Guía para desarrolladores

AWS App Runner

AWS App Runner: Guía para desarrolladores

Table of Contents

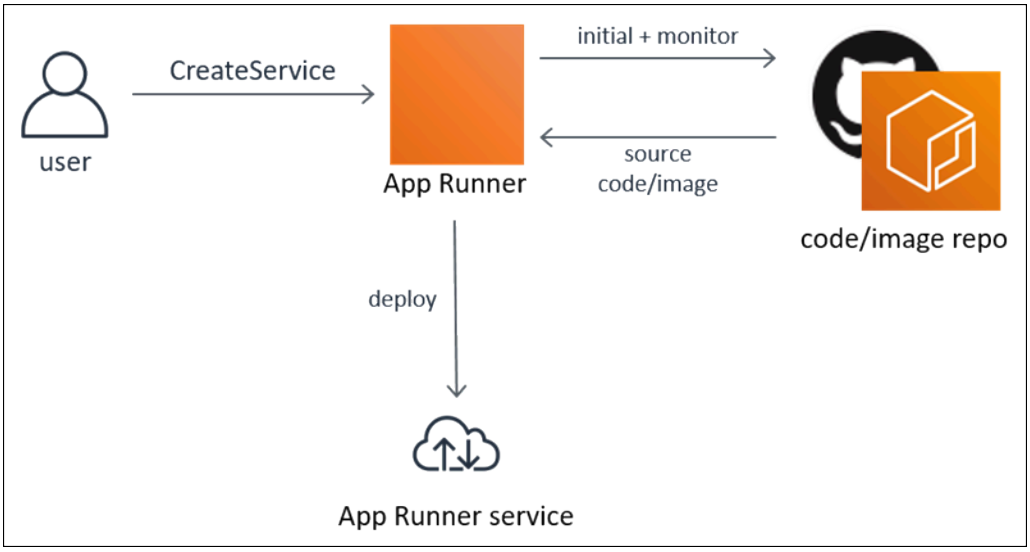
AWS App Runner ya no está abierto a nuevos clientes. Los clientes existentes pueden seguir utilizando el servicio con normalidad. Para obtener más información, consulte Cambio en la disponibilidad de AWS App Runner.

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.

¿Qué es AWS App Runner?

AWS App Runner es un AWS servicio que proporciona una forma rápida, sencilla y rentable de implementar directamente desde el código fuente o una imagen de contenedor hasta una aplicación web escalable y segura en la AWS nube. No necesita aprender nuevas tecnologías, decidir qué servicio de cómputo usar ni saber cómo aprovisionar y configurar AWS los recursos.

App Runner se conecta directamente a tu repositorio de código o imágenes. Proporciona un proceso automático de integración y entrega con operaciones totalmente gestionadas, alto rendimiento, escalabilidad y seguridad.



¿A quién va dirigido App Runner?

Si eres desarrollador, puedes usar App Runner para simplificar el proceso de implementación de una nueva versión de tu repositorio de código o imágenes.

Para los equipos de operaciones, App Runner permite realizar despliegues automáticos cada vez que se envía una confirmación al repositorio de código o se envía una nueva versión de una imagen de contenedor al repositorio de imágenes.

Acceder a App Runner

Puede definir y configurar las implementaciones del servicio de App Runner mediante cualquiera de las siguientes interfaces:

- Consola de App Runner: proporciona una interfaz web para administrar los servicios de App Runner.
- API de App Runner: proporciona una RESTful API para realizar acciones de App Runner. Para obtener más información, consulte Referencia de la API de AWS App Runner.
- AWS Interfaz de línea de comandos (AWS CLI): proporciona comandos para un amplio conjunto de AWS servicios, incluida Amazon VPC, y es compatible con Windows, macOS y Linux. Para obtener más información, consulte AWS Command Line Interface.
- AWS SDKs— Proporciona un idioma específico APIs y se ocupa de muchos de los detalles de la conexión, como el cálculo de las firmas, la gestión de los reintentos de las solicitudes y la gestión de los errores. Para obtener más información, consulte AWS SDKs.

Precios de App Runner

App Runner proporciona una forma rentable de ejecutar su aplicación. Solo pagas por los recursos que consuma tu servicio de App Runner. Su servicio se reduce a menos instancias de cómputo cuando el tráfico de solicitudes es menor. Usted tiene el control de la configuración de escalabilidad: el número mínimo y máximo de instancias aprovisionadas y la carga máxima que gestiona una instancia.

Para obtener más información sobre el escalado automático de App Runner, consulte. the section called “Escalado automático”

Para obtener información sobre precios, consulte Precios de AWS App Runner.

Siguientes pasos

Obtenga información sobre cómo empezar a usar App Runner en los siguientes temas:

- Configuración— Complete los pasos previos para usar App Runner.
- Introducción— Implemente su primera aplicación en App Runner.

AWS App Runner cambio de disponibilidad

Tras pensarlo detenidamente, decidimos cerrar la venta AWS App Runner de nuevos clientes. AWS App Runner Los clientes actuales pueden seguir utilizando el servicio con normalidad, incluida la creación de nuevos recursos y servicios. AWS seguimos invirtiendo en seguridad y disponibilidad AWS App Runner, pero no tenemos previsto introducir nuevas funciones.

Recomendamos que los clientes exploren el modo exprés de Amazon Elastic Container Service (Amazon ECS) al migrar desde. AWS App Runner El modo Amazon ECS Express conserva la simplicidad operativa de App Runner y, al mismo tiempo, proporciona acceso al conjunto más amplio de funciones de Amazon ECS. Con una sola llamada a la API, usted proporciona una imagen de contenedor y dos funciones de IAM, y Amazon ECS aprovisiona un conjunto completo de aplicaciones en su AWS cuenta, que incluye un servicio de ECS en Fargate, un Application Load Balancer, autoescalado y redes. El uso del modo Amazon ECS Express Mode no conlleva ningún cargo adicional. Solo paga por los AWS recursos subyacentes creados para ejecutar su aplicación.

En esta guía, se describe cómo migrar un servicio de App Runner existente al modo ECS Express y cómo desplazar el tráfico gradualmente mediante el enrutamiento de DNS.

Información general sobre la migración

Esta guía utiliza un enfoque de blue/green implementación con enrutamiento ponderado de DNS para migrar el tráfico de App Runner al modo ECS Express. Ambos servicios se ejecutan simultáneamente durante la migración. Utiliza Amazon Route 53 (o su proveedor de DNS) para transferir gradualmente el tráfico del servicio App Runner al servicio ECS Express Mode, empezando por un pequeño porcentaje y aumentando con el tiempo. Este enfoque minimiza el tiempo de inactividad y le permite revertirlo ajustando las ponderaciones del DNS en caso de que surjan problemas.

Una migración típica incluye los siguientes pasos:

1. Revise la configuración del servicio App Runner existente
2. Cree un servicio ECS Express Mode con la misma imagen de contenedor
3. Configure el mismo dominio personalizado para el servicio ECS Express Mode, si utiliza un dominio personalizado
4. Cambie el tráfico de App Runner al modo ECS Express mediante el enrutamiento de DNS
5. Complete la migración y elimine el servicio App Runner cuando ya no sea necesario

Requisitos previos

Antes de empezar, asegúrate de tener lo siguiente:

- Una AWS cuenta con AWS Identity and Access Management los permisos adecuados para crear y gestionar los recursos de Amazon ECS AWS App Runner, Amazon Route 53 y Application Load Balancer
- AWS CLI instalada y configurada con las credenciales de su cuenta AWS
- Una imagen de contenedor almacenada en Amazon Elastic Container Registry (u otro registro de contenedores) para implementarla en ECS Express Mode
- Las funciones de IAM requeridas por ECS Express Mode: `ecsTaskExecutionRole` para la ejecución de tareas de Amazon ECS y `ecsInfrastructureRoleForExpressServices` para el aprovisionamiento de la infraestructura ECS Express Mode

Si desea conservar un dominio personalizado existente durante la migración, también necesitará:

- Un nombre de dominio registrado que usted controle, por ejemplo`app.example.com`, mediante Amazon Route 53 o un registrador de dominios de terceros
- Un SSL/TLS certificado en AWS Certificate Manager(ACM) que coincida con su dominio personalizado. Solicita un certificado ACM público en el mismo Región de AWS lugar en el que vas a implementar tus recursos. Tanto App Runner como Amazon ECS Express Mode requieren un certificado ACM para habilitar el acceso HTTPS con dominios personalizados.

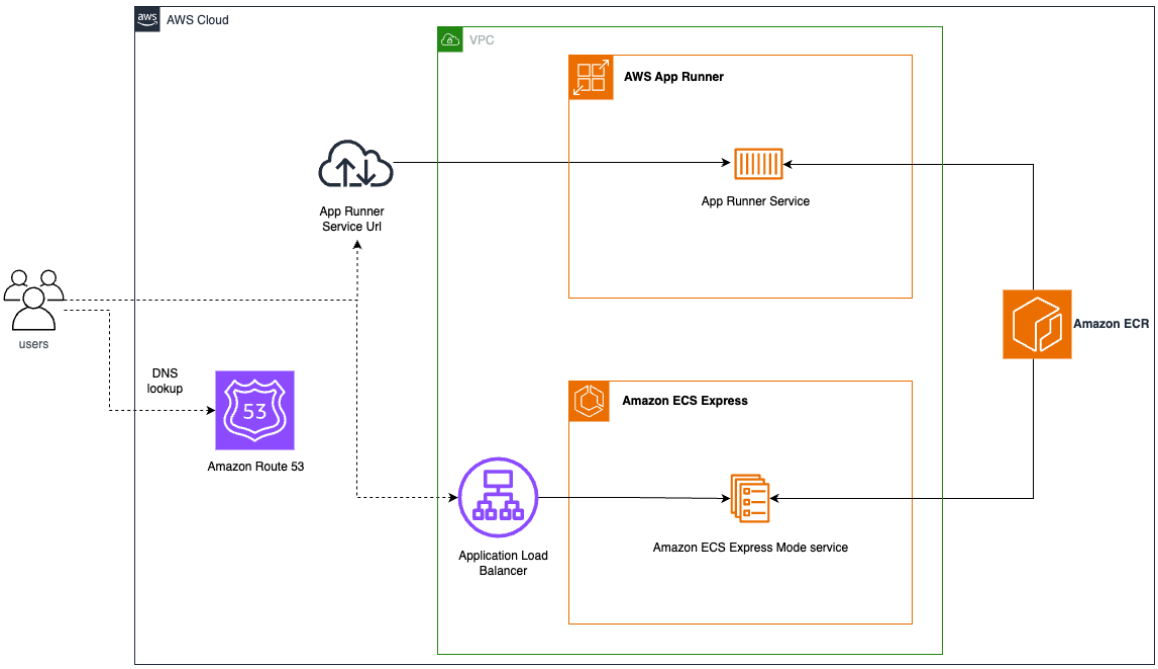
Antes de empezar

- Requisito de imagen de contenedor: ECS Express Mode implementa una imagen de contenedor. Si el servicio App Runner se implementa desde el código fuente, primero agrega un paso de compilación que cree una imagen de contenedor y la envíe a un registro como Amazon Elastic Container Registry. A continuación, implemente esa imagen en ECS Express Mode. Consulte Migración de las implementaciones basadas en el origen para obtener más información sobre la migración de las implementaciones basadas en el origen.
- Comportamiento del dominio: si tu servicio de App Runner ya usa un dominio personalizado, por ejemplo`app.example.com`, puedes reutilizar ese mismo nombre de host durante la migración y cambiar gradualmente el tráfico entre App Runner y ECS Express Mode actualizando el DNS.

Si tu servicio App Runner usa solo la URL predeterminada del servicio App Runner, el servicio ECS Express Mode tendrá un punto final diferente. En este caso, no hay ningún nombre de host compartido que pueda usarse para un cambio gradual del tráfico. Debe crear y validar el servicio ECS Express Mode y, a continuación, actualizar los clientes o el DNS para usar el nuevo punto final.

Tutorial de migración

El siguiente diagrama muestra cómo funciona la migración con Route 53 para cambiar los registros de DNS entre el servicio App Runner y el servicio ECS Express Mode.



Paso 1: Revise la configuración actual de App Runner

En la consola de App Runner, revisa tu servicio actual y anota los valores que deseas transferir. Como mínimo, ten en cuenta lo siguiente:

- Imagen de contenedor
- Puerto de aplicación
- Variables de entorno
- Nombre de dominio personalizado, si está configurado
- Certificado ACM asociado al dominio personalizado, si está configurado

También puede revisar cualquier otra configuración de tiempo de ejecución que desee transferir al nuevo servicio.

Para obtener detalles de dominio personalizados, consultethe section called “Nombres de dominio personalizados”.

Paso 2: Cree el servicio ECS Express Mode

Cree un servicio ECS Express Mode con la misma imagen de contenedor que utiliza su servicio App Runner. Puede crear el servicio mediante el Consola de administración de AWSo el AWS CLI.

Ejemplo de comando de la CLI:

```
aws ecs create-express-gateway-service \
    --execution-role-arn arn:aws:iam::123456789012:role/ecsTaskExecutionRole \
    --infrastructure-role-arn arn:aws:iam::123456789012:role/
ecsInfrastructureRoleForExpressServices \
    --primary-container '{
        "image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/my-app:latest",
        "containerPort": 8080,
        "environment": [{
            "name": "ENV_VAR_NAME",
            "value": "value"
        }]
    }' \
    --service-name "my-application" \
    --health-check-path "/" \
    --scaling-target '{"minTaskCount":1,"maxTaskCount":4}' \
    --monitor-resources
```

Sustituya la imagen, el puerto, las variables de entorno y los valores de escalado por los de su servicio App Runner.

Este comando proporciona una pila de aplicaciones completa en su AWS cuenta, que incluye un servicio de ECS en Fargate, un Application Load Balancer con grupos objetivo y comprobaciones de estado, políticas de autoescalado, grupos de seguridad y configuración de redes, y una URL predeterminada.

El aprovisionamiento suele tardar entre 3 y 5 minutos. Puede realizar un seguimiento del progreso en la consola de Amazon ECS, en la pestaña Recursos.

