnvironments",
        "proton:ListEnvironmentTemplateMajorVersions",
        "proton:ListEnvironmentTemplateMinorVersions",
        "proton:ListEnvironmentTemplates",
        "proton:ListEnvironmentTemplateVersions",
        "proton:ListRepositories",
        "proton:ListRepositorySyncDefinitions",
        "proton:ListServiceInstanceOutputs",
        "proton:ListServiceInstanceProvisionedResources",
        "proton:ListServiceInstances",
        "proton:ListServicePipelineOutputs",
        "proton:ListServicePipelineProvisionedResources",
        "proton:ListServices",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateVersions",
        "proton:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS política gestionada: AWSProtonSyncServiceRolePolicy

AWS Proton adjunta esta política a la función AWSServiceRoleForProtonSync vinculada al servicio que permite realizar la sincronización de AWS Proton plantillas.

Esta política concede permisos que permiten un acceso limitado a AWS Proton las acciones y a otras acciones de AWS servicio que AWS Proton dependan de ellas.

La política incluye los siguientes espacios de nombres de acciones clave:

- `proton`— Permite AWS Proton sincronizar el acceso limitado a AWS Proton las API.
- `codeconnections`— Permite AWS Proton sincronizar el acceso limitado a CodeConnections las API.

Para obtener información sobre los detalles de los permisos AWSProtonSyncServiceRolePolicy, consulte Permisos de roles vinculados al servicio para. AWS Proton

AWS política gestionada: AWSProtonCodeBuildProvisioningBasicAccess

Los permisos CodeBuild deben ejecutar una compilación para el AWS Proton CodeBuild aprovisionamiento. Puede asignarlo `AWSProtonCodeBuildProvisioningBasicAccess` a su función de CodeBuild aprovisionamiento.

Esta política otorga los permisos mínimos para que AWS Proton CodeBuild Provisioning funcione. Otorga permisos que permiten CodeBuild generar registros de compilación. También otorga permiso a Proton para poner a disposición de los usuarios las salidas de Infraestructura como Código (IaC). AWS Proton No proporciona los permisos que necesitan las herramientas de IaC para administrar la infraestructura.

La política incluye los siguientes espacios de nombres de acciones clave:

- `logs`- Permite CodeBuild generar registros de construcción. Sin este permiso, no CodeBuild podrá iniciarse.
- `proton`- Permite que un comando de CodeBuild aprovisionamiento solicite la actualización de los resultados de la IaaC de un recurso determinado AWS Proton . `aws proton notify-resource-deployment-status-change`

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents"
   ],
   "Resource": [
    "arn:aws:logs:*:*:log-group:/aws/codebuild/AWSProton-*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "proton:NotifyResourceDeploymentStatusChange",
   "Resource": "arn:aws:proton:*:*:*"
  }
 ]
}
```

AWS política gestionada: AWSProtonCodeBuildProvisioningServiceRolePolicy

AWS Proton asocia esta política a la función AWSServiceRoleForProtonCodeBuildProvisioning vinculada al servicio que permite realizar CodeBuild el aprovisionamiento AWS Proton basado.

Esta política concede permisos que permiten un acceso limitado a las acciones del AWS servicio en función de ellas. AWS Proton

La política incluye los siguientes espacios de nombres de acciones clave:

- `cloudformation`— Permite el aprovisionamiento AWS Proton CodeBuild basado en un acceso limitado a AWS CloudFormation las API.
- `codebuild`— Permite el aprovisionamiento AWS Proton CodeBuild basado y el acceso limitado a las CodeBuild API.
- `iam`: permite a los administradores transferir roles a AWS Proton. Esto es necesario para AWS Proton poder realizar llamadas a la API a otros servicios en nombre del administrador.

- `servicequotas`— Permite AWS Proton comprobar el límite de compilaciones CodeBuild simultáneas, lo que garantiza una correcta puesta en cola de creación.

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
```

```
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
          "servicequotas:GetServiceQuota"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS política gestionada: AwsProtonServiceGitSyncServiceRolePolicy

AWS Proton asocia esta política a la función AwsProtonServiceGitSyncServiceRolePolicy vinculada al servicio que permite AWS Proton realizar la sincronización del servicio.

Esta política concede permisos que permiten un acceso limitado a AWS Proton las acciones y a otras acciones de AWS servicio que AWS Proton dependan de ellas.

La política incluye los siguientes espacios de nombres de acciones clave:

- `proton`: permite a AWS Proton sincronizar el acceso limitado a las API de AWS Proton.

Para obtener información sobre los detalles de los permisos AwsProtonServiceGitSyncServiceRolePolicy, consulte Permisos de roles vinculados al servicio para. AWS Proton

AWS Proton actualizaciones de las políticas gestionadas AWS

Consulte los detalles sobre las actualizaciones de las políticas AWS administradas AWS Proton desde que este servicio comenzó a realizar el seguimiento de estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS de la página del historial del AWS Proton documento.

Cambio	Descripción	Fecha
AWSProtonFullAccess: actualización de una política actual	Se ha actualizado la política gestionada para que el rol vinculado al servicio utilice la sincronización de Git con los repositorios de Git para los recursos con ambos prefijos de servicio. Para obtener más información, consulte Uso de roles vinculados a servicios para AWS CodeConnections y políticas administradas.	25 de abril de 2024
AWSProtonDeveloperAccess: actualización de una política actual	Se ha actualizado la política gestionada para que el rol vinculado al servicio utilice la sincronización de Git con los repositorios de Git para los recursos con ambos prefijos de servicio. Para obtener más información, consulte Uso de roles vinculados a servicios para AWS CodeConnections y políticas administradas.	25 de abril de 2024
AWSProtonSyncServiceRolePolicy: actualización de una política actual	Se ha actualizado la política gestionada para que el rol vinculado al servicio utilice la sincronización de Git con los repositorios de Git para los recursos con ambos prefijos de servicio. Para obtener más información, consulte Uso de roles vinculados a servicios	25 de abril de 2024

Cambio	Descripción	Fecha
	para AWS CodeConnections y políticas administradas.	
AWSProtonCodeBuildProvisioningServiceRolePolicy: actualización de una política actual	AWS Proton actualizó esta política para añadir permisos y garantizar que las cuentas tengan el límite de creación CodeBuild simultánea necesario para poder utilizar Provisioning. CodeBuild	12 de mayo de 2023
AwsProtonServiceGitSyncServiceRolePolicy: política nueva	AWS Proton agregó una nueva política para permitir la sincronización AWS Proton de los servicios. La política se usa en la función vinculada al AWSServiceRoleForProtonServiceSyncservicio.	31 de marzo de 2023
AWSProtonDeveloperAccess: actualización de una política actual	AWS Proton agregó una nueva `GetResourcesSummary` acción que le permite ver un resumen de sus plantillas, los recursos de plantillas implementados y los recursos desactualizados.	18 de noviembre de 2022
AWSProtonReadOnlyAccess: actualización de una política actual	AWS Proton agregó una nueva `GetResourcesSummary` acción que le permite ver un resumen de sus plantillas, los recursos de plantillas implementados y los recursos desactualizados.	18 de noviembre de 2022

Cambio	Descripción	Fecha
AWSProtonCodeBuildProvisioningBasicAccess: política nueva	AWS Proton agregó una nueva política que otorga CodeBuild los permisos necesarios para ejecutar una compilación para AWS Proton CodeBuild Provisioning.	16 de noviembre de 2022
AWSProtonCodeBuildProvisioningServiceRolePolicy: política nueva	AWS Proton agregó una nueva política que permite AWS Proton realizar operaciones relacionadas con el aprovisionamiento CodeBuild basado. La política se usa en la función vinculada al AWSServiceRoleForProtonCodeBuildProvisioningservicio.	2 de septiembre de 2022
AWSProtonFullAccess: actualización de una política actual	AWS Proton actualizó esta política para proporcionar acceso a las nuevas operaciones de la AWS Proton API y corregir los problemas de permisos en algunas operaciones de la AWS Proton consola.	30 de marzo de 2022
AWSProtonDeveloperAccess: actualización de una política actual	AWS Proton actualiza esta política para proporcionar acceso a las nuevas operaciones de la AWS Proton API y solucionar los problemas de permisos de algunas operaciones de AWS Proton la consola.	30 de marzo de 2022

Cambio	Descripción	Fecha
AWSProtonReadOnlyAccess: actualización de una política actual	AWS Proton actualice esta política para proporcio nar acceso a las nuevas operaciones de la AWS Proton API y solucionar los problemas de permisos de algunas operaciones de AWS Proton la consola.	30 de marzo de 2022
AWSProtonSyncServi ceRolePolicy: política nueva	AWS Proton agregó una nueva política que permite AWS Proton realizar operaciones relacionadas con la sincronización de plantillas. La política se usa en la función AWSServic eRoleForProtonSyncvinculada al servicio.	23 de noviembre de 2021
AWSProtonFullAccess: política nueva	AWS Proton se agregó una nueva política para proporcionar acceso a los roles administrativos a las operaciones de la AWS Proton API y a la AWS Proton consola.	9 de junio de 2021
AWSProtonDeveloperAccess: política nueva	AWS Proton se agregó una nueva política para proporcio nar acceso a las operaciones de la AWS Proton API y a la AWS Proton consola desde el rol de desarrollador.	9 de junio de 2021

Cambio	Descripción	Fecha
AWSProtonReadOnlyAccess: política nueva	AWS Proton se agregó una nueva política para proporcio nar acceso de solo lectura a las operaciones de la AWS Proton API y a la AWS Proton consola.	9 de junio de 2021
AWS Proton comenzó a rastrear los cambios.	AWS Proton comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas.	9 de junio de 2021

Uso de roles vinculados a servicios de AWS Proton

AWS Proton usa roles vinculados al AWS Identity and Access Management servicio (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM al que se vincula directamente. AWS Proton Los roles vinculados al servicio están predefinidos AWS Proton e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en su nombre.

Temas

- Uso de roles para la sincronización AWS Proton
- Uso de roles para el aprovisionamiento basado CodeBuild

Uso de roles para la sincronización AWS Proton

AWS Proton usa roles vinculados al AWS Identity and Access Management servicio (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM al que se vincula directamente. AWS Proton Los roles vinculados al servicio están predefinidos AWS Proton e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en su nombre.

Un rol vinculado a un servicio facilita la configuración AWS Proton , ya que no es necesario añadir manualmente los permisos necesarios. AWS Proton define los permisos de sus funciones vinculadas al servicio y, a menos que se defina lo contrario, solo AWS Proton puede asumir sus funciones. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda adjuntar a ninguna otra entidad de IAM.

Solo es posible eliminar un rol vinculado a un servicio después de eliminar sus recursos relacionados. Esto protege sus AWS Proton recursos porque no puede eliminar inadvertidamente el permiso de acceso a los recursos.

Para obtener información sobre otros servicios que admiten roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Roles vinculados a servicios. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de rol vinculados al servicio para AWS Proton

AWS Proton utiliza dos funciones vinculadas al servicio denominadas y. AWSServiceRoleForProtonSyncAWSServiceRoleForProtonServiceSync

El rol AWSServiceRoleForProtonSync vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `sync.proton.amazonaws.com`

La política de permisos de roles denominada AWSProtonSyncServiceRolePolicy permite AWS Proton realizar las siguientes acciones en los recursos especificados:

- Acción: crear, administrar y leer en plantillas y versiones de plantillas de AWS Proton
- Acción: utilizar la conexión en CodeConnections

AWSProtonSyncServiceRolePolicy

Esta política incluye los permisos siguientes:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SyncToProton",
            "Effect": "Allow",
            "Action": [
                "proton:UpdateServiceTemplateVersion",
                "proton:UpdateServiceTemplate",
                "proton:UpdateEnvironmentTemplateVersion",
                "proton:UpdateEnvironmentTemplate",
```

```
                "proton:GetServiceTemplateVersion",
                "proton:GetServiceTemplate",
                "proton:GetEnvironmentTemplateVersion",
                "proton:GetEnvironmentTemplate",
                "proton:DeleteServiceTemplateVersion",
                "proton:DeleteEnvironmentTemplateVersion",
                "proton:CreateServiceTemplateVersion",
                "proton:CreateServiceTemplate",
                "proton:CreateEnvironmentTemplateVersion",
                "proton:CreateEnvironmentTemplate",
                "proton:ListEnvironmentTemplateVersions",
                "proton:ListServiceTemplateVersions",
                "proton:CreateEnvironmentTemplateMajorVersion",
                "proton:CreateServiceTemplateMajorVersion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AccessGitRepos",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:UseConnection",
                "codeconnections:UseConnection"
             ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ]
        }
    ]
}
```

Para obtener información sobre la AWSProtonSyncServiceRolePolicy, consulte la política AWS administrada: AWSProtonSyncServiceRolePolicy.

El rol AWSServiceRoleForProtonServiceSync vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `service-sync.proton.amazonaws.com`

La política de permisos de roles denominada AWSServiceRoleForProtonServiceSync permite AWS Proton realizar las siguientes acciones en los recursos especificados:

- Acción: crear, administrar y leer AWS Proton servicios e instancias de servicio

AwsProtonServiceGitSyncServiceRolePolicy

Esta política incluye los permisos siguientes:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "ProtonServiceSync",
   "Effect": "Allow",
   "Action": [
    "proton:GetService",
    "proton:UpdateService",
    "proton:UpdateServicePipeline",
    "proton:CreateServiceInstance",
    "proton:GetServiceInstance",
    "proton:UpdateServiceInstance",
    "proton:ListServiceInstances",
    "proton:GetComponent",
    "proton:CreateComponent",
    "proton:ListComponents",
    "proton:UpdateComponent",
    "proton:GetEnvironment",
    "proton:CreateEnvironment",
    "proton:ListEnvironments",
    "proton:UpdateEnvironment"
   ],
   "Resource": "*"
  }
 ]
}
```

Para obtener información sobre la AwsProtonServiceSyncServiceRolePolicy, consulte la política AWS administrada: AwsProtonServiceSyncServiceRolePolicy.

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio de AWS Proton

No necesita crear manualmente un rol vinculado a servicios. Cuando configuras un repositorio o servicio para sincronizarlo AWS Proton en la AWS Management Console AWS CLI, la o la AWS API, AWS Proton crea el rol vinculado al servicio para ti.

Si elimina este rol vinculado a servicios y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al configurar un repositorio o servicio para sincronizarlo AWS Proton, vuelve a AWS Proton crear el rol vinculado al servicio automáticamente.

Para volver a crear el rol AWSServiceRoleForProtonSync vinculado al servicio, querrá configurar un repositorio para la sincronización y, para volver a crearlo AWSServiceRoleForProtonServiceSync, querrá configurar un servicio para la sincronización.

Modificación de un rol vinculado a servicios de AWS Proton

AWS Proton no le permite editar el rol vinculado al servicio. AWSServiceRoleForProtonSync Después de crear un rol vinculado a un servicio, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia a él. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio de AWS Proton

No es necesario eliminar el rol manualmente. AWSServiceRoleForProtonSync Al eliminar todos los repositorios AWS Proton enlazados para sincronizarlos en la AWS Management Console AWS CLI, la API o la AWS API, se AWS Proton limpian los recursos y se elimina automáticamente la función vinculada al servicio.

Regiones admitidas para los roles vinculados a un servicio de AWS Proton

AWS Proton admite el uso de funciones vinculadas al servicio en todos los lugares en los que el servicio esté disponible. Regiones de AWS Para obtener más información, consulte Puntos de conexión y cuotas de AWS Proton en la Referencia general de AWS.

Uso de roles para el aprovisionamiento basado CodeBuild

AWS Proton usa roles vinculados al AWS Identity and Access Management servicio (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM al que se vincula directamente. AWS Proton Los roles vinculados al servicio están predefinidos AWS Proton e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en su nombre.

Un rol vinculado a un servicio facilita la configuración AWS Proton , ya que no es necesario añadir manualmente los permisos necesarios. AWS Proton define los permisos de sus funciones vinculadas al servicio y, a menos que se defina lo contrario, solo AWS Proton puede asumir sus funciones. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda adjuntar a ninguna otra entidad de IAM.

Solo es posible eliminar un rol vinculado a un servicio después de eliminar sus recursos relacionados. Esto protege sus AWS Proton recursos porque no puede eliminar inadvertidamente el permiso de acceso a los recursos.

Para obtener información sobre otros servicios que admiten roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Roles vinculados a servicios. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de rol vinculados al servicio para AWS Proton

AWS Proton utiliza el rol vinculado al servicio denominado «AWSServiceRoleForProtonCodeBuildProvisioningUn rol vinculado al servicio» para el aprovisionamiento. AWS Proton CodeBuild

El rol AWSServiceRoleForProtonCodeBuildProvisioning vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `codebuild.proton.amazonaws.com`

La política de permisos de roles denominada AWSProtonCodeBuildProvisioningServiceRolePolicy permite AWS Proton realizar las siguientes acciones en los recursos especificados:

- Acción: crear, administrar y leer en transformaciones y pilas de AWS CloudFormation
- Acción: crear, gestionar y leer CodeBuild proyectos y compilaciones

AWSProtonCodeBuildProvisioningServiceRolePolicy

Esta política incluye los permisos siguientes:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
          "servicequotas:GetServiceQuota"
      ],
```