Una vez completado, pruebe el servicio ECS Express Mode utilizando la URL predeterminada que aparece en la consola. Compruebe que la aplicación funciona correctamente antes de continuar con el cambio de tráfico.

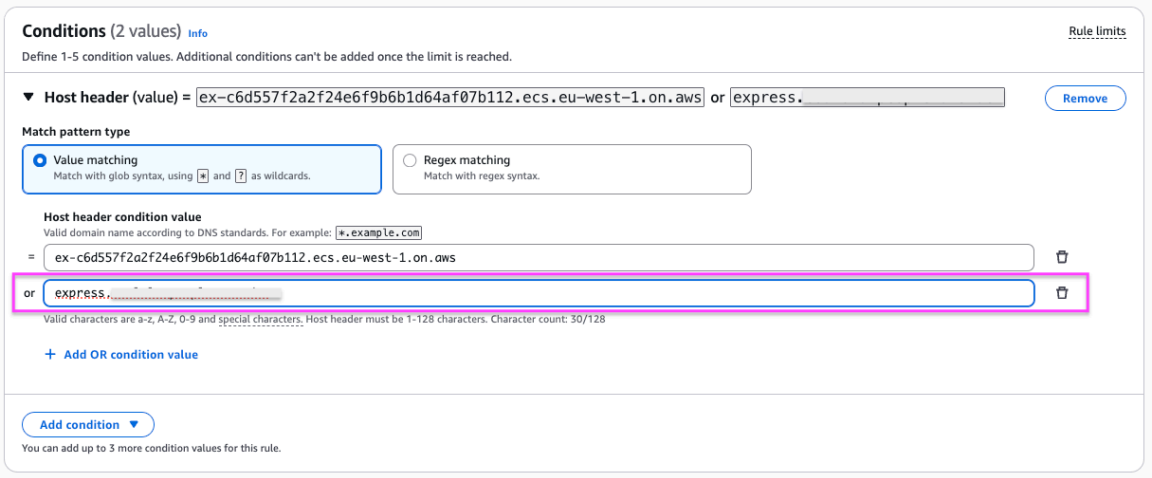
Paso 3: Configure el dominio personalizado para el modo exprés de ECS

Si su servicio App Runner usa un dominio personalizado, configure el mismo dominio personalizado para el servicio ECS Express Mode antes de cambiar el tráfico. Este paso configura el Application Load Balancer creado para el servicio ECS Express Mode para que acepte el tráfico de su dominio y utilice el certificado ACM para HTTPS.

- Agregue su dominio personalizado como condición de encabezado de host en la regla de escucha de Application Load Balancer. Usa el mismo nombre de dominio que asociaste a tu servicio de App Runner (por ejemplo,`app.example.com`). Esto le indica al Application Load Balancer que dirija el tráfico desde su dominio al grupo objetivo de ECS Express Mode.
- Agregue el certificado SSL al agente de escucha HTTPS de Application Load Balancer. Agregue el certificado ACM indicado en el paso 1 al listener HTTPS.

Para obtener instrucciones detalladas, consulte Añadir un dominio personalizado a su servicio en la Guía para desarrolladores de Amazon ECS.

La siguiente imagen muestra un ejemplo de configuración de la condición del encabezado del host en la regla de escucha de Application Load Balancer.



Paso 4: Cambie el tráfico mediante el enrutamiento ponderado de Route 53

Si su servicio App Runner ya usa un dominio personalizado, puede transferir gradualmente el tráfico al servicio ECS Express Mode mediante el enrutamiento ponderado de Route 53. El enrutamiento ponderado le permite enrutar el tráfico del mismo nombre de host a varios puntos finales. Cada punto final se define como un registro DNS independiente con su propio peso, y Route 53 distribuye las solicitudes de acuerdo con esos pesos.

> **Note**
>
> En esta guía, se utiliza Route 53 como ejemplo. Si usa otro proveedor de DNS, realice cambios de DNS equivalentes utilizando las funciones de administración de tráfico de su proveedor.

Convierte el registro existente de App Runner en un registro ponderado:

1. Abre la consola de Route 53.
2. Selecciona Zonas alojadas y, a continuación, selecciona la zona alojada para tu dominio.
3. Busca el registro existente para tu nombre de host (por ejemplo`app.example.com`) que actualmente apunta a App Runner.
4. Edita el registro y cambia su política de enrutamiento a Weighted.
5. Defina el peso en `100` (esto dirige todo el tráfico inicial a App Runner).
6. En ID de registro, introduce un identificador descriptivo como`app-runner-service`.
7. Seleccione Save changes (Guardar cambios).

Cree un registro ponderado para el modo ECS Express:

1. Cree un registro nuevo en la misma zona alojada.
2. Utilice el mismo nombre de registro (por ejemplo`app.example.com`).
3. Utilice el mismo tipo de registro.
4. Establezca la política de enrutamiento en Ponderada.
5. En Enrutar el tráfico a, selecciona Alias to Application y Classic Load Balancer.
6. Elija su ECS Express Mode Application Load Balancer en el menú desplegable.

7. Establezca el peso en `0` (no fluirá tráfico hacia el modo ECS Express hasta que aumente el peso de forma explícita).
8. En ID de registro, introduzca un identificador descriptivo como`ecs-express-service`.
9. Elija Crear registros.

Cambie el tráfico gradualmente:

Una vez configurados los registros de DNS, comience a desplazar el tráfico aumentando el peso del ECS Express Mode y reduciendo proporcionalmente el peso de App Runner. Un enfoque recomendado:

- Configure el modo ECS Express en 10 y App Runner en 90
- Supervise y valide que el servicio gestione las solicitudes correctamente
- Aumente a 25/75
- Aumente a 50/50
- Aumente a 75/25
- Completa a 100/0

En cada paso, pruebe la aplicación antes de cambiar el tráfico adicional. Si se producen problemas en algún momento, retroceda ajustando las ponderaciones a sus valores anteriores.

> **Important**
>
> Mantén tu servicio App Runner en funcionamiento durante un período de validación (por ejemplo, de 24 a 48 horas) para confirmar que los cambios en el DNS se han propagado a nivel mundial y ofrecer una opción de reversión si es necesario. Si tienes problemas, puedes revertir rápidamente las ponderaciones de Route 53 a App Runner.

Paso 5: Completa la migración

Tras comprobar que el servicio ECS Express Mode gestiona correctamente el tráfico de producción y que ha pasado el período de validación, complete la migración:

- En Route 53, elimina el registro ponderado que apunta a App Runner (o establece su peso en 0).
- Elimine la asociación de dominio personalizada del servicio App Runner.

Elimine el servicio App Runner:

```
aws apprunner delete-service --service-arn your-app-runner-service-arn
```

Considere también la posibilidad de eliminar los recursos que ya no sean necesarios:

- Registros de enrutamiento ponderados de Route 53 para App Runner
- Imágenes de contenedores no utilizados de Amazon Elastic Container Registry
- Los roles de IAM se crearon específicamente para App Runner, si ya no se necesitan

> **Note**
>
> No elimine el servicio ECS Express Mode, su Application Load Balancer ni los recursos asociados si el servicio se ejecuta en producción.

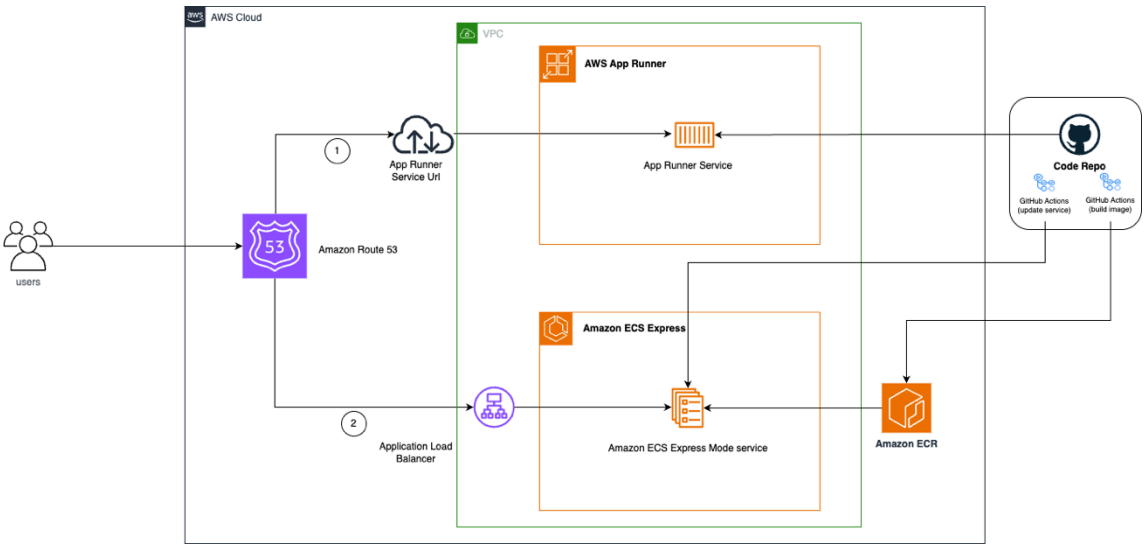
Migración de las implementaciones basadas en el origen

Si su servicio actual de App Runner se implementa desde el código fuente y no desde una imagen de contenedor, debe agregar un paso de contenerización antes de implementarlo en ECS Express Mode. A diferencia de App Runner, el modo ECS Express requiere una imagen de contenedor. Sin embargo, puede replicar la experiencia de implementación automatizada de App Runner mediante CI/CD herramientas como GitHub Actions con Amazon ECS Deploy Express Service GitHub Action.

El flujo de trabajo de migración consta de tres etapas:

1. Cree la imagen del contenedor mediante un Dockerfile
2. Inserte la imagen en un registro de contenedores, como Amazon Elastic Container Registry
3. Implemente la imagen en ECS Express Mode

El siguiente diagrama muestra cómo funciona este flujo de trabajo mediante GitHub Actions:



Coloque su aplicación en contenedores

Si su aplicación aún no tiene un Dockerfile, cree uno en la raíz del repositorio. El Dockerfile sirve como modelo para crear y empaquetar el código fuente en una imagen de contenedor.

La estructura de tu repositorio debe incluir:

```
your-app/
### src/                    # Application source code
### Dockerfile              # Container build instructions
### package.json            # Dependencies and scripts
### .github/                # GitHub configuration
    ### workflows/          # GitHub Actions workflows
        ### deploy.yml      # ECS Express Mode deployment workflow
```

Configure GitHub las acciones para el despliegue automatizado

Para replicar la implementación automática de App Runner mediante la inserción de código, configure GitHub las acciones con lo siguiente:

- Cree un proveedor de OpenID Connect (OIDC) para permitir que GitHub Actions asuma una función de IAM
- Cree un rol de IAM con permisos de ECS Express Mode y Amazon Elastic Container Registry
- Cree las dos funciones de IAM requeridas por ECS Express Mode
- Cree variables de GitHub entorno para sus recursos de ECS: `ECS_SERVICE`,`ECS_CLUSTER`, `AWS_REGION`,`AWS_ACCOUNT_ID`, y `ECR_REPOSITORY`

Ejemplo de flujo de trabajo de GitHub acciones

Cree un archivo de flujo de trabajo en `.github/workflows/deploy.yml`:

```
name: Build and Deploy to ECS

on:
  push:
    branches: [ main ]

env:
  AWS_REGION: ${{ vars.AWS_REGION }}
  AWS_ACCOUNT_ID: ${{ vars.AWS_ACCOUNT_ID }}
  ECR_REPOSITORY: ${{ vars.ECR_REPOSITORY }}
  ECS_SERVICE: ${{ vars.ECS_SERVICE }}
  ECS_CLUSTER: ${{ vars.ECS_CLUSTER }}

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    environment: production
    permissions:
      id-token: write
      contents: read

    steps:
    - name: Checkout
      uses: actions/checkout@v6

    - name: Configure AWS credentials
      uses: aws-actions/configure-aws-credentials@v5
      with:
        aws-region: ${{ env.AWS_REGION }}
        role-to-assume: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/github-actions-ecs-
role
        role-session-name: GitHubActionsECSDeployment

    - name: Login to Amazon ECR
      id: login-ecr
      uses: aws-actions/amazon-ecr-login@v2

    - name: Get short commit hash
      run: echo "IMAGE_TAG=${GITHUB_SHA:0:7}" >> $GITHUB_ENV
```

```
    - name: Build, tag, and push image to Amazon ECR
      id: build-image
      env:
        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
      uses: docker/build-push-action@v6
      with:
        context: .
        push: true
        tags: ${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:latest,
${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}

    - name: Deploy to ECS Express Mode
      uses: aws-actions/amazon-ecs-deploy-express-service@v1
      env:
        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
      with:
        service-name: ${{ env.ECS_SERVICE }}
        image: ${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}
        execution-role-arn: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/
ecsTaskExecutionRole
        infrastructure-role-arn: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/
ecsInfrastructureRoleForExpressServices
        cluster: ${{ env.ECS_CLUSTER }}
        container-port: 8080
        environment-variables: |
          [
            {"name": "ENV", "value": "Prod"}
          ]
        cpu: '1024'
        memory: '2048'
        health-check-path: /health
        min-task-count: 1
        max-task-count: 4
        auto-scaling-metric: AVERAGE_CPU
        auto-scaling-target-value: 70
```

Cuando insertas cambios de código en tu rama principal, GitHub Actions crea automáticamente una nueva imagen de contenedor, la envía a Amazon Elastic Container Registry y la implementa en tu servicio ECS Express Mode. Esto reproduce la experiencia de implementación automatizada que tuvo con App Runner.

Una vez que se ejecute el servicio ECS Express Mode, siga los pasos 3 y 5 del tutorial de migración para configurar el dominio personalizado, transferir el tráfico mediante el enrutamiento de DNS y completar la migración.

Recursos adicionales

- Cree su primer servicio de modo exprés mediante el AWS CLI
- Cree su primer servicio Amazon ECS Express Mode en la consola
- Actualización de recursos fuera del modo exprés
- the section called “Nombres de dominio personalizados”
- Enrutamiento ponderado de Amazon Route 53

Configuración de App Runner

Si es un AWS cliente nuevo, complete los requisitos previos de configuración que se indican en esta página antes de empezar a usarlos. AWS App Runner

Para estos procedimientos de configuración, utiliza el servicio AWS Identity and Access Management (IAM). Para obtener información completa sobre IAM, consulte los siguientes materiales de referencia:

- AWS Identity and Access Management (IAM)
- Guía del usuario de IAM

Inscríbase en una Cuenta de AWS

Para empezar AWS, necesitas un Cuenta de AWS. Para obtener información sobre cómo crear un Cuenta de AWS, consulte Cómo empezar con un Cuenta de AWS en la Guía de AWS Account Management referencia.