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

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio de AWS Proton

No necesita crear manualmente un rol vinculado a servicios. Al crear un entorno que utiliza el aprovisionamiento CodeBuild basado AWS Proton en la AWS Management Console, la o la AWS API AWS CLI, se AWS Proton crea automáticamente la función vinculada al servicio.

Si elimina este rol vinculado a servicios y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Cuando crea un entorno que utiliza el aprovisionamiento CodeBuild basado en AWS Proton, vuelve a crear el rol AWS Proton vinculado al servicio para usted.

Modificación de un rol vinculado a servicios de AWS Proton

AWS Proton no le permite editar el rol vinculado al AWSServiceRoleForProtonCodeBuildProvisioning servicio. Después de crear un rol vinculado al servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al rol. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio de AWS Proton

Si ya no necesita usar una característica o servicio que requieran un rol vinculado a un servicio, le recomendamos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se monitoree ni mantenga de forma activa. Sin embargo, debe eliminar todos los entornos y servicios (instancias y canalizaciones) que utilizan el aprovisionamiento CodeBuild basado para AWS Proton poder eliminarlos manualmente.

Eliminación manual de un rol vinculado a servicios

Utilice la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSServiceRoleForProtonCodeBuildProvisioning servicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones admitidas para los roles vinculados a un servicio de AWS Proton

AWS Proton admite el uso de roles vinculados al servicio en todos los Regiones de AWS lugares en los que el servicio esté disponible. Para obtener más información, consulte Puntos de conexión y cuotas de AWS Proton en la Referencia general de AWS.

Solución de problemas de AWS Proton identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas habituales que pueden surgir al trabajar con un AWS Proton IAM.

Temas

- No estoy autorizado a realizar ninguna acción en AWS Proton
- No estoy autorizado a realizar lo siguiente: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS Proton recursos

No estoy autorizado a realizar ninguna acción en AWS Proton

Si AWS Management Console le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `proton:`*`GetWidget`*.

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

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`my-example-widget`* mediante la acción `proton:`*`GetWidget`*.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS Proton.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en AWS Proton. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

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

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS Proton recursos

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si AWS Proton es compatible con estas funciones, consulte¿Cómo AWS Proton funciona con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.

- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Configuración y análisis de vulnerabilidades en AWS Proton

AWS Proton no proporciona parches ni actualizaciones para el código que proporciona el cliente. Los clientes son responsables de actualizar y aplicar los parches a su propio código, incluido el código fuente de sus servicios y aplicaciones que se ejecutan en AWS Proton y el código incluido en sus paquetes de plantillas tanto de servicio como de entorno.

Los clientes son responsables de actualizar y aplicar parches a los recursos de infraestructura de sus entornos y servicios. AWS Proton no actualizará ni parcheará automáticamente ningún recurso. Los clientes deben consultar la documentación de los recursos de su arquitectura para comprender sus respectivas políticas de aplicación de parches.

Aparte de proporcionar las actualizaciones del entorno y del servicio que soliciten los clientes para las versiones secundarias de las plantillas de servicios y de entornos, AWS Proton no proporciona parches ni actualizaciones para los recursos que los clientes definan en sus plantillas y paquetes de plantillas de servicios y de entornos.

Para obtener más detalles, consulte los siguientes recursos:

- Modelo de responsabilidad compartida
- Amazon Web Services: Overview of Security Processes

Protección de los datos en AWS Proton

AWS Proton cumple el modelo de responsabilidad compartida de los AWS, que incluye reglamentos y directrices para la protección de los datos. AWS es responsable de proteger la infraestructura global que ejecuta todos los Servicios de AWS. AWS mantiene el control de los datos alojados en esta infraestructura, incluidos los controles de configuración de la seguridad para el tratamiento del contenido y los datos personales de los clientes. Los clientes de AWS y los socios de APN,

que actúan como controladores o procesadores de datos, son responsables de todos los datos personales que colocan en la Nube de AWS.

Para fines de protección de datos, le recomendamos proteger las credenciales de Cuenta de AWS y configurar cuentas de usuario individuales con AWS Identity and Access Management (IAM), de modo que a cada usuario se le concedan únicamente los permisos necesarios para llevar a cabo su trabajo. También recomendamos proteger sus datos de las siguientes formas:

- Utilice la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. Recomendamos TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de Servicios de AWS.

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de texto de formato libre, como el campo Nombre. Incluye las situaciones en las que debe trabajar con la AWS Proton u otros Servicios de AWS a través de la consola, la API, la AWS CLI o los SDK de AWS. Es posible que cualquier dato que introduzca en los campos de texto de formato libre para los identificadores de recursos o elementos similares relacionados con la administración de los recursos de AWS se incluya en los registros de diagnóstico. Cuando proporcione una URL a un servidor externo, no incluya información de credenciales en la URL para validar la solicitud para ese servidor.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al modelo de responsabilidad compartida de AWS y GDPR en el blog de seguridad de AWS.

Cifrado del lado del servidor en reposo

Si el usuario decide cifrar los datos confidenciales de los paquetes de plantillas en resposo en el bucket de S3 en el que se almacenan los paquetes de plantillas, deberá utilizar una clave SSE-S3 o SSE-KMS para permitir que AWS Proton recupere los paquetes de plantillas con el fin de poder asociarlos a una plantilla registrada de AWS Proton.

Cifrado en tránsito

Toda la comunicación de un servicio a otro se cifra en tránsito mediante SSL/TLS.

Administración de claves de cifrado de AWS Proton

En AWS Proton, todos los datos de los clientes se cifran de forma predeterminada mediante una clave propia de AWS Proton. Si proporciona una clave de AWS KMS administrada por el cliente y propiedad del cliente, todos los datos del cliente se cifrarán con la clave proporcionada por el cliente, tal y como se describe en los párrafos siguientes.

Al crear una plantilla de AWS Proton, se especifica la clave y AWS Proton utiliza las credenciales para crear una autorización que permita a AWS Proton utilizar dicha clave.

Si retira la concesión manualmente o deshabilita o elimina la clave especificada, AWS Proton no podrá leer los datos cifrados por la clave especificada y arrojará `ValidationException`.