Siguientes pasos

Ha completado los pasos previos. Para implementar su primera aplicación en App Runner, consulteIntroducción.

Cómo empezar a usar App Runner

AWS App Runner es un AWS servicio que proporciona una forma rápida, sencilla y rentable de convertir directamente una imagen o un código fuente de un contenedor existente en un servicio web en ejecución en Nube de AWS.

En este tutorial, se explica cómo AWS App Runner implementar una aplicación en un servicio de App Runner. En él se explica cómo configurar el código fuente y la implementación, la compilación del servicio y el tiempo de ejecución del servicio. También muestra cómo implementar una versión de código, realizar un cambio de configuración y ver los registros. Por último, el tutorial muestra cómo limpiar los recursos que ha creado siguiendo los procedimientos del tutorial.

Temas

Requisitos previos

Antes de empezar el tutorial, asegúrese de realizar las siguientes acciones:

1. Complete los pasos de configuración que se indican enConfiguración.
2. Decide si quieres trabajar con un GitHub repositorio o con un repositorio de Bitbucket.
   - Para trabajar con un Bitbucket, primero crea una cuenta de Bitbucket, si aún no tienes una. Si eres nuevo en Bitbucket, consulta Cómo empezar a usar Bitbucket en la documentación de Bitbucket Cloud.
   - Para trabajar con ella GitHub, crea una GitHubcuenta, si aún no la tienes. Si es la primera vez que lo GitHub usas, consulta Cómo empezar con GitHub ella en los GitHubdocumentos.

> **Note**
>
> Puedes crear conexiones con varios proveedores de repositorios desde tu cuenta. Por lo tanto, si quieres realizar el despliegue desde un repositorio de Bitbucket GitHub y desde uno de ellos, puedes repetir este procedimiento. La próxima vez, crea un nuevo servicio de App Runner y crea una nueva conexión de cuenta para el otro proveedor de repositorios.

3. Crea un repositorio en tu cuenta de proveedor de repositorios. En este tutorial se usa el nombre del repositorio`python-hello`. Cree archivos en el directorio raíz del repositorio, con los nombres y el contenido especificados en los siguientes ejemplos.