Contexto de cifrado de AWS Proton

AWS Proton admite encabezados de contexto de cifrado. Un contexto de cifrado es un conjunto opcional de pares clave-valor que pueden contener información contextual adicional sobre los datos. Para obtener información general sobre el contexto de cifrado, consulte Conceptos de AWS Key Management Service: contexto de cifrado en la Guía para desarrolladores de AWS Key Management Service.

Un contexto de cifrado es un conjunto de pares de clave-valor que contienen datos no secretos arbitrarios. Al incluir un contexto de cifrado en una solicitud para cifrar datos, AWS KMS vincula criptográficamente el contexto de cifrado a los datos cifrados. Para descifrar los datos, es necesario pasar el mismo contexto de cifrado.

Los clientes pueden utilizar el contexto de cifrado para identificar el uso de las claves administradas por el cliente en los registros de auditoría y en los registros. También aparece en texto no cifrado en los registros, como AWS CloudTrail y Registros de Amazon CloudWatch.

AWS Proton no incluye ningún contexto de cifrado especificado externamente o por el cliente.

AWS Proton agrega el siguiente contexto de cifrado.

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

El primer contexto de cifrado identifica la plantilla de AWS Proton a la que está asociado el recurso y también sirve como restricción para los permisos y la concesión de claves administradas por el cliente.

El segundo contexto de cifrado identifica el recurso de AWS Proton que está cifrado.

Los siguientes ejemplos muestran el uso del contexto de cifrado en AWS Proton.

Un desarrollador que crea una instancia de servicio.

```
{
  "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"
}
```

Un administrador que crea una plantilla.

```
{
  "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"
}
```

Seguridad de la infraestructura en AWS Proton

Como servicio gestionado, AWS Proton está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a AWS Proton través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.

- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Para mejorar el aislamiento de la red, puede AWS PrivateLink utilizarlas como se describe en la siguiente sección.

AWS Proton y puntos finales de VPC de interfaz ()AWS PrivateLink

Puede establecer una conexión privada entre su VPC y crear un punto final de AWS Proton la VPC de interfaz. Los puntos finales de la interfaz funcionan con una tecnología que le permite acceder de forma privada a AWS Proton las API sin una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. AWS PrivateLink AWS Direct Connect Las instancias de su VPC no necesitan direcciones IP públicas para comunicarse con AWS Proton las API. El tráfico entre tu VPC y AWS Proton no sale de la red de Amazon.

Cada punto de conexión de la interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte Puntos de conexión de VPC de interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones sobre los puntos AWS Proton finales de VPC

Antes de configurar un punto de enlace de VPC de interfaz AWS Proton, asegúrese de revisar las propiedades y limitaciones del punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

AWS Proton admite realizar llamadas a todas sus acciones de API desde su VPC.

Se admiten las políticas de puntos finales de VPC. AWS Proton De forma predeterminada, AWS Proton se permite el acceso total a través del punto final. Para más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

Creación de un punto final de VPC de interfaz para AWS Proton

Puede crear un punto de enlace de VPC para el AWS Proton servicio mediante la consola de Amazon VPC o el (). AWS Command Line Interface AWS CLI Para más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto final de VPC para AWS Proton usar el siguiente nombre de servicio:

- com.amazonaws.*region*.proton

Si habilitas el DNS privado para el punto final, puedes realizar solicitudes a la API para AWS Proton utilizar su nombre de DNS predeterminado para la región, por ejemplo. `proton.region.amazonaws.com`

Para más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Crear una política de puntos de conexión de VPC para AWS Proton

Puede asociar una política de punto de conexión con su punto de conexión de VPC que controla el acceso a AWS Proton. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

Ejemplo: política de puntos finales de VPC para acciones AWS Proton

El siguiente es un ejemplo de una política de puntos finales para AWS Proton. Cuando se adjunta a un punto final, esta política otorga acceso a las AWS Proton acciones enumeradas a todos los principales de todos los recursos.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Principal": "*",
      "Action": [
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServices",
        "proton:ListServiceInstances",
        "proton:ListEnvironments",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetEnvironment",
        "proton:CreateService",
        "proton:UpdateService",
        "proton:UpdateServiceInstance",
        "proton:UpdateServicePipeline",
        "proton:DeleteService"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Registro y monitoreo en AWS Proton

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS Proton y de sus otras soluciones de AWS. AWS ofrece las siguientes herramientas de monitorización para vigilar las instancias que se estén ejecutando en AWS Proton, informar cuando algo no va bien y tomar medidas automáticamente cuando proceda.

En este momento, AWS Proton no está integrado con Registros de Amazon CloudWatch o AWS Trusted Advisor. Los administradores pueden configurar y utilizar CloudWatch para monitorear otros Servicios de AWS según se defina en sus plantillas de servicio y de entorno. AWS Proton está integrado con AWS CloudTrail.

- Amazon CloudWatch monitorea los recursos de AWS y las aplicaciones que ejecuta en AWS en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles

personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puede hacer que CloudWatch haga un seguimiento del uso de la CPU u otras métricas de las instancias de Amazon EC2 y lanzar nuevas instancias automáticamente cuando sea necesario. Para obtener más información, consulte la Guía del usuario de Amazon CloudWatch.

- Registros de Amazon CloudWatch le permite monitorear, almacenar y tener acceso a los archivos de registro desde instancias de Amazon EC2, CloudTrail u otras fuentes. CloudWatch Logs puede monitorear información en los registros y enviarle una notificación cuando se llega a determinados umbrales. También se pueden archivar los datos de los registros en un almacenamiento de larga duración. Para obtener más información, consulte la Guía del usuario de Amazon CloudWatch Logs.
- AWS CloudTrail captura las llamadas a la API y otros eventos relacionados que realiza la Cuenta de AWS o se realizan en nombre de esta. Además, entrega los archivos de registros a un bucket de Amazon S3 especificado. También pueden identificar qué usuarios y cuentas llamaron a AWS, la dirección IP de origen de las llamadas y el momento en que se hicieron. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.
- Amazon EventBridge: es un bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de varios orígenes. EventBridge no solo proporciona un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones de software como servicio (SaaS) y Servicios de AWS, sino que dirige dichos datos a destinos como Lambda. Esto le permite monitorear los eventos que ocurren en los servicios y crear arquitecturas basadas en eventos. Para obtener más información, consulte Automatice AWS Proton con EventBridge y la Guía del usuario de EventBridge.

Resiliencia en AWS Proton

La infraestructura global de AWS se divide en Región de AWS y zonas de disponibilidad. Las Región de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las Región de AWS y las zonas de disponibilidad, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, AWS Proton ofrece características que le ayudan con sus necesidades de resiliencia y copia de seguridad de los datos.

Copias de seguridad de AWS Proton

AWS Proton mantiene una copia de seguridad de todos los datos de los clientes. En el caso de una interrupción total, esta copia de seguridad se puede utilizar para restaurar AWS Proton y los datos de los clientes a partir de un estado válido anterior.

Prácticas recomendadas de seguridad para AWS Proton

AWS Proton proporciona varias características de seguridad para tener en cuenta a medida que el usuario vaya desarrollando e implementando sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no suponen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

Temas

Utilice IAM para controlar el acceso

IAM es un Servicio de AWS que se puede utilizar para administrar a los usuarios y sus permisos en AWS. Puede utilizar IAM con AWS Proton para especificar qué acciones de AWS Proton pueden llevar a cabo los administradores y desarrolladores, como, por ejemplo, administrar plantillas, entornos o servicios. Puede utilizar roles de servicio de IAM para permitir a AWS Proton hacer llamadas a otros servicios en su nombre.

Para obtener más información sobre AWS Proton y los roles de IAM, consulte Identity and Access Management para AWS Proton.

Implemente el acceso a los privilegios mínimos. Para obtener más información, consulte Políticas y permisos en IAM en la Guía del usuario de AWS Identity and Access Management.