Archivos para el repositorio de **`python-hello`** ejemplo

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
```

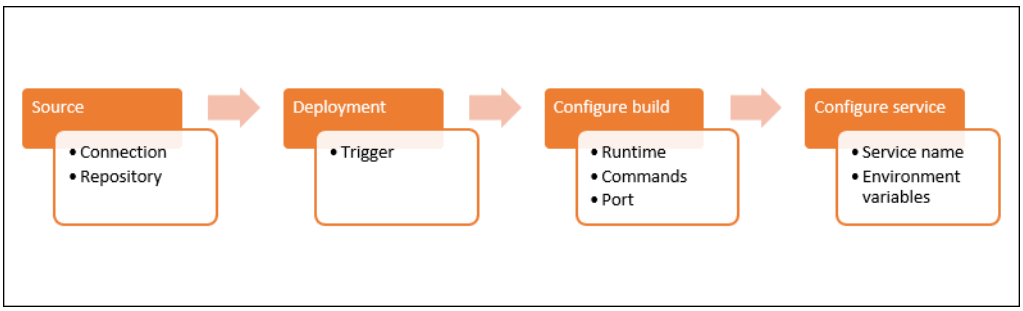
```
    server.serve_forever()
```

Paso 1: Crear un servicio de App Runner

En este paso, crearás un servicio de App Runner basado en el repositorio de código fuente de ejemplo que creaste en Bitbucket GitHub o del que formaste partethe section called “Requisitos previos”. El ejemplo contiene un sitio web de Python simple. Estos son los pasos principales que debe seguir para crear un servicio:

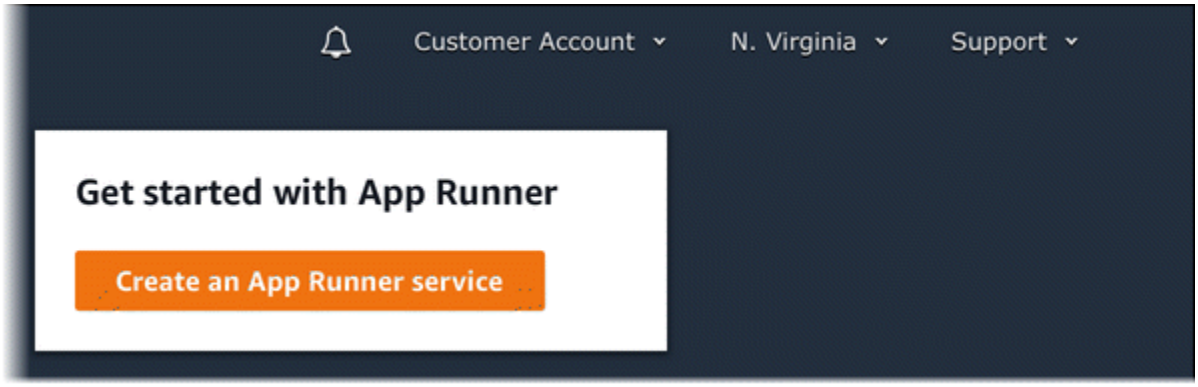
1. Configura tu código fuente.
2. Configure la implementación de código fuente.
3. Configure la creación de la aplicación.
4. Configure su servicio.
5. Revise y confirme.

En el siguiente diagrama se describen los pasos para crear un servicio de App Runner:

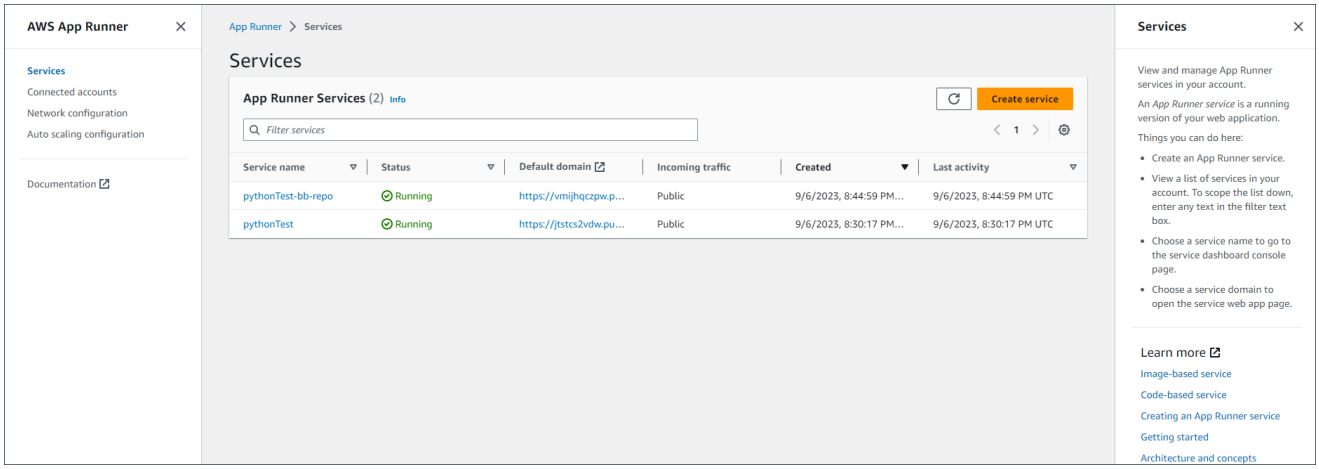


Para crear un servicio de App Runner basado en un repositorio de código fuente

1. Configura tu código fuente.
    a. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
    b. Si aún Cuenta de AWS no tiene ningún servicio de App Runner, aparecerá la página de inicio de la consola. Selecciona Crear un servicio de App Runner.



Si Cuenta de AWS tiene servicios existentes, se muestra la página de servicios con una lista de sus servicios. Elija Crear servicio.



c. En la página Origen e implementación, en la sección Fuente, para Tipo de repositorio, elija Repositorio de código fuente.

d. Seleccione un tipo de proveedor. Elige entre Bitbucket GitHubo Bitbucket.

e. A continuación, selecciona Añadir nuevo. Si se te solicita, proporciona tus credenciales GitHub o las de Bitbucket.

f. Elige el siguiente conjunto de pasos en función del tipo de proveedor que hayas seleccionado anteriormente.

> **Note**
>
> Los siguientes pasos para instalar el conector de AWS en su GitHub cuenta se realizan una sola vez. GitHub Puede reutilizar la conexión para crear varios servicios de App Runner basados en los repositorios de esta cuenta. Si ya tienes una conexión, selecciónela y pasa a la selección de repositorios.

Lo mismo ocurre con el conector de AWS de tu cuenta de Bitbucket. Si utilizas GitHub tanto Bitbucket como repositorios de código fuente para tus servicios de App Runner, tendrás que instalar un conector de AWS para cada proveedor. A continuación, podrás reutilizar cada conector para crear más servicios de App Runner.

- Para GitHubello, sigue estos pasos.

  i. En la siguiente pantalla, introduce un nombre de conexión.

  ii. Si es la primera vez que usas GitHub App Runner, selecciona Instalar otra.

  iii. En el cuadro de GitHub diálogo AWS Connector for, si se le solicita, elija el nombre de su GitHub cuenta.

  iv. Si se le solicita que autorice el AWS conector GitHub, elija Authorize AWS Connections.

  v. En el cuadro de GitHub diálogo Instalar AWS conector para, seleccione Instalar.

  El nombre de su cuenta aparece como la GitHub cuenta/organización seleccionada. Ahora puedes elegir un repositorio en tu cuenta.

  vi. En Repositorio, elige el repositorio de ejemplo que has creado,`python-hello`. En Branch, elige el nombre de sucursal predeterminado de tu repositorio (por ejemplo, principal).

  vii. Deje el directorio de origen con el valor predeterminado. El directorio predeterminado es la raíz del repositorio. Guardó el código fuente en el directorio raíz del repositorio en los pasos previos de los requisitos previos.

- En el caso de Bitbucket, sigue estos pasos.

  i. En la siguiente pantalla, introduce un nombre de conexión.

  ii. Si es la primera vez que utilizas Bitbucket con App Runner, selecciona Instalar otro.

  iii. En el cuadro de diálogo de AWS CodeStar solicitudes de acceso, puedes seleccionar tu espacio de trabajo y conceder el acceso a él AWS CodeStar para la integración con Bitbucket. Selecciona tu espacio de trabajo y, a continuación, selecciona Conceder acceso.

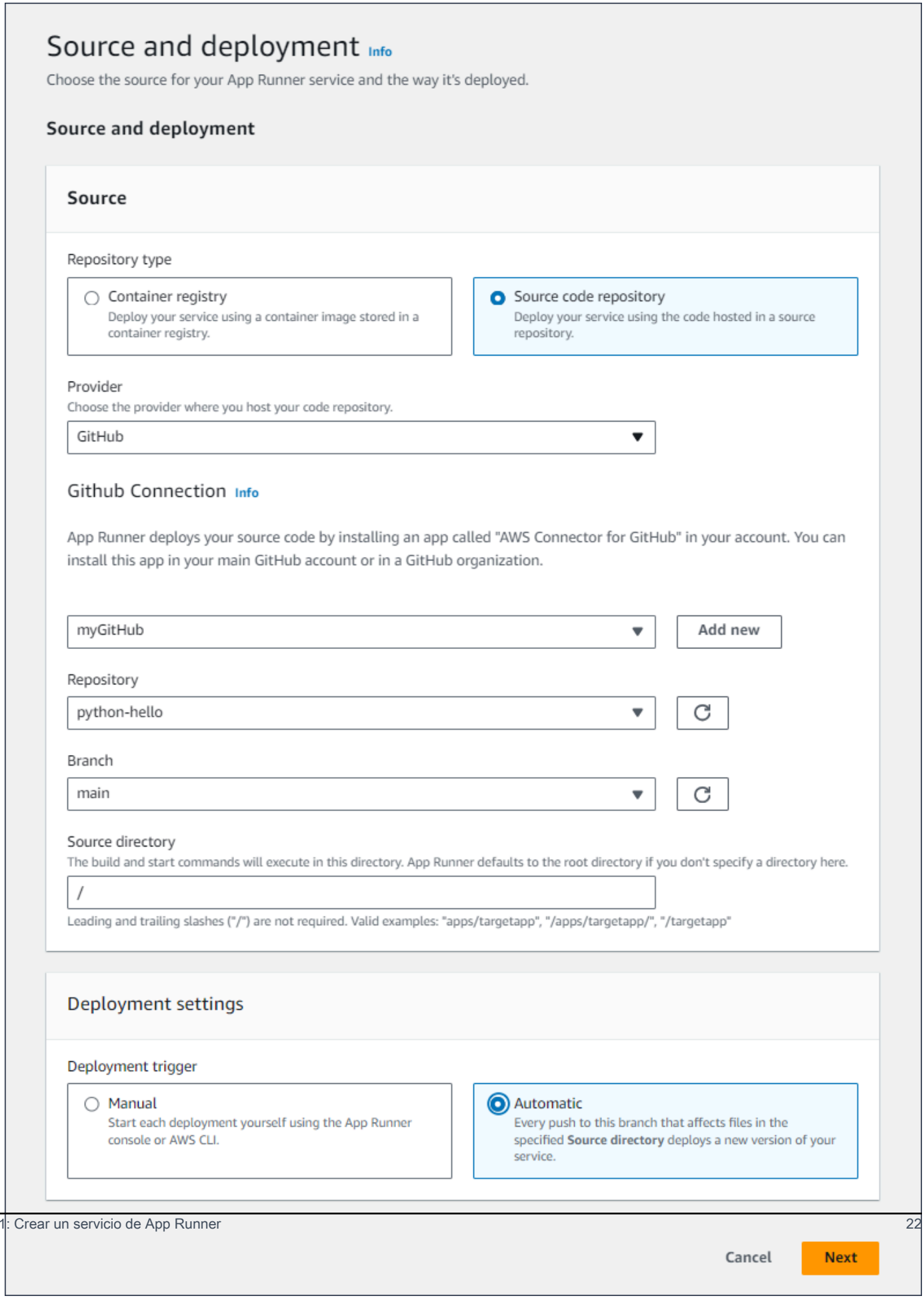
iv. A continuación, se te redirigirá a la AWS consola. Comprueba que la aplicación de Bitbucket esté configurada en el espacio de trabajo de Bitbucket correcto y selecciona Siguiente.

v. En Repositorio, elige el repositorio de ejemplo que has creado,. `python-hello` En Branch, elige el nombre de sucursal predeterminado de tu repositorio (por ejemplo, principal).

vi. Deje el directorio de origen con el valor predeterminado. El directorio predeterminado es la raíz del repositorio. Guardó el código fuente en el directorio raíz del repositorio en los pasos previos de los requisitos previos.

2. Configure sus despliegues: en la sección Configuración de despliegue, elija Automático y, a continuación, elija Siguiente.

> **Note**
>
> Con la implementación automática, cada nueva confirmación en el directorio de origen del repositorio implementa automáticamente una nueva versión del servicio.


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source and deployment
Source
Repository type
Container registry
Deploy your service using a container image stored in a container registry.
Source code repository
Deploy your service using the code hosted in a source repository.
Provider
Choose the provider where you host your code repository.
GitHub
Github Connection Info
App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.
myGitHub
Add new
Repository
python-hello
Branch
main
Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.
/
Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
Every push to this branch that affects files in the specified Source directory deploys a new version of your service.
Cancel
Next

3. Configure la creación de la aplicación.

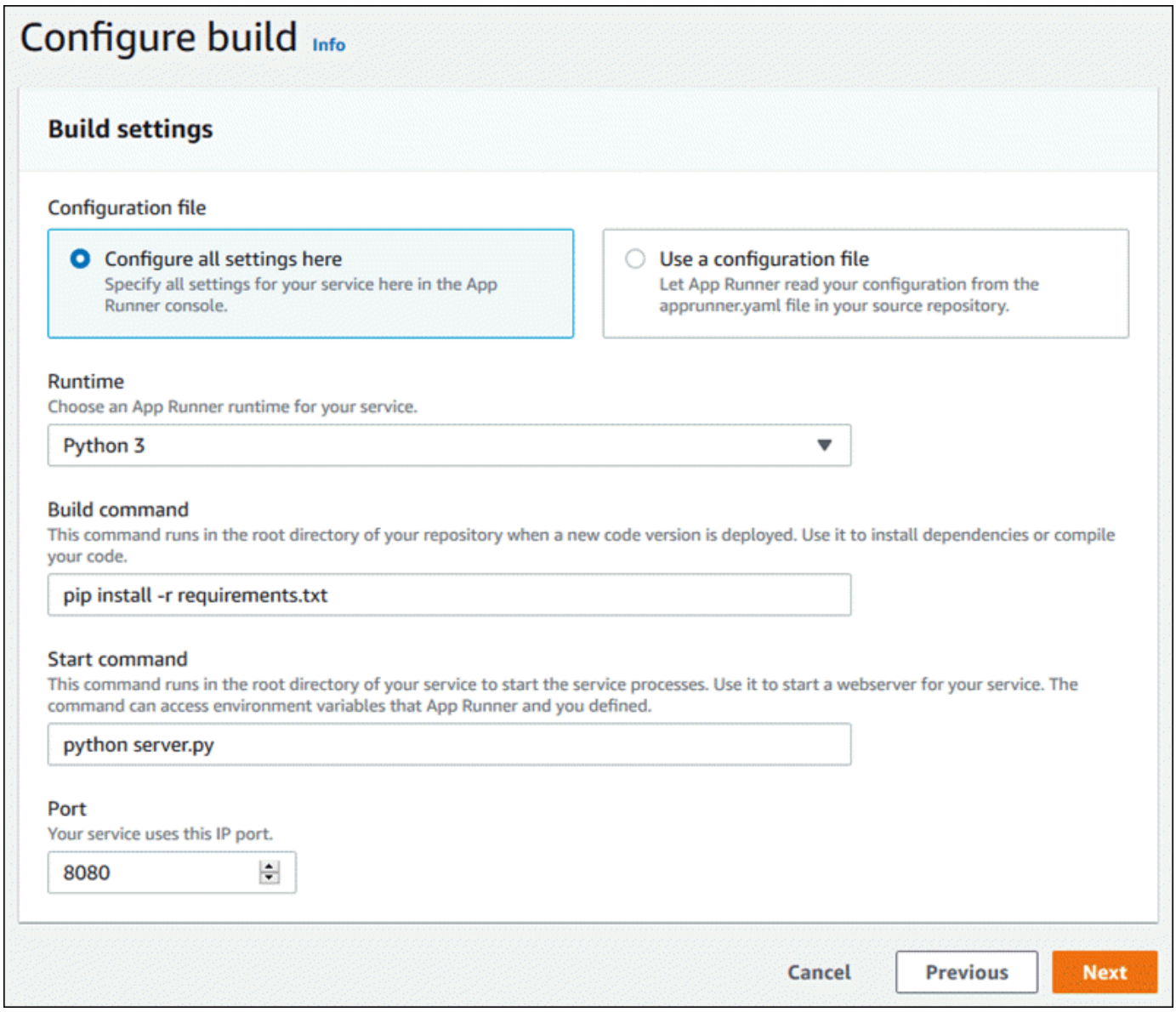
   a. En la página Configurar compilación, en Archivo de configuración, elija Configurar todos los ajustes aquí.

   b. Proporcione los siguientes ajustes de compilación:

      - Tiempo de ejecución: elija Python 3.
      - Comando de compilación: ingrese**`pip install -r requirements.txt`**.
      - Comando de inicio: Entrar**`python server.py`**.
      - Puerto: introduzca**`8080`**.

   c. Elija Siguiente.

> **Note**
>
> El motor de ejecución de Python 3 crea una imagen de Docker utilizando una imagen base de Python 3 y tu código de Python de ejemplo. A continuación, lanza un servicio que ejecuta una instancia contenedora de esta imagen.



4. Configure su servicio.

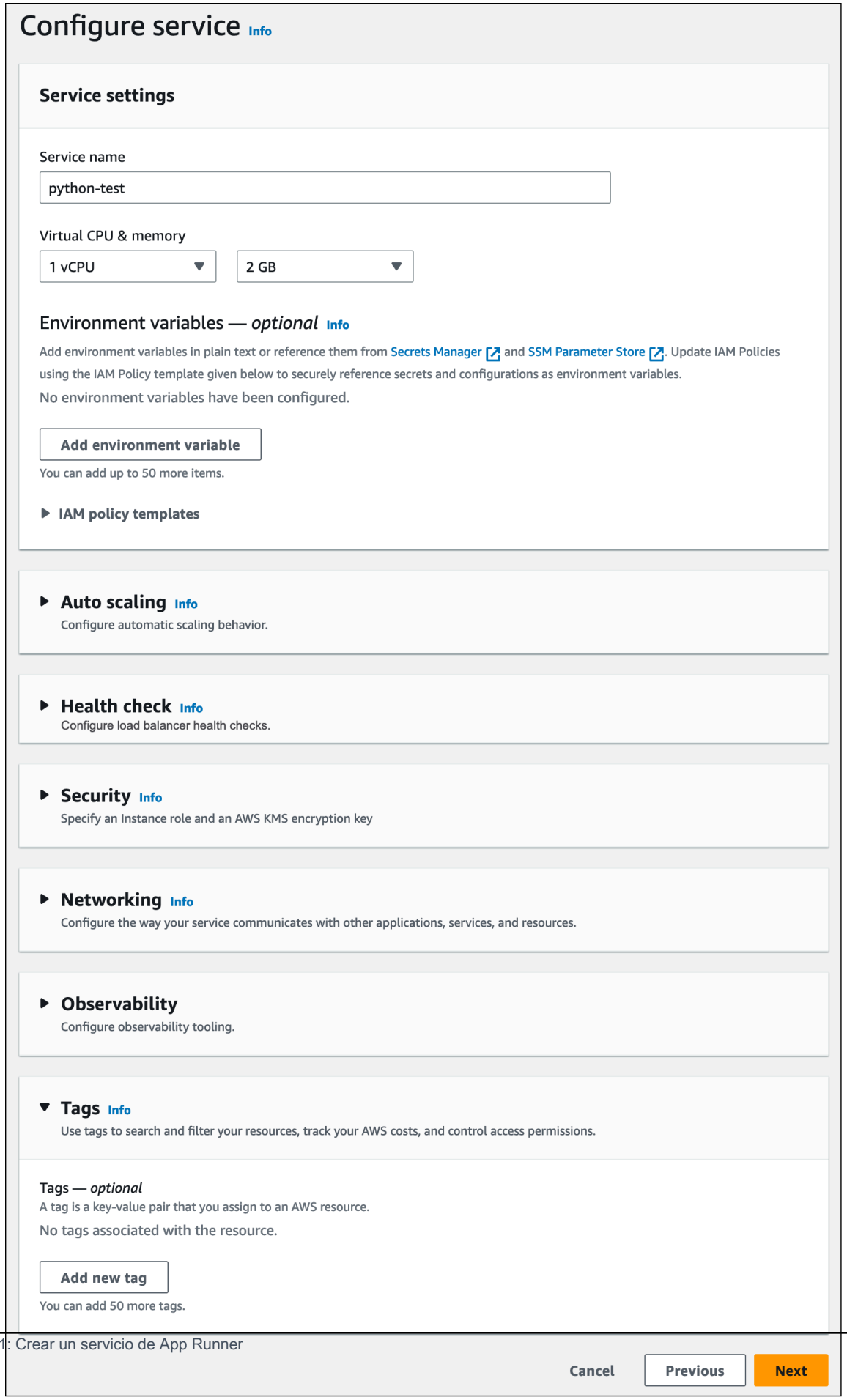
   a. En la página Configurar el servicio, en la sección Configuración del servicio, introduzca un nombre de servicio.

   b. En Variables de entorno, selecciona Añadir variable de entorno. Proporcione los siguientes valores para la variable de entorno.

      - Fuente: elija texto sin formato
      - Nombre de la variable de entorno: **`NAME`**
      - Valor de la variable de entorno: cualquier nombre (por ejemplo, su nombre).

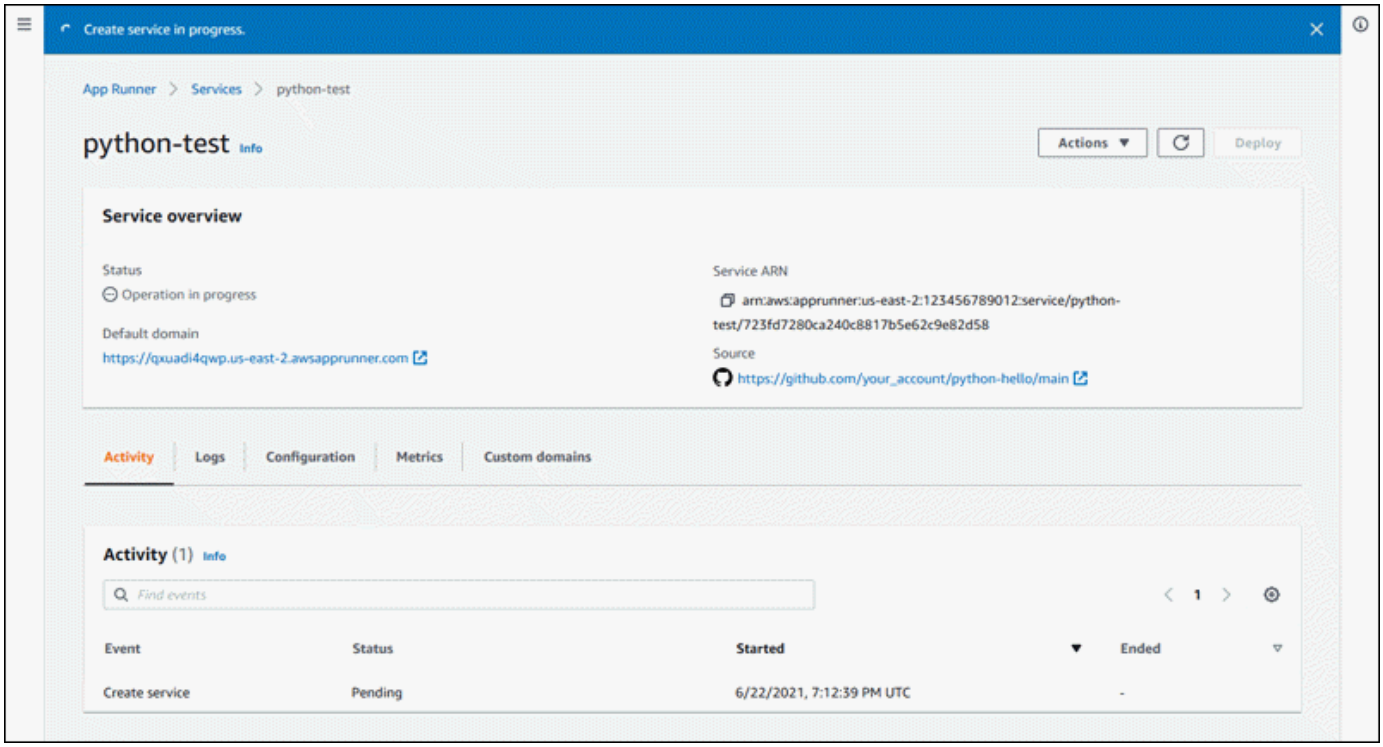
> **Note**
>
> La aplicación de ejemplo lee el nombre que ha establecido en esta variable de entorno y lo muestra en su página web.

c. Elija Siguiente.


Configure service Info
Service settings
Service name
python-test
Virtual CPU & memory
1 vCPU
2 GB
Environment variables — optional Info
Add environment variables in plain text or reference them from Secrets Manager and SSM Parameter Store. Update IAM Policies using the IAM Policy template given below to securely reference secrets and configurations as environment variables.
No environment variables have been configured.
Add environment variable
You can add up to 50 more items.
IAM policy templates
Auto scaling Info
Configure automatic scaling behavior.
Health check Info
Configure load balancer health checks.
Security Info
Specify an Instance role and an AWS KMS encryption key
Networking Info
Configure the way your service communicates with other applications, services, and resources.
Observability
Configure observability tooling.
Tags Info
Use tags to search and filter your resources, track your AWS costs, and control access permissions.
Tags — optional
A tag is a key-value pair that you assign to an AWS resource.
No tags associated with the resource.
Add new tag
You can add 50 more tags.
Cancel
Previous
Next

5. En la página Revisar y crear, compruebe todos los detalles que ha introducido y, a continuación, seleccione Crear e implementar.

   Si el servicio se ha creado correctamente, la consola muestra el panel de control del servicio, con una descripción general del nuevo servicio.



6. Verifica que el servicio esté en ejecución.

   a. En la página del panel de control del servicio, espere hasta que el estado del servicio sea En ejecución.

   b. Elige el valor de dominio predeterminado: es la URL del sitio web de tu servicio.

> **Note**
>
> Para aumentar la seguridad de las aplicaciones de App Runner, el dominio*.awsapprunner.com está registrado en la lista pública de sufijos (PSL). Para mayor seguridad, te recomendamos que utilices cookies con un `__Host-` prefijo si alguna vez necesitas configurar cookies confidenciales en el nombre de dominio predeterminado de tus aplicaciones de App Runner. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

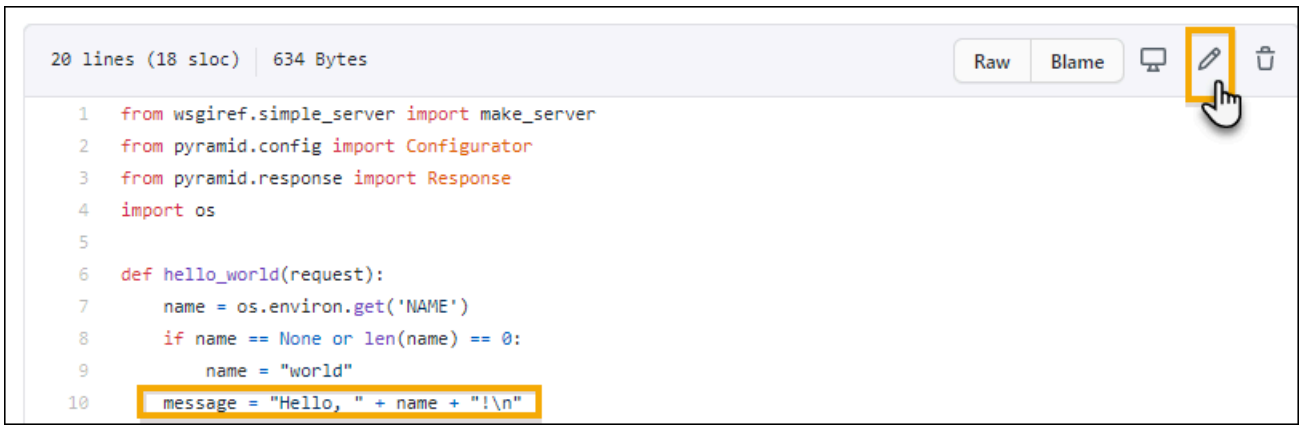
Aparece una página web: ¡Hola,*`your name`*!

Paso 2: Cambia el código de servicio

En este paso, realizas un cambio en tu código en el directorio fuente del repositorio. La función App CI/CD Runner crea e implementa automáticamente el cambio en tu servicio.

Para realizar un cambio en el código de servicio

1. Navega hasta tu repositorio de ejemplos.
2. Edita el archivo denominado`server.py`.
3. En la expresión asignada a la variable`message`, cambie el texto `Hello` a`Good morning`.
4. Guarde los cambios y confirme en el repositorio.
5. Los siguientes pasos ilustran el cambio del código de servicio en un GitHub repositorio.
    a. Navegue hasta su GitHub repositorio de ejemplo.
    b. Elige el nombre del archivo `server.py` para ir a ese archivo.
    c. Selecciona Editar este archivo (el icono del lápiz).
    d. En la expresión asignada a la variable`message`, cambie el texto `Hello` a`Good morning`.



e. Seleccione Confirmar cambios.

6. La nueva confirmación comienza a implementarse en tu servicio de App Runner. En la página del panel de control del servicio, el estado del servicio cambia a Operación en curso.

   Espere a que finalice la implementación. En la página del panel de control del servicio, el estado del servicio debería volver a ser En ejecución.

7. Compruebe que la implementación se ha realizado correctamente: actualice la pestaña del navegador donde se muestra la página web de su servicio.

   La página muestra ahora el mensaje modificado: ¡Buenos días***your name***!

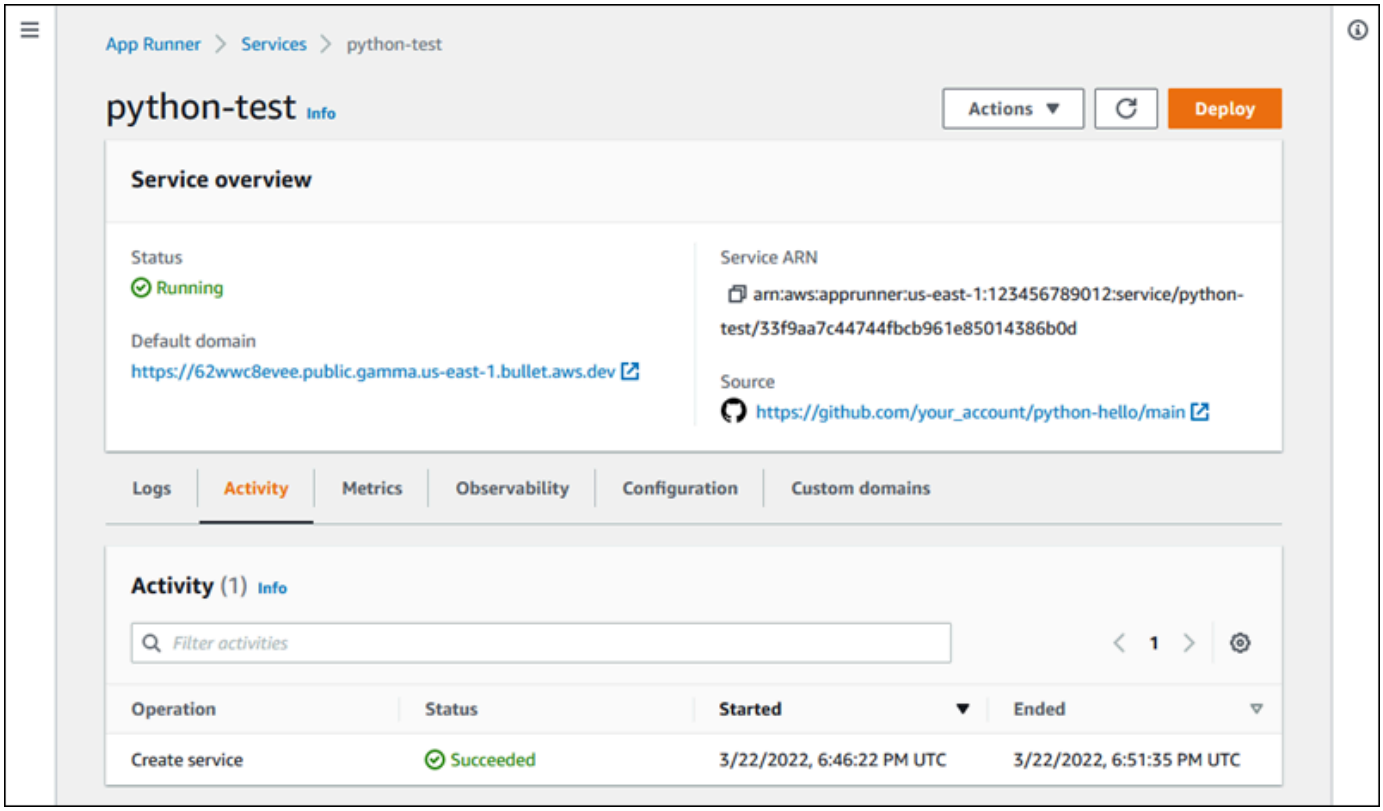