No integre credenciales en sus plantillas ni en sus paquetes de plantillas

En lugar de integrar información confidencial en sus plantillas y paquetes de plantillas de AWS CloudFormation, le recomendamos que utilice referencias dinámicas en su plantilla de pila.

Las referencias dinámicas son una forma eficaz y coherente de hacer referencia a valores externos almacenados y administrados en otros servicios, como AWS Systems Manager Parameter Store o AWS Secrets Manager. Cuando se utiliza una referencia dinámica, CloudFormation recupera el valor de la referencia especificada cuando es necesario durante las operaciones de pila y de conjunto de cambios, y pasa el valor al recurso correspondiente. Sin embargo, CloudFormation no almacena nunca el valor de referencia real. Para obtener más información, consulte Uso de referencias dinámicas para especificar valores de plantillas en la Guía del usuario de AWS CloudFormation.

AWS Secrets Manager le ayuda a cifrar, almacenar y recuperar de forma segura las credenciales de las bases de datos y otros servicios. AWS Systems Manager Parameter Store proporciona un almacenamiento seguro y jerárquico para administrar los datos de configuración.

Para obtener más información acerca de cómo definir parámetros en plantillas, consulte https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html en la Guía del usuario de AWS CloudFormation.

Uso del cifrado para proteger la información confidencial

En AWS Proton, todos los datos de los clientes se cifran de forma predeterminada mediante una clave propia de AWS Proton.

Como miembro del equipo de la plataforma, puede proporcionar una clave administrada por el cliente a AWS Proton para cifrar y proteger su información confidencial. Cifre los datos en reposo confidenciales almacenados en su bucket de S3. Para obtener más información, consulte Protección de los datos en AWS Proton.

Uso de AWS CloudTrail para ver y registrar las llamadas a la API

AWS CloudTrail realiza un seguimiento a cualquier persona que realice llamadas a la API en su Cuenta de AWS. Las llamadas a la API se registran cuando se utilice la API de AWS Proton, la consola de AWS Proton o cualquier comando de la AWS Proton AWS CLI. Active el registro y especifique un bucket de Amazon S3 para almacenar los registros. De ese modo, si lo necesita, puede auditar quién hizo qué llamada a AWS Proton en su cuenta. Para obtener más información, consulte Registro y monitoreo en AWS Proton.

Prevención del suplente confuso entre servicios

El problema del suplente confuso es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema del suplente confuso. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Recomendamos utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que AWS Proton otorga a otro servicio al recurso. Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar ambas claves de contexto de condición global para limitar los permisos. Si utiliza claves de contexto de condición global y el valor de `aws:SourceArn` contiene el ID de cuenta, el valor de `aws:SourceAccount` y la cuenta en el valor de `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utiliza en la misma instrucción de política. Utilice `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utilice `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

El valor de `aws:SourceArn` debe ser un recurso que AWS Proton almacene.

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si especifica varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws::proton:*:123456789012:environment/*`.

El siguiente ejemplo muestra cómo se pueden utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` en AWS Proton para evitar el problema del adjunto confundido.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "ExampleProtonConfusedDeputyPreventionPolicy",
```

```
        "Effect": "Allow",
        "Principal": {"Service": "proton.amazonaws.com"},
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
            }
        }
    }
}
```

Soporte personalizado de una VPC de Amazon para el aprovisionamiento de CodeBuild

El aprovisionamiento de AWS Proton CodeBuild ejecuta comandos de la CLI arbitrarios que proporciona el cliente en un proyecto de CodeBuild ubicado en una cuenta del entorno de AWS Proton. Estos comandos suelen administrar los recursos mediante una herramienta de infraestructura como código (IaC), como CDK. Si tiene recursos en una VPC de Amazon, es posible que CodeBuild no pueda acceder a ellos. Para habilitar esto, CodeBuild admite la capacidad de ejecutarse en una VPC de Amazon específica. Algunos ejemplos de casos de uso incluyen:

- Recuperar las dependencias de repositorios de artefactos internos y autoalojados, como `PyPI` para Python, `Maven` para Java y `npm` para Node.js
- CodeBuild necesita acceder a un servidor Jenkins en una VPC de Amazon determinada para registrar una canalización.
- Acceder a los objetos de un bucket de Amazon S3 configurado para permitir el acceso únicamente a través de un punto de conexión de VPC de Amazon.
- Ejecutar pruebas de integración desde la compilación con los datos de una base de datos de Amazon RDS aislada en una subred privada.

Para obtener más información, consulte la documentación de VPC y CodeBuild.

Si desea que el aprovisionamiento de CodeBuild se ejecute en una VPC personalizada, AWS Proton le ofrece una solución sencilla. En primer lugar, debe agregar el ID de la VPC, las subredes y los grupos de seguridad a la plantilla de entorno. A continuación, introduzca esos valores en la

especificación del entorno. Esto dará como resultado la creación de un proyecto de CodeBuild dirigido a una VPC determinada.

Actualización de la plantilla de entorno

Esquema

El ID de la VPC, las subredes y los grupos de seguridad deben agregarse al esquema de la plantilla para que puedan existir en las especificaciones del entorno.

Un ejemplo `schema.yaml`:

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentInputType"
  types:
    EnvironmentInputType:
      type: object
      properties:
        codebuild_vpc_id:
          type: string
        codebuild_subnets:
          type: array
          items:
            type: string
        codebuild_security_groups:
          type: array
          items:
            type: string
```

Esto agrega tres propiedades nuevas que utilizará el manifiesto:

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifiesto

Para configurar los ajustes de la VPC de Amazon en CodeBuild, hay disponible una propiedad opcional llamada `project_properties` en el manifiesto de la plantilla. El contenido de

`project_properties` se añade a la pila de AWS CloudFormation que crea el proyecto de CodeBuild. Esto permite añadir no solo propiedades de AWS CloudFormation a la VPC de Amazon, sino también cualquier propiedad compatible de CloudFormation a CodeBuild, como el tiempo de espera de la compilación. Los mismos datos proporcionados a `proton-inputs.json` están disponibles para los valores de `project_properties`.

Añada esta sección a su `manifest.yaml`:

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

El `manifest.yaml` resultante puede tener el siguiente aspecto:

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy -- --force
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Creación del entorno

Al crear un entorno con su plantilla habilitada para la VPC del aprovisionamiento de CodeBuild, debe proporcionar el ID de la VPC de Amazon, las subredes y los grupos de seguridad.

Para obtener una lista de todos los ID de VPC de Amazon en su región, ejecute el siguiente comando:

```
aws ec2 describe-vpcs
```

Para obtener una lista de todos los ID de subred, ejecute:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> Incluya solo las subredes privadas. CodeBuild fallará si proporciona subredes públicas. A diferencia de las subredes privadas, las subredes públicas tienen una ruta predeterminada a una puerta de enlace de Internet.

Ejecute el siguiente comando para obtener los ID de los grupos de seguridad. Estos ID también se pueden obtener a través de la AWS Management Console:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Los valores se parecerán a los siguientes:

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

Garantizar los permisos de CodeBuild

La compatibilidad con la VPC de Amazon requiere ciertos permisos, como la capacidad de crear una interfaz de red elástica.