Paso 3: Realice un cambio de configuración

En este paso, realiza un cambio en el valor de la variable de **NAME** entorno para demostrar un cambio en la configuración del servicio.

Para cambiar el valor de una variable de entorno

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

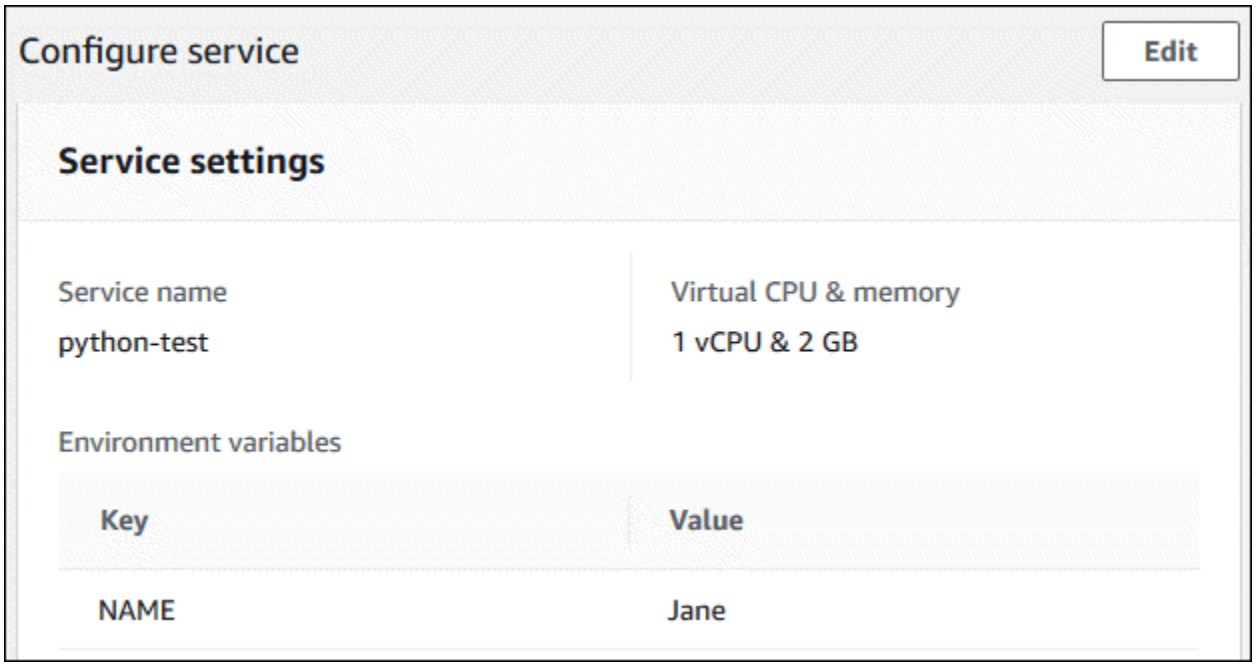
   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, seleccione la pestaña Configuración.

   La consola muestra los ajustes de configuración del servicio en varias secciones.

4. En la sección Configurar el servicio, elija Editar.



5. Para la variable de entorno con la clave**`NAME`**, cambie el valor por un nombre diferente.

6. Elija Apply changes.

   App Runner inicia el proceso de actualización. En la página del panel de control del servicio, el estado del servicio cambia a Funcionamiento en curso.

7. Espere a que finalice la actualización. En la página del panel de control del servicio, el estado del servicio debería volver a ser En ejecución.

8. Compruebe que la actualización se ha realizado correctamente: actualice la pestaña del navegador donde se muestra la página web de su servicio.

   La página ahora muestra el nombre modificado: ¡Buenos días,***new name***!

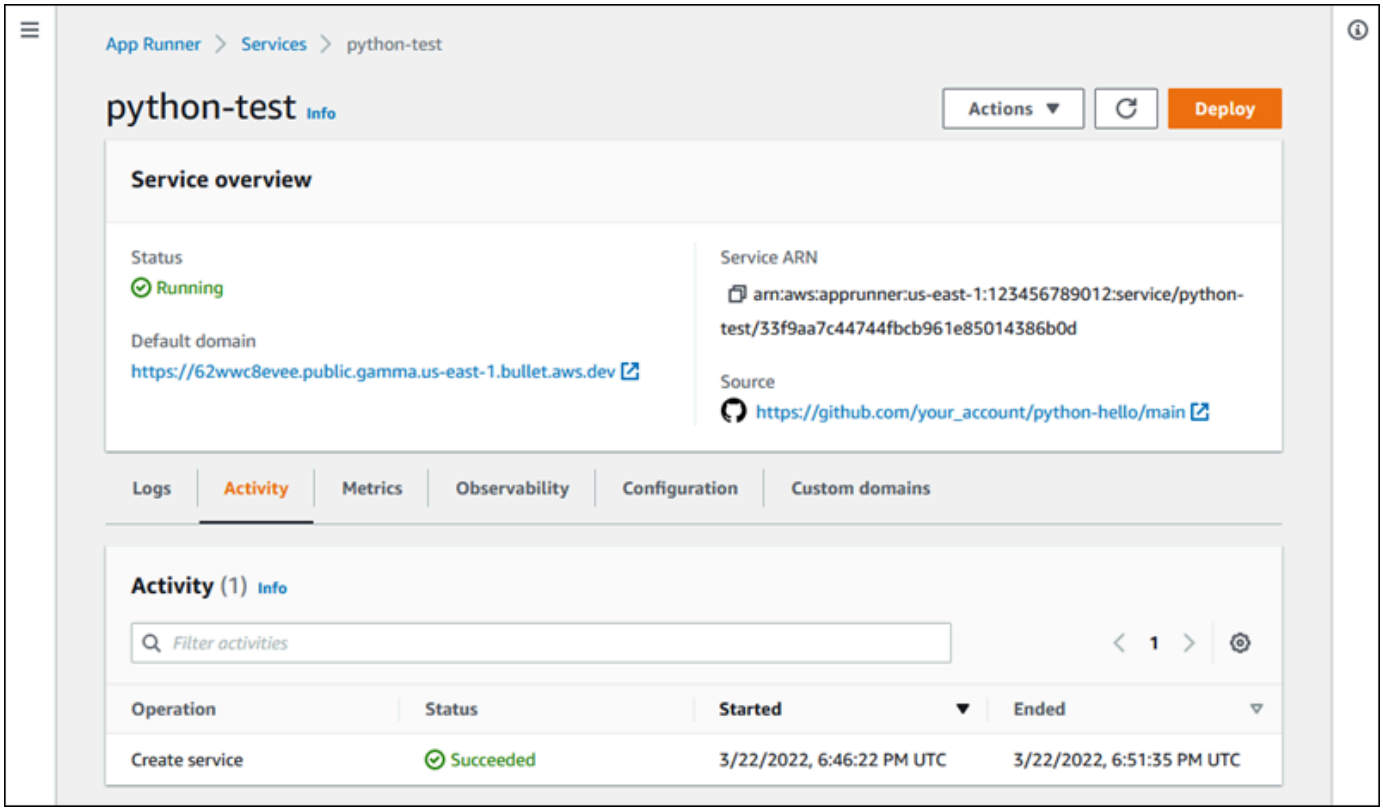
Paso 4: Ver los registros de su servicio

En este paso, utiliza la consola de App Runner para ver los registros de su servicio de App Runner. App Runner transmite los registros a Amazon CloudWatch Logs (CloudWatch Logs) y los muestra en el panel de control de su servicio. Para obtener información sobre los registros de App Runner, consultethe section called “Registros (CloudWatch registros)”.

Para ver los registros de su servicio

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

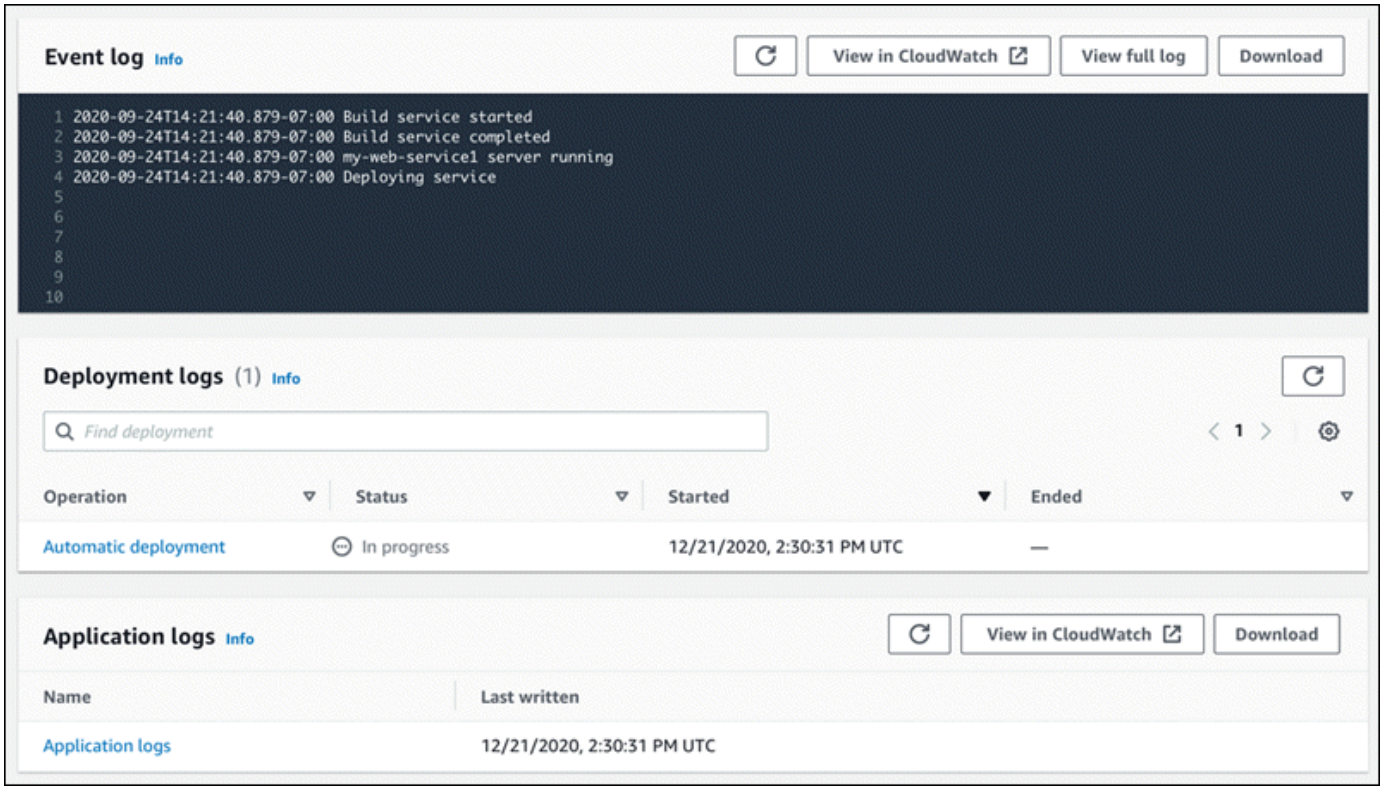
   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, selecciona la pestaña Registros.

   La consola muestra algunos tipos de registros en varias secciones:

   - Registro de eventos: actividad del ciclo de vida del servicio App Runner. La consola muestra los eventos más recientes.
   - Registros de implementación: orienta las implementaciones del repositorio a tu servicio de App Runner. La consola muestra un flujo de registros independiente para cada implementación.
   - Registros de aplicaciones: el resultado de la aplicación web que se implementó en el servicio App Runner. La consola combina los resultados de todas las instancias en ejecución en un único flujo de registro.



4. Para encontrar despliegues específicos, consulta la lista de registros de despliegues introduciendo un término de búsqueda. Puede buscar cualquier valor que aparezca en la tabla.
5. Para ver el contenido de un registro, elija Ver registro completo (registro de eventos) o el nombre del flujo de registro (registros de implementación y aplicación).
6. Seleccione Descargar para descargar un registro. Para un flujo de registro de despliegue, seleccione primero un flujo de registro.
7. Selecciona Ver en CloudWatch para abrir la CloudWatch consola y utilizar todas sus funciones para explorar los registros de servicio de App Runner. Para un flujo de registro de implementación, primero seleccione un flujo de registro.

> **Note**
>
> La CloudWatch consola resulta especialmente útil si desea ver los registros de aplicaciones de instancias específicas en lugar del registro de aplicaciones combinado.

Paso 5: Eliminar

Ahora has aprendido a crear un servicio de App Runner, ver los registros y realizar algunos cambios. En este paso, eliminas el servicio para eliminar los recursos que ya no necesitas.

Para eliminar tu servicio

1. En la página del panel de control del servicio, selecciona Acciones y, a continuación, selecciona Eliminar servicio.
2. En el cuadro de diálogo de confirmación, introduce el texto solicitado y, a continuación, selecciona Eliminar.

   Resultado: la consola navega a la página de servicios. El servicio que acaba de eliminar muestra el estado ELIMINANDO. Poco tiempo después, desaparece de la lista.

También puedes eliminar las conexiones GitHub y las conexiones de Bitbucket que creaste como parte de este tutorial. Para obtener más información, consulte the section called “Conexiones”.

Siguientes pasos

Ahora que has implementado tu primer servicio de App Runner, consulta los siguientes temas para obtener más información:

- Arquitectura y conceptos— La arquitectura, los conceptos principales y AWS los recursos relacionados con App Runner.
- Servicio basado en imágenesy Servicio basado en código — Los dos tipos de fuentes de aplicaciones que App Runner puede implementar.
- Desarrollando para App Runner— Cosas que debe saber al desarrollar o migrar el código de la aplicación para implementarlo en App Runner.
- Consola App Runner— Administra y supervisa tu servicio mediante la consola de App Runner.
- Administrar su servicio— Gestione el ciclo de vida de su servicio App Runner.
- Observabilidad— Obtenga visibilidad de las operaciones de su servicio de App Runner supervisando las métricas, leyendo los registros, gestionando los eventos, rastreando las llamadas a los servicios y rastreando los eventos de las aplicaciones, como las llamadas HTTP.

- Archivo de configuración de App Runner— Una forma basada en la configuración de especificar opciones para el comportamiento de compilación y tiempo de ejecución de tu servicio de App Runner.
- API de App Runner— Use la interfaz de programación de aplicaciones (API) de App Runner para crear, leer, actualizar y eliminar los recursos de App Runner.
- Seguridad— Las diferentes formas en AWS las que puedes garantizar la seguridad en la nube mientras utilizas App Runner y otros servicios.

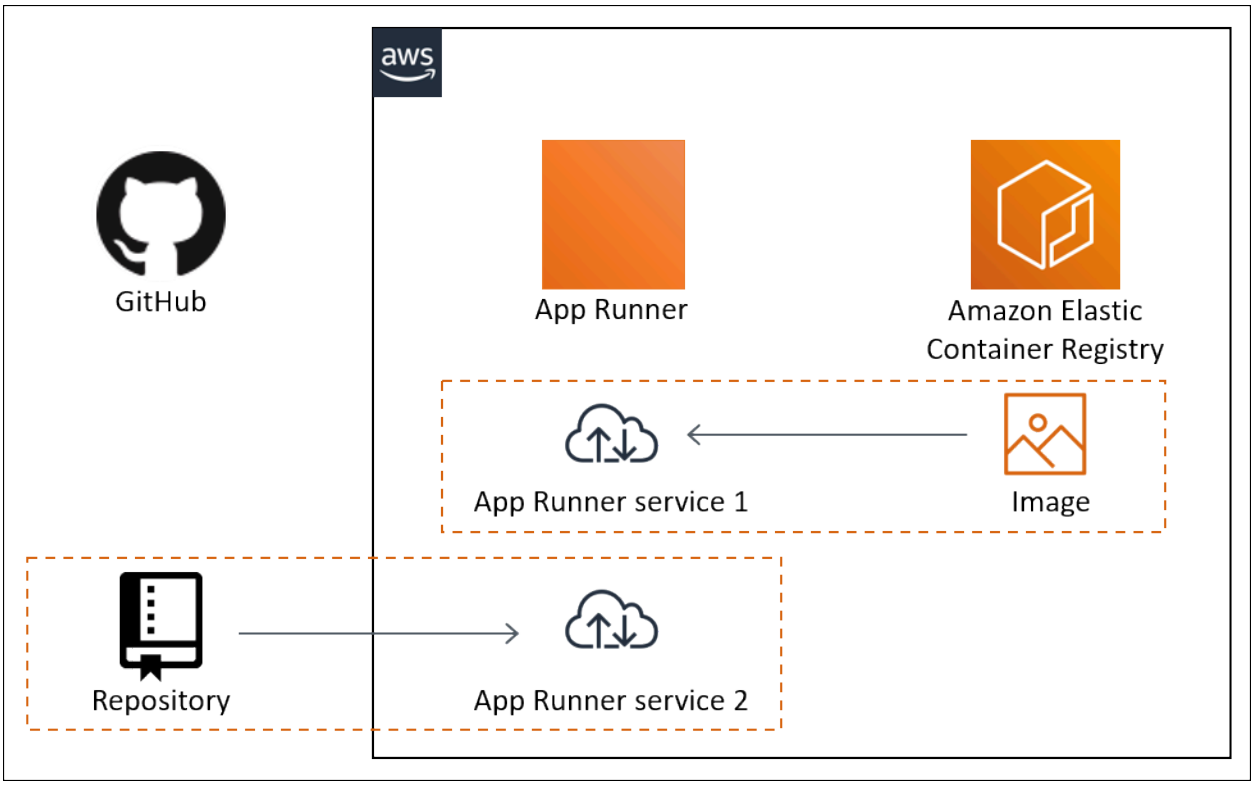
Arquitectura y conceptos de App Runner

AWS App Runner toma el código fuente o la imagen fuente de un repositorio y crea y mantiene un servicio web en ejecución para usted en el Nube de AWS. Normalmente, solo necesitas llamar a una acción de App Runner, CreateService, para crear tu servicio.

Con un repositorio de imágenes de origen, se proporciona una imagen de ready-to-use contenedor que App Runner puede implementar para ejecutar el servicio web. Con un repositorio de código fuente, se proporcionan el código y las instrucciones para crear y ejecutar un servicio web, y se orienta a un entorno de ejecución específico. App Runner es compatible con varias plataformas de programación, cada una con uno o más tiempos de ejecución gestionados para las versiones principales de la plataforma.

En este momento, App Runner puede recuperar el código fuente de un Bitbucket o un GitHubrepositorio, o puede recuperar la imagen fuente de Amazon Elastic Container Registry (Amazon ECR) en su. Cuenta de AWS

El siguiente diagrama muestra una descripción general de la arquitectura del servicio de App Runner. En el diagrama, hay dos servicios de ejemplo: uno implementa el código fuente desde Amazon GitHub ECR y el otro implementa una imagen fuente desde Amazon ECR. El mismo flujo se aplica al repositorio de Bitbucket.



Conceptos de App Runner

Los siguientes son conceptos clave relacionados con el servicio web que se ejecuta en App Runner:

- Servicio App Runner: AWS recurso que App Runner usa para implementar y administrar tu aplicación en función de su repositorio de código fuente o imagen de contenedor. Un servicio de App Runner es una versión en ejecución de la aplicación. Para obtener más información sobre la creación de un servicio, consultethe section called “Creación”.
- Tipo de fuente: el tipo de repositorio de fuentes que proporciona para implementar el servicio de App Runner: código fuente o imagen fuente.
- Proveedor de repositorios: el servicio de repositorio que contiene la fuente de la aplicación (por ejemplo GitHub, Bitbucket o Amazon ECR).
- Conexión a App Runner: AWS recurso que permite a App Runner acceder a la cuenta de un proveedor de repositorios (por ejemplo, una GitHub cuenta u organización). Para obtener más información acerca de las conexiones, consulte the section called “Conexiones”.
- Tiempo de ejecución: imagen base para implementar un repositorio de código fuente. App Runner proporciona una variedad de tiempos de ejecución gestionados para distintas plataformas y versiones de programación. Para obtener más información, consulte Servicio basado en código.
- Implementación: acción que aplica una versión del repositorio fuente (código o imagen) a un servicio de App Runner. La primera implementación en el servicio se produce como parte de la creación del servicio. Las implementaciones posteriores se pueden realizar de dos maneras:
  - Despliegue automático: una CI/CD capacidad. Puede configurar un servicio de App Runner para que compile e implemente automáticamente (para el código fuente) cada versión de la aplicación tal como aparece en el repositorio. Puede ser una nueva confirmación en un repositorio de código fuente o una nueva versión de imagen en un repositorio de imágenes fuente.
  - Implementación manual: una implementación en el servicio de App Runner que se inicia de forma explícita.
- Dominio personalizado: dominio que asocias a tu servicio de App Runner. Los usuarios de tu aplicación web pueden usar este dominio para acceder a tu servicio web en lugar del subdominio predeterminado de App Runner. Para obtener más información, consulte the section called “Nombres de dominio personalizados”.

> **Note**
>
> Para aumentar la seguridad de tus aplicaciones de App Runner, el dominio*.awsapprunner.com está registrado en la lista pública de sufijos (PSL). Para

mayor seguridad, te recomendamos que utilices cookies con un `__Host-` prefijo si alguna vez necesitas configurar cookies confidenciales en el nombre de dominio predeterminado de tus aplicaciones de App Runner. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

- Mantenimiento: actividad que App Runner realiza ocasionalmente en la infraestructura en la que se ejecuta tu servicio de App Runner. Cuando el mantenimiento está en curso, el estado del servicio cambia temporalmente a `OPERATION_IN_PROGRESS` (Operación en curso en la consola) durante unos minutos. Las acciones del servicio (por ejemplo, la implementación, la actualización de la configuración, la pausa/reanudación o la eliminación) se bloquean durante este tiempo. Vuelva a intentar la acción unos minutos más tarde, cuando el estado del servicio vuelva a ser. `RUNNING`

  > **Note**
  >
  > Si la acción falla, no significa que el servicio App Runner esté inactivo. Tu aplicación está activa y sigue gestionando las solicitudes. Es poco probable que su servicio sufra algún tiempo de inactividad.

  En concreto, App Runner migra tu servicio si detecta problemas en el hardware subyacente que aloja el servicio. Para evitar cualquier tiempo de inactividad del servicio, App Runner lo implementa en un nuevo conjunto de instancias y desplaza el tráfico hacia ellas (una implementación azul-verde). En ocasiones, es posible que veas un ligero aumento temporal en los cargos.

Configuraciones compatibles con App Runner

Al configurar un servicio de App Runner, se especifica la configuración de memoria y CPU virtual que se va a asignar al servicio. El pago se basa en la configuración de procesamiento que seleccione. Para obtener más información acerca de los precios, consulte Precios de Grupos de recursos de AWS.

En la siguiente tabla se proporciona información sobre las configuraciones de vCPU y memoria compatibles con App Runner:

CPU	Memoria
0,25 vCPU	0,5 GB
0,25 vCPU	1 GB
0,5 vCPU	1 GB
1 vCPU	2 GB
1 vCPU	3 GB
1 vCPU	4 GB
2 vCPU	4 GB
2 vCPU	6 GB
4 vCPU	8 GB
4 vCPU	10 GB
4 vCPU	12 GB

Recursos de App Runner

Cuando usas App Runner, creas y administras algunos tipos de recursos en tu Cuenta de AWS. Estos recursos se utilizan para acceder a tu código y administrar tus servicios.

La siguiente tabla proporciona una descripción general de estos recursos:

Nombre del recurso	Descripción
Service	Representa una versión en ejecución de la aplicación. Gran parte del resto de esta guía describe los tipos de servicio, la administración, la configuración y la supervisión. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :service/`*`service-name`*` [/`*`service-id`*` ]`

Nombre del recurso	Descripción
Connection	Proporciona a tus servicios de App Runner acceso a repositorios privados almacenados en proveedores externos. Existe como un recurso independiente para compartirlo entre varios servicios. Para obtener más información acerca de las conexiones, consulte the section called “Conexiones”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:connection/` *`connection-name`* `[/`*`connection-id`* `]`
AutoScalingConfiguration	Proporciona a los servicios de App Runner ajustes que controlan el escalado automático de la aplicación. Existe como un recurso independiente para compartirlo entre varios servicios. Para obtener más información sobre el escalado automático, consulte the section called “Escalado automático”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:autoscalingconfiguration/` *`config-name`* `[/`*`config-revision`* `[/`*`config-id`* `]]`
ObservabilityConfiguration	Configura funciones adicionales de observabilidad de aplicaciones para tus servicios de App Runner. Existe como un recurso independiente para compartirlo entre varios servicios. Para obtener más información sobre la configuración de la observabilidad, consultethe section called “Configuración de observabilidad”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:observabilityconfiguration/` *`config-name`* `[/`*`config-revision`* `[/`*`config-id`* `]]`

Nombre del recurso	Descripción
VpcConnector	Configura los ajustes de VPC para los servicios de App Runner. Existe como un recurso independiente para compartirlo entre varios servicios . Para obtener más información sobre la funcionalidad de la VPC, consulte. the section called “Tráfico saliente” ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :vpcconnector/` *`connector-name`* `[/`*`connector-revision`* `[/`*`connector-id`*` ]]`
VpcIngressConnection	Es un AWS App Runner recurso que se utiliza para configurar el tráfico entrante. Establece una conexión entre un punto final de la interfaz de VPC y el servicio App Runner, para que solo se pueda acceder al servicio de App Runner desde una Amazon VPC. Para obtener más información sobre la funcionalidad de VPCIngress Connection, consulte. the section called “Habilite el punto final privado” ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :vpcingressconnection/` *`vpc-ingress-connection-name`* `[/`*`connector-id`*` ]]`

Cuotas de recursos de App Runner

AWS impone algunas cuotas (también conocidas como límites) a tu cuenta para el uso de AWS recursos en cada una Región de AWS de ellas. En la siguiente tabla, se muestran las cuotas relacionadas con los recursos de App Runner. Las cuotas también se muestran en AWS App Runner los puntos finales y las cuotas en. Referencia general de AWS

Cuota de recursos	Descripción	Valor predeterminado	¿Ajustable?
Services	El número máximo de servicios que puedes crear en tu cuenta para cada uno Región de AWS.	30	✓ Sí

Cuota de recursos		Descripción	Valor predeterminado	¿Ajustable?
Connections		El número máximo de conexiones que puedes crear en tu cuenta para cada uno Región de AWS. Puede utilizar un solo conector en varios servicios.	10	✓ Sí
Auto scaling configurations	nombres	El número máximo de nombres únicos que puede tener en las configuraciones de autoescalado que cree en su cuenta para cada uno Región de AWS. Puede utilizar una única configuración de escalado automático en múltiples servicios.	10	✓ Sí
	revisiones por nombre	El número máximo de revisiones de configuración de autoescalado que puede crear en su cuenta Región de AWS para cada nombre único. Puede usar una sola revisión de configuración de autoescalado en varios servicios.	5	X No
Observability configurations	nombres	El número máximo de nombres únicos que puede tener en las configuraciones de observabilidad que cree en su cuenta para cada uno Región de AWS. Puede utilizar una única configuración de observabilidad en múltiples servicios.	10	✓ Sí
	revisiones por nombre	El número máximo de revisiones de la configuración de observabilidad que puedes crear en tu cuenta Región de AWS para cada nombre único. Puede usar una sola revisión de la configuración de observabilidad en varios servicios.	10	X No

Cuota de recursos	Descripción	Valor predeterminado	¿Ajustable?
VPC connectors	El número máximo de conectores de VPC que puede crear en su cuenta para cada uno. Región de AWS Puede utilizar un único conector de VPC en múltiples servicios.	10	✓ Sí
VPC Ingress Connection	El número máximo de conexiones de entrada de VPC que puede crear en su cuenta para cada una. Región de AWS Puede usar una única conexión de entrada de VPC para acceder a varios servicios de App Runner.	1	✗ No

La mayoría de las cuotas son ajustables y puedes solicitar un aumento de cuota para ellas. Para obtener más información, consulte Solicitar un aumento de cuota en la Guía del usuario de Service Quotas.

Servicio App Runner basado en una imagen de origen

Puede utilizarlos AWS App Runner para crear y administrar servicios basados en dos tipos de fuentes de servicios fundamentalmente diferentes: el código fuente y la imagen fuente. Independientemente del tipo de fuente, App Runner se encarga de iniciar, ejecutar, escalar y equilibrar la carga del servicio. Puedes usar la CI/CD capacidad de App Runner para realizar un seguimiento de los cambios en la imagen o el código fuente. Cuando App Runner descubre un cambio, compila automáticamente (para el código fuente) e implementa la nueva versión en tu servicio de App Runner.

En este capítulo se analizan los servicios basados en una imagen de origen. Para obtener información sobre los servicios basados en el código fuente, consulteServicio basado en código.

Una imagen de origen es una imagen contenedora pública o privada almacenada en un repositorio de imágenes. Apuntas App Runner a una imagen e inicia un servicio que ejecuta un contenedor basado en esta imagen. No es necesaria ninguna etapa de creación. Más bien, usted proporciona una ready-to-deploy imagen.

> **Note**
>
> Al proporcionar imágenes de contenedores, usted es responsable de actualizar y corregir estas imágenes con regularidad. Mientras App Runner administra la infraestructura, debes garantizar la seguridad y el up-to-date estado de las imágenes de contenedor proporcionadas. Para obtener más información, consulte la documentación de AWS App Runner

Proveedores de repositorios de imágenes

App Runner es compatible con los siguientes proveedores de repositorios de imágenes:

- Amazon Elastic Container Registry (Amazon ECR): almacena imágenes que son privadas para un. Cuenta de AWS
- Amazon Elastic Container Registry Public (Amazon ECR Public): almacena imágenes que se pueden leer públicamente.

Casos de uso de proveedores

- Uso de una imagen almacenada en Amazon ECR en tu cuenta AWS
- Uso de una imagen almacenada en Amazon ECR en una cuenta diferente AWS
- Uso de una imagen almacenada en Amazon ECR Public

Uso de una imagen almacenada en Amazon ECR en tu cuenta AWS

Amazon ECR almacena imágenes en repositorios. Hay repositorios públicos y privados. Para implementar la imagen en un servicio de App Runner desde un repositorio privado, App Runner necesita permiso para leer la imagen de Amazon ECR. Para conceder ese permiso a App Runner, debe proporcionarle a App Runner un rol de acceso. Se trata de un rol AWS Identity and Access Management (IAM) que cuenta con los permisos de acción de Amazon ECR necesarios. Cuando utilizas la consola de App Runner para crear el servicio, puedes elegir un rol existente en tu cuenta. Como alternativa, puedes usar la consola de IAM para crear un nuevo rol personalizado. O bien, puedes elegir que la consola de App Runner cree un rol para ti en función de las políticas administradas.

Cuando usas la API de App Runner o la AWS CLI, completas un proceso de dos pasos. En primer lugar, se utiliza la consola de IAM para crear un rol de acceso. Puedes usar una política administrada que proporcione App Runner o introducir tus propios permisos personalizados. A continuación, proporciona la función de acceso durante la creación del servicio mediante la acción de la CreateServiceAPI.

Para obtener información sobre la creación del servicio App Runner, consultethe section called “Creación”.

Uso de una imagen almacenada en Amazon ECR en una cuenta diferente AWS

Al crear un servicio de App Runner, puede usar una imagen almacenada en un repositorio de Amazon ECR que pertenezca a una AWS cuenta distinta a la que se encuentra su servicio. Hay algunas consideraciones adicionales que se deben tener en cuenta a la hora de utilizar una imagen de varias cuentas, además de las enumeradas en la sección anterior sobre una imagen de la misma cuenta.

- El repositorio multicuenta debe tener una política adjunta. La política del repositorio proporciona a tu rol de acceso permisos para leer las imágenes del repositorio. Usa la siguiente política para este

propósito. Sustituya la función del `Principal` elemento por el nombre de recurso de Amazon (ARN) de su función de acceso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::123456789012:role/MyApplicationRole"},
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
        ],
        "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/*"
    }
  ]
}
```