Si el entorno se está creando en la consola, introduzca estos valores durante el asistente de creación del entorno. Si desea crear el entorno mediante programación, su `spec.yaml` tendrá el siguiente aspecto:

```
proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

Recursos y etiquetado de AWS Proton

Los recursos de AWS Proton a los que se les asigna un nombre de recurso de Amazon (ARN) incluyen plantillas de entorno y sus versiones principales y secundarias, así como plantillas de servicio y sus versiones principales y secundarias, entornos, servicios, instancias de servicio, componentes y repositorios. Puede etiquetar estos recursos como ayuda para identificarlos y organizarlos. Utilice etiquetas para clasificar los recursos según su finalidad, propietario, entorno u otro criterio. Para obtener más información, consulte Estrategias de etiquetado de . Para realizar un seguimiento de sus recursos de AWS Proton y administrarlos, puede utilizar las características de etiquetado que se describen en las siguientes secciones.

Etiquetado de AWS

Puede asignar metadatos a los recursos de AWS en forma de etiquetas. Cada etiqueta consta de una clave definida por el cliente y un valor opcional. Las etiquetas pueden ayudarle a administrar, identificar, organizar, buscar y filtrar recursos.

> **Important**
>
> No agregue información de identificación personal (PII) ni otra información confidencial en las etiquetas. Las etiquetas son accesibles para muchos Servicios de AWS, incluida la facturación. Las etiquetas no se han diseñado para usarse con información privada o confidencial.

Cada etiqueta de tiene dos partes:

- Una clave de etiqueta (por ejemplo, `CostCenter`, `Environment`, o `Project`). Las claves de etiqueta distinguen entre mayúsculas y minúsculas.
- Un valor de etiqueta (opcional) (por ejemplo, `111122223333` o `Production`). Al igual que las claves de etiqueta, los valores de etiqueta distinguen entre mayúsculas y minúsculas.

Los siguientes requisitos básicos de nomenclatura y uso se aplican a las etiquetas.

- Cada recurso puede tener un máximo de 50 etiquetas creadas por el usuario.

> **Note**
>
> Las etiquetas creadas por el sistema que comienzan por el prefijo `aws:` están reservadas para uso de AWS y no cuentan con este límite. No puede editar ni eliminar una etiqueta que comience con el prefijo `aws:`.

- Para cada recurso, cada clave de etiqueta debe ser única y solo puede tener un valor.
- La clave de etiqueta debe tener un mínimo de 1 y un máximo de 128 caracteres Unicode en UTF-8.
- El valor de etiqueta debe ser un mínimo de 1 y un máximo de 256 caracteres Unicode en UTF-8.
- Los caracteres permitidos en las etiquetas son letras, números y espacios representables en UTF-8, además de los siguientes caracteres:* _ . : / = + - @.

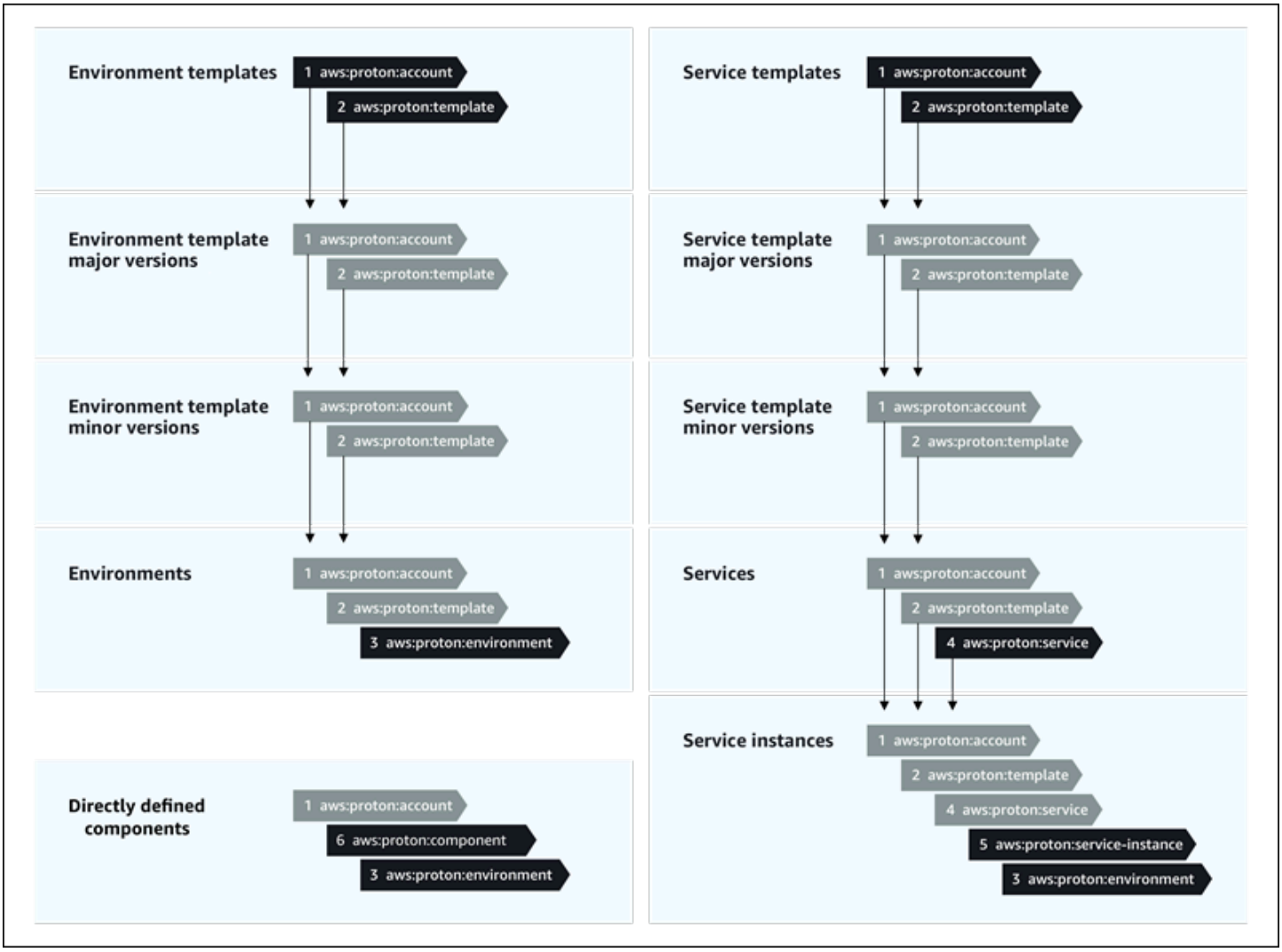
Etiquetado de AWS Proton

Con AWS Proton, puede utilizar tanto las etiquetas que cree como las que AWS Proton genere automáticamente.

Etiquetas administradas por AWS ProtonAWS

Al crear un recurso de AWS Proton, AWS Proton genera automáticamente etiquetas administradas por AWS para el nuevo recurso, como se muestra en el siguiente diagrama. Las etiquetas administradas por AWS se propagan posteriormente a otros recursos de AWS Proton que se basan en el nuevo recurso. Por ejemplo, las etiquetas administradas de una plantilla de entorno se propagan a sus versiones y las etiquetas administradas de un servicio se propagan a sus instancias de servicio.

> **Note**
>
> Las etiquetas administradas por AWS no se generan para las conexiones de las cuentas del entorno. Para obtener más información, consulte the section called “Conexiones de cuentas”.



Propagación de etiquetas a recursos aprovisionados

Si los recursos aprovisionados, como los definidos en las plantillas de servicio y entorno, admiten el etiquetado de AWS, las etiquetas administradas por AWS se propagan como etiquetas administradas por el cliente a los recursos aprovisionados. Estas etiquetas no se propagarán a un recurso aprovisionado que no admita el etiquetado de AWS.

AWS Proton aplica etiquetas a los recursos por cuentas de AWS Proton, plantillas registradas y entornos implementados, así como por servicios e instancias de servicio, tal y como se describe en la siguiente tabla. Puede utilizar etiquetas administradas por AWS para ver y administrar los recursos de AWS Proton, pero no puede modificarlas.

Clave de etiqueta administrada por AWS	Clave administrada por el cliente propagada	Descripción
`aws:proton:account`	`proton:account`	La cuenta de AWS que crea e implementa los recursos de AWS Proton.
`aws:proton:template`	`proton:template`	El ARN de una plantilla seleccionada.
`aws:proton:environment`	`proton:environment`	El ARN de un entorno seleccionado.
`aws:proton:service`	`proton:service`	El ARN de un servicio seleccionado.
`aws:proton:service-instance`	`proton:service-instance`	El ARN de una instancia de servicio seleccionada.
`aws:proton:component`	`proton:component`	El ARN de un componente seleccionado.

A continuación, se muestra un ejemplo de una etiqueta administrada por AWS para un recurso de AWS Proton.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

A continuación, se muestra un ejemplo de una etiqueta administrada por el cliente aplicada a un recurso aprovisionado que se propagó desde una etiqueta administrada por AWS.

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

Con el aprovisionamiento administrado por AWS, AWS Proton aplica las etiquetas propagadas directamente a los recursos aprovisionados.

Con el aprovisionamiento autoadministrado, AWS Proton permite que las etiquetas propagadas estén disponibles junto con los archivos de IaC representados que envía en la solicitud de extracción

(PR) de aprovisionamiento. Las etiquetas se proporcionan en la variable de mapa de cadenas `proton_tags`. Le recomendamos que haga referencia a esta variable en su configuración de Terraform para incluirla en etiquetas de AWS Proton en `default_tags`. Esto propaga las etiquetas de AWS Proton a todos los recursos aprovisionados.

El siguiente ejemplo muestra este método de propagación de etiquetas en una plantilla de Terraform de entorno.

Esta es la definición de la variable `proton_tags`:

proton.environment.variables.tf:

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

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

Así es como se asignan los valores de las etiquetas a esta variable:

proton.auto.tfvars.json:

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

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

```
}
```

Y así es como puede agregar etiquetas de AWS Proton a su configuración de Terraform para que se agreguen a los recursos aprovisionados:

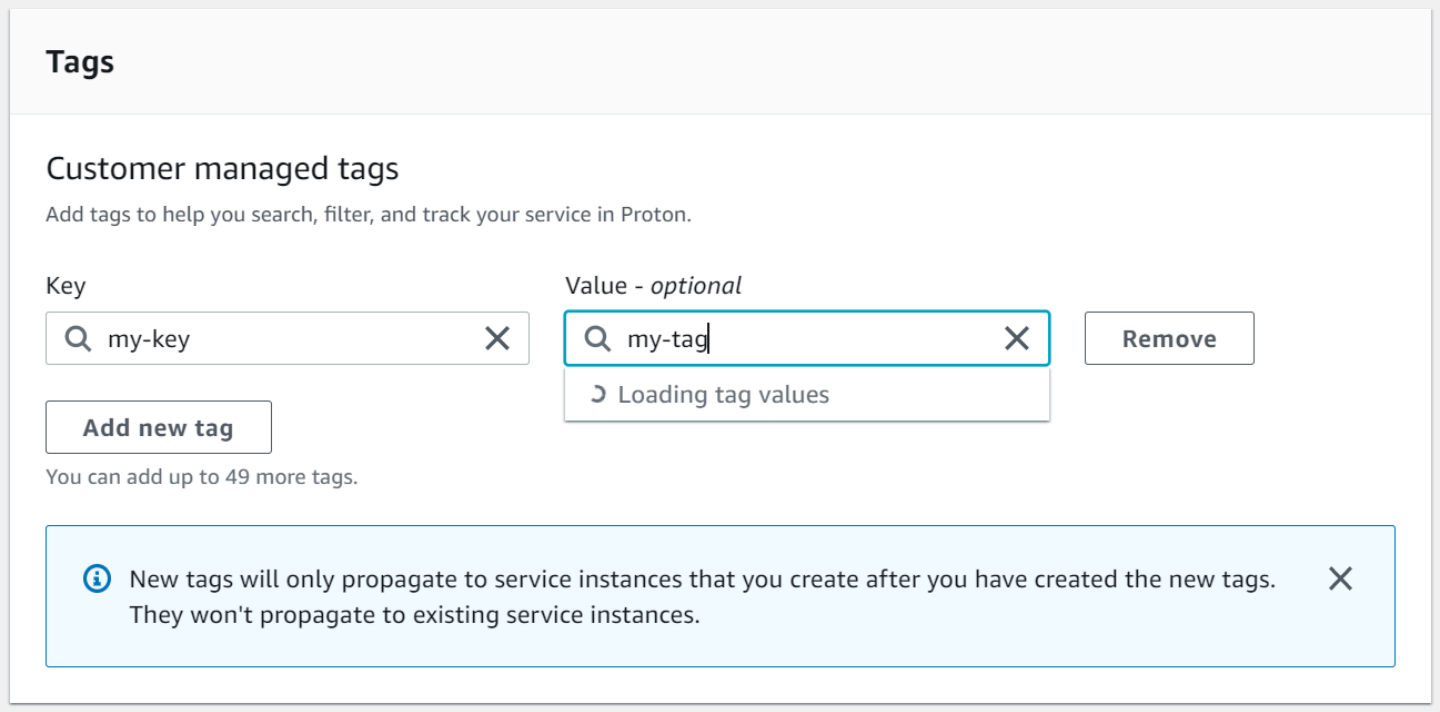
```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Etiquetas administradas por el cliente

Cada recurso de AWS Proton tiene una cuota máxima de 50 etiquetas administradas por el cliente. Las etiquetas administradas por el cliente se propagan a los recursos de AWS Proton secundarios de la misma manera que las etiquetas administradas por AWS, excepto que no se propagan a los recursos de AWS Proton existentes ni a los recursos aprovisionados. Si aplica una etiqueta nueva a un recurso de AWS Proton con recursos secundarios existentes y desea etiquetar los recursos secundarios existentes con la nueva etiqueta, deberá etiquetar cada recurso secundario existente manualmente, mediante la consola o la AWS CLI.

Creación de etiquetas mediante la consola y la CLI

Cuando el usuario crea un recurso de AWS Proton mediante la consola, tiene la oportunidad de crear etiquetas administradas por el cliente tanto en la primera como en la segunda página del procedimiento de creación, como se muestra en la siguiente instantánea de la consola. Seleccione Agregar nueva etiqueta e introduzca la clave y el valor.



Tras crear un nuevo recurso mediante la consola de AWS Proton, podrá ver su lista de etiquetas administradas por AWS y administradas por el cliente en la página de detalles.

Creación o edición de etiquetas

1. En la consola de AWS Proton, abra una página de detalles del recurso de AWS Proton en la que podrá ver una lista de etiquetas.
2. Elija Administrar etiquetas.
3. En la página Administrar etiquetas, puede ver, crear, eliminar y editar etiquetas. No puede modificar las etiquetas administradas por AWS que aparecen en la parte superior. Sin embargo, puede añadir y modificar las etiquetas administradas por el cliente mediante campos de edición, que aparecen después de las etiquetas administradas por AWS.

   Para crear una etiqueta nueva, elija Agregar nueva etiqueta.
4. Escriba una clave y un valor para la nueva etiqueta.
5. Para editar una etiqueta, introduzca un valor en el campo de valor de la etiqueta para una clave seleccionada.
6. Para eliminar una etiqueta, seleccione Eliminar para una etiqueta seleccionada.
7. Cuando haya completado los cambios, elija Guardar cambios.

Creación de etiquetas mediante la AWS ProtonAWS CLI

Puede ver, crear, eliminar y editar etiquetas mediante la AWS Proton AWS CLI.

También puede crear o editar una etiqueta para un recurso, tal y como se muestra en el siguiente ejemplo.

```
$ 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"}]'
```

Puede eliminar una etiqueta de un recurso, como se muestra en el siguiente ejemplo.

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

Puede enumerar las etiquetas de un recurso, como se muestra en el ejemplo final.

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

Solución de problemas de AWS Proton

Obtenga información sobre cómo solucionar problemas de AWS Proton.

Temas

Errores de implementación que hacen referencia a parámetros dinámicos de AWS CloudFormation

Si ve errores de implementación que hagan referencia a las Variables dinámicas de CloudFormation, compruebe si son casos de Jinja escaping. Estos errores pueden deberse a una mala interpretación por parte de Jinja de las variables dinámicas. La sintaxis de los parámetros dinámicos de CloudFormation es muy similar a la sintaxis de Jinja que se utiliza con los parámetros de AWS Proton.