Para obtener información sobre cómo adjuntar una política de repositorio a un repositorio de Amazon ECR, consulte Configuración de una declaración de política de repositorio en la Guía del usuario de Amazon Elastic Container Registry.

- App Runner no admite la implementación automática de imágenes de Amazon ECR en una cuenta diferente a la cuenta en la que se encuentra tu servicio.

Uso de una imagen almacenada en Amazon ECR Public

Amazon ECR Public almacena imágenes legibles públicamente. Estas son las principales diferencias entre Amazon ECR y Amazon ECR Public que debes tener en cuenta en el contexto de los servicios de App Runner:

- Las imágenes públicas de Amazon ECR se pueden leer públicamente. No necesita proporcionar un rol de acceso al crear un servicio basado en una imagen pública de Amazon ECR. El repositorio no necesita ninguna política adjunta.
- App Runner no admite la implementación automática (continua) de imágenes públicas de Amazon ECR.

Lance un servicio directamente desde Amazon ECR Public

Puede lanzar directamente imágenes de contenedores de aplicaciones web compatibles que estén alojadas en la Galería pública de Amazon ECR como servicios web que se ejecutan en App Runner. Cuando navegue por la galería, busque Iniciar con App Runner en la página de la galería para ver una imagen. Una imagen con esta opción es compatible con App Runner. Para obtener más información sobre la galería, consulte Uso de la galería pública de Amazon ECR en la guía del usuario de Amazon ECR Public.

Para lanzar una imagen de galería como un servicio de App Runner

1. En la página de galería de una imagen, selecciona Iniciar con App Runner.

   Resultado: la consola de App Runner se abre en una nueva pestaña del navegador. La consola muestra el asistente de creación de servicios, con la mayoría de los nuevos detalles de servicio necesarios rellenados previamente.
2. Si desea crear el servicio en una AWS región distinta de la que muestra la consola, seleccione la región que aparece en el encabezado de la consola. A continuación, seleccione otra región.
3. En Puerto, introduzca el número de puerto en el que escucha la aplicación de imágenes. Por lo general, lo encontrarás en la página de la galería de la imagen.
4. Si lo desea, cambie cualquier otro detalle de configuración.
5. Seleccione Siguiente, revise la configuración y, a continuación, elija Crear e implementar.

Ejemplo de imagen

El equipo de App Runner mantiene la imagen de hello-app-runnerejemplo en una galería pública de Amazon ECR. Puedes usar este ejemplo para empezar a crear un servicio de App Runner basado en imágenes. Para obtener más información, consulte hello-app-runner.

Servicio App Runner basado en código fuente

Puede utilizarlos AWS App Runner para crear y gestionar servicios en función de dos tipos de fuente de servicio fundamentalmente diferentes: el código fuente y la imagen fuente. Independientemente del tipo de fuente, App Runner se encarga de iniciar, ejecutar, escalar y equilibrar la carga del servicio. Puedes usar la CI/CD capacidad de App Runner para realizar un seguimiento de los cambios en la imagen o el código fuente. Cuando App Runner descubre un cambio, compila automáticamente (para el código fuente) e implementa la nueva versión en tu servicio de App Runner.

En este capítulo se analizan los servicios basados en el código fuente. Para obtener información sobre los servicios basados en una imagen fuente, consulteServicio basado en imágenes.

El código fuente es el código de aplicación que App Runner crea e implementa para usted. Dirige App Runner a un directorio fuente de un repositorio de código y eliges un tiempo de ejecución adecuado que corresponda a una versión de la plataforma de programación. App Runner crea una imagen basada en la imagen base del motor de ejecución y en el código de la aplicación. A continuación, inicia un servicio que ejecuta un contenedor en función de esta imagen.

App Runner proporciona cómodos tiempos de ejecución gestionados específicos de la plataforma. Cada uno de estos tiempos de ejecución crea una imagen de contenedor a partir del código fuente y agrega dependencias de tiempo de ejecución del idioma a la imagen. No es necesario que proporciones instrucciones de configuración y compilación del contenedor, como un Dockerfile.

Los subtemas de este capítulo tratan sobre las distintas plataformas compatibles con App Runner, es decir, plataformas administradas que proporcionan tiempos de ejecución gestionados para diferentes versiones y entornos de programación.

Temas

- Uso de la plataforma Node.js de
- Uso de la plataforma Java
- Uso de la plataforma .NET de
- Uso de la plataforma PHP de
- Uso de la plataforma Ruby de
- Uso de la plataforma Go de

Proveedores de repositorios de código fuente

App Runner despliega el código fuente leyéndolo desde un repositorio de código fuente. App Runner es compatible con dos proveedores de repositorios de código fuente: GitHuby Bitbucket.

Implementación desde tu proveedor de repositorios de código fuente

Para implementar el código fuente en un servicio de App Runner desde un repositorio de código fuente, App Runner establece una conexión con él. Cuando utilizas la consola de App Runner para crear un servicio, proporcionas los detalles de la conexión y un directorio fuente para que App Runner despliegue tu código fuente.

Connections

Los detalles de conexión se proporcionan como parte del procedimiento de creación del servicio. Cuando utilizas la API de App Runner o la AWS CLI, una conexión es un recurso independiente. En primer lugar, se crea la conexión mediante la acción CreateConnectionde la API. A continuación, proporciona el ARN de la conexión durante la creación del servicio mediante la acción de la CreateServiceAPI.

Directorio de origen

Al crear un servicio, también se proporciona un directorio de origen. De forma predeterminada, App Runner usa el directorio raíz del repositorio como directorio de origen. El directorio fuente es la ubicación del repositorio de código fuente que almacena el código fuente y los archivos de configuración de la aplicación. Los comandos build e start también se ejecutan desde el directorio fuente. Cuando utilizas la API de App Runner o la AWS CLI para crear o actualizar un servicio, proporcionas el directorio de origen en las acciones de la UpdateServiceAPI CreateServicey la API. Para obtener más información, consulte la sección Directorio de fuentes siguiente.

Para obtener más información sobre la creación del servicio App Runner, consultethe section called “Creación”. Para obtener más información sobre las conexiones de App Runner, consultethe section called “Conexiones”.

Directorio de fuentes

Al crear un servicio de App Runner, puedes proporcionar el directorio de origen, junto con el repositorio y la rama. Defina el valor del campo Directorio fuente en la ruta del directorio del repositorio que almacena el código fuente y los archivos de configuración de la aplicación. App Runner ejecuta los comandos de compilación e inicio desde la ruta del directorio de origen que proporciones.

Introduzca el valor de la ruta del directorio de origen como absoluto desde el directorio del repositorio raíz. Si no especificas un valor, el valor predeterminado es el directorio de nivel superior del repositorio, también conocido como directorio raíz del repositorio.

También tienes la opción de proporcionar diferentes rutas de directorio de origen además del directorio del repositorio de nivel superior. Esto admite una arquitectura de repositorio monorepo, lo que significa que el código fuente de varias aplicaciones se almacena en un repositorio. Para crear y dar soporte a varios servicios de App Runner desde un único monorepo, especifica diferentes directorios de origen al crear cada servicio.

> **Note**
>
> Si especificas el mismo directorio de origen para varios servicios de App Runner, ambos servicios se implementarán y funcionarán de forma individual.

Si opta por utilizar un archivo de `apprunner.yaml` configuración para definir los parámetros del servicio, colóquelo en la carpeta del directorio de origen del repositorio.

Si la opción de activación del despliegue está configurada como Automático, los cambios que realices en el directorio de origen activarán un despliegue automático. Solo los cambios en la ruta del directorio de origen activarán una implementación automática. Es importante entender cómo afecta la ubicación del directorio de origen al alcance de una implementación automática. Para obtener más información, consulte las implementaciones automatizadas enMétodos de implementación.

> **Note**
>
> Si tu servicio de App Runner utiliza los tiempos de ejecución gestionados por PHP y quieres designar un directorio fuente distinto del repositorio raíz predeterminado, es importante que utilices la versión de tiempo de ejecución de PHP correcta. Para obtener más información, consulte Uso de la plataforma PHP de.

Plataformas gestionadas por App Runner

Las plataformas gestionadas de App Runner proporcionan tiempos de ejecución gestionados para varios entornos de programación. Cada tiempo de ejecución administrado facilita la creación y la ejecución de contenedores basados en una versión de un lenguaje de programación o entorno de ejecución. Cuando usas un tiempo de ejecución administrado, App Runner comienza con una imagen de tiempo de ejecución administrado. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene un paquete de ejecución de idiomas, así como algunas herramientas y paquetes de dependencias populares. App Runner usa esta imagen de tiempo de ejecución administrado como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*<language-name><major-version>*.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Fin del soporte para las versiones de tiempo de ejecución gestionado

Cuando el proveedor oficial o la comunidad de un entorno de ejecución en un lenguaje gestionado declara oficialmente que una versión ha finalizado su ciclo de vida útil (EOL), App Runner declara el estado de la versión como End of Support. Si su servicio se ejecuta en una versión de tiempo de ejecución de lenguaje administrado que ha llegado al final de Support, se aplican las siguientes políticas y recomendaciones.

Fin del soporte para una versión de ejecución de idiomas:

- Los servicios existentes seguirán funcionando y atendiendo el tráfico incluso si utilizan un tiempo de ejecución que haya llegado al final del soporte. Sin embargo, se ejecutarán en tiempos de ejecución no compatibles y ya no recibirán actualizaciones, parches de seguridad ni soporte técnico.
- Se siguen permitiendo las actualizaciones de los servicios existentes que utilizan tiempos de ejecución de End of Support, pero no recomendamos seguir utilizando los tiempos de ejecución de End of Support para un servicio.
- No se pueden crear nuevos servicios utilizando los tiempos de ejecución que han alcanzado la fecha de fin de Support.

Acciones necesarias para las versiones de ejecución de idiomas con estado de End of Support:

- Si el servicio se basa en una imagen original, no es necesario que realices ninguna otra acción para ese servicio.
- Si el servicio se basa en el código fuente, actualice la configuración del servicio para utilizar una versión de tiempo de ejecución compatible. Para ello, selecciona una versión de tiempo de ejecución compatible en la consola de App Runner, actualiza el `runtime` campo del archivo de configuración apprunner.yaml o usa las operaciones CreateService/UpdateServiceAPI o las herramientas IaC para configurar el parámetro. `runtime` Para obtener una lista de los tiempos de ejecución compatibles, consulta la página de información sobre la versión correspondiente a cualquier tiempo de ejecución específico de este capítulo.
- Como alternativa, puedes cambiar a la opción de fuente de imagen de contenedor de App Runner. Para obtener más información, consulte Servicio basado en imágenes.

> **Note**
>
> Si vas a pasar de Node.js 12, 14 o 16 a Node.js 22, o de Python 3.7 o 3.8 a Python 3.11, ten en cuenta que Node.js 22 y Python 3.11 utilizan un proceso de compilación revisado de App Runner que ofrece compilaciones más rápidas y eficientes. Para garantizar la compatibilidad antes de realizar la actualización, te recomendamos que consultes las instrucciones sobre el proceso de compilación de la siguiente sección.

En las siguientes tablas, se enumeran las versiones de tiempo de ejecución gestionado por App Runner que tienen una fecha de fin de soporte designada.

Versiones en ejecución	Fecha de fin de soporte de App Runner
Tiempos de ejecución compatibles con Python 3.8	1 de diciembre de 2025
Tiempos de ejecución compatibles con Python 3.7	1 de diciembre de 2025
Node.js 1.8 Tiempos de ejecución compatibles	1 de diciembre de 2025
Node.js 16 Tiempos de ejecución compatibles	1 de diciembre de 2025
Node.js 1.4 Tiempos de ejecución compatibles	1 de diciembre de 2025
Node.js 1.2 Tiempos de ejecución compatibles	1 de diciembre de 2025
.NET 6 *	1 de diciembre de 2025
PHP 8.1 *	31 de diciembre de 2025
Ruby 3.1 *	1 de diciembre de 2025
Ve 1 *	1 de diciembre de 2025

* App Runner no lanzará ninguna versión en ningún idioma nuevo para los tiempos de ejecución marcados con un asterisco (*). Estos tiempos de ejecución son los siguientes: .NET, PHP, Ruby

y Go. Si tienes un servicio basado en código configurado para estos tiempos de ejecución, te recomendamos que realices una de las siguientes acciones:

- Si corresponde, cambie la configuración del servicio a un entorno de ejecución gestionado compatible diferente.
- Como alternativa, puedes crear una imagen de contenedor personalizada con la versión de tiempo de ejecución que prefieras e implementarla con la Servicio basado en imágenes opción de App Runner. Puede alojar su imagen en Amazon ECR.

Versiones administradas en tiempo de ejecución y compilación de App Runner

App Runner ofrece un proceso de compilación actualizado para las aplicaciones que se ejecutan en los tiempos de ejecución de las versiones principales más recientes. Este proceso de compilación revisado es más rápido y eficiente. También crea una imagen final con un tamaño más reducido que solo contiene el código fuente, los artefactos de compilación y los tiempos de ejecución necesarios para ejecutar la aplicación.

Nos referimos al proceso de compilación más reciente como la versión revisada de App Runner y al proceso de compilación original como la versión original de App Runner. Para evitar cambios repentinos en las versiones anteriores de las plataformas de ejecución, App Runner solo aplica la compilación revisada a versiones de ejecución específicas, normalmente a las versiones principales recién publicadas.

Hemos introducido un nuevo componente en el archivo de `apprunner.yaml` configuración para que la compilación revisada sea compatible con versiones anteriores para un caso de uso muy específico y, además, para ofrecer más flexibilidad a la hora de configurar la compilación de la aplicación. Este es el `pre-run`parámetro opcional. Explicamos cuándo usar este parámetro junto con otra información útil sobre las compilaciones en las secciones siguientes.

En la siguiente tabla se indica qué versión de la compilación de App Runner se aplica a versiones específicas de tiempo de ejecución gestionado. Seguiremos actualizando este documento para manteneros informados sobre nuestros tiempos de ejecución actuales.

Plataforma	Versiones de ejecución	Proceso de compilación	Fecha de fin de soporte de App Runner
Python – Información de lanzamiento	Python 3.1 (!)	Revisado	
	Python 3.8	Original	1 de diciembre de 2025
	Python 3.7	Original	1 de diciembre de 2025
Node.js: Información de publicación	Node.js 22	Revisado	
	Node.js 18	Revisado	1 de diciembre de 2025
	Node.js 16	Original	1 de diciembre de 2025
	Node.js 14	Original	1 de diciembre de 2025
	Node.js 12	Original	1 de diciembre de 2025
Corretto — Información de lanzamiento	Corretto 11	Original	
	Corretto 8	Original	
.NET: Información sobre la versión	.NET 6 *	Original	1 de diciembre de 2025
PHP: Información sobre la versión	PHP 8.1 *	Original	31 de diciembre de 2025
Ruby: Información de lanzamiento	Ruby 3.1 *	Original	1 de diciembre de 2025

Plataforma	Versiones de ejecución	Proceso de compilación	Fecha de fin de soporte de App Runner
Go: Información sobre el lanzamiento	Ve 1 *	Original	1 de diciembre de 2025

> **Note**
>
> Algunos de los tiempos de ejecución listados incluyen una fecha de fin de Support. Para obtener más información, consulte the section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

> **Important**
>
> Python 3.11: tenemos recomendaciones específicas para la configuración de compilación de los servicios que utilizan el tiempo de ejecución gestionado de Python 3.11. Para obtener más información, consulte Llamadas para versiones de tiempo de ejecución específicas el tema de la plataforma Python.

Más información sobre las compilaciones y la migración de App Runner

Al migrar la aplicación a un entorno de ejecución más reciente que utilice la compilación revisada, es posible que tengas que modificar ligeramente la configuración de la compilación.

Para dar contexto a las consideraciones de migración, primero describiremos los procesos de alto nivel tanto para la compilación original de App Runner como para la versión revisada. Seguiremos con una sección en la que se describen los atributos específicos de tu servicio que podrían requerir algunas actualizaciones de configuración.

La versión original de App Runner

El proceso de creación de la aplicación App Runner original aprovecha el AWS CodeBuild servicio. Los pasos iniciales se basan