Ejemplo de sintaxis de variables dinámicas de CloudFormation:

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

Ejemplo de sintaxis de Jinja del parámetro de AWS Proton:

`'{{ service_instance.environment.outputs.env-outputs }}'`.

Para evitar estos errores de interpretación errónea, Jinja escapa los parámetros dinámicos de CloudFormation como se muestra en los siguientes ejemplos.

Este ejemplo proviene de la Guía del usuario de AWS CloudFormation. Los segmentos de AWS Secrets Manager nombre secreto y clave json se pueden utilizar para recuperar las credenciales de inicio de sesión almacenadas en el secreto.

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
```

```
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
```

Para escapar los parámetros dinámicos de CloudFormation, se pueden utilizar dos métodos diferentes:

- Incluya un bloque entre `{% raw %}` and `{% endraw %}`:

```
'{% raw %}'
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
'{% endraw %}'
```

- Incluya un parámetro entre `"{{ }}"`:

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}' }}"
      MasterUserPassword:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}' }}"
```

Para obtener más información, consulte Jinja escaping.

Cuotas de AWS Proton

En la siguiente tabla se muestran las cuotas de AWS Proton. Todos los valores son por cuenta de AWS y por región de AWS admitida.

Cuota de recursos	Límite predeterminado	¿Ajustable?
Tamaño máximo de un paquete de plantillas	10 MB	✗ No
Tamaño máximo del archivo de manifiesto de plantilla	2 MB	✗ No
Tamaño máximo del archivo de esquema de plantilla	2 MB	✗ No
Tamaño máximo de cada archivo de plantilla	2 MB	✗ No
Longitud máxima para nombres de plantillas	100 caracteres	✗ No
Número máximo de archivos de plantilla de CloudFormation por paquete	1	✗ No
Número máximo de plantillas registradas por cuenta y plantillas de servicio y de entorno combinadas	1000	✓ Sí
Número máximo de versiones de plantillas registradas por plantilla	1000	✓ Sí
Número máximo de archivos por paquete de aprovisionamiento de CodeBuild	500	✗ No
Número máximo de entornos por cuenta	1000	✓ Sí
Número máximo de servicios por cuenta	1000	✓ Sí
Número máximo de instancias de servicio por servicio	20	✓ Sí
Número máximo de componentes por cuenta	1000	✓ Sí

Cuota de recursos	Límite predeterminado	¿Ajustable?
Número máximo de conexiones de cuentas de entorno por cuenta de entorno	1000	✓ Sí

Historial del documento

En la siguiente tabla se describen los cambios importantes en la documentación relacionados con la última versión AWS Proton y los comentarios de los clientes. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

- Versión de la API: 20/7/2020

Cambio	Descripción	Fecha
Actualización de la política administrada	La política `AWSProtonDeveloperAccess` se actualizó.	25 de abril de 2024
Actualización de la política administrada	La política `AWSProtonFullAccess` se actualizó.	25 de abril de 2024
Actualización de la política administrada	La política `AWSProtonSyncServiceRolePolicy` se actualizó.	25 de abril de 2024
Actualización de la política administrada	La política `AWSProtonCodeBuildProvisioningServiceRolePolicy` se actualizó.	12 de mayo de 2023
Configuraciones de sincronización de servicios.	AWS Proton añade compatibilidad con las configuraciones de sincronización de servicios.	31 de marzo de 2023
CodeBuild	AWS Proton añade soporte para el CodeBuildaprovisionamiento.	16 de noviembre de 2022
Actualización de la política administrada	Se agregó una `AWSProtonCodeBuildProvisioningBasicAccess` política	11 de noviembre de 2022

	que otorga CodeBuild los permisos necesarios para ejecutar una compilación para el AWS Proton CodeBuild aprovisionamiento.	
Propagación de etiquetas de Terraform	Se agregó la propagación de etiquetas de Terraform al capítulo de Etiquetado.	16 de septiembre de 2022
Guía sobre la migración de API	Se eliminó la guía sobre la migración de la API anterior a la disponibilidad general (GA).	12 de agosto de 2022
AWS Proton objetos	Se ha añadido un tema sobre AWS Proton los objetos y su relación con objetos AWS ajenos y de terceros. Consulte objetos de AWS Proton.	29 de julio de 2022
Aclaraciones sobre los repositorios enlazados	Se aclaró el propósito de los repositorios enlazados (registrados) y su uso en toda la guía.	18 de julio de 2022
Combinación de guías	Se combinaron las dos guías independientes destinadas a administradores y usuarios en una sola guía: la Guía del usuario de AWS Proton .	30 de junio de 2022

Actualización de la política administrada	Se actualizaron las políticas gestionadas para proporcio nar acceso a las nuevas operaciones de la AWS Proton API y solucionar problemas de permisos en algunas operaciones de AWS Proton la consola. Consulte las políticas AWS gestionadas para AWS Proton.	20 de junio de 2022
Introducción a la CLI	Se actualizó la sección Introducción a la AWS CLI con un nuevo tutorial que utiliza la nueva biblioteca de plantillas.	14 de junio de 2022
Componentes definidos directamente	Se agregó el capítulo Componentes y se realizaron las modificaciones correspon dientes a lo largo de la guía.	1 de junio de 2022
AWS Proton biblioteca de plantillas	Se agregó el tema «La biblioteca de AWS Proton plantillas».	6 de mayo de 2022
Disponibilidad general (GA) de Terraform	Se cambió el nombre de aprovisionamiento de solicitud es de extracción por el de aprovisionamiento autoadmin istrado. Se ha agregado un tema sobre Métodos de aprovisionamiento.	23 de marzo de 2022
Etiquetado de un repositorio	Se agregó soporte para etiquetar recursos de repositor ios. Consulte Crear un enlace a su repositorio.	23 de marzo de 2022

Actualización de la documentación	Se agregó el etiquetado de conexiones de cuentas de entorno.	26 de noviembre de 2021
Sincronización de plantillas y versión preliminar de Terraform	Se agregó el control de versiones automático de plantillas con la característica de sincronización de plantillas para una disponibilidad general y el aprovisionamiento de solicitudes de extracción con Terraform en versión preliminar. Se volvió a agregar la guía de migración de la API.	24 de noviembre de 2021
Actualizaciones de la documentación	Se han añadido mejoras en el EventBridgetutorial, el flujo de trabajo de AWS Proton introducción, el funcionamiento y la sección de paquetes de plantillas.	17 de septiembre de 2021
AWS Proton lanzamiento de los paneles de ayuda de la consola	Se agregaron paneles de ayuda a la consola. Al eliminar la versión de una plantilla de la consola ya no se eliminan las versiones anteriores. Se eliminó la guía sobre la migración de la API.	8 de septiembre de 2021
AWS Proton versión de disponibilidad general (GA)	Se agregaron entornos multicuentas, EventBridge monitoreo, claves de condición de IAM, compatibilidad con la idempotencia y mayores cuotas.	9 de junio de 2021

Añada y elimine instancias de servicio para un servicio y utilice la infraestructura externa existente para los entornos con AWS Proton	Esta versión preliminar pública incluye actualizaciones que permiten añadir y eliminar instancias de servicio de un servicio, utilizar la infraestructura externa existente en un AWS Proton entorno y cancelar las implementaciones de entornos, instancias de servicio y canalizaciones. AWS Proton ahora es compatible PrivateLink. Se agregó una validación de eliminación adicional para evitar que una versión secundaria se elimine por error mientras un recurso la esté utilizando.	27 de abril de 2021
Etiquetar con AWS Proton	La versión preliminar pública 2 incluye el AWS Proton etiquetado y la posibilidad de lanzar servicios sin una canalización de servicios.	5 de marzo de 2021
Versión inicial	La versión preliminar pública ya está disponible en determinadas regiones.	1 de diciembre de 2020

Glosario de AWS

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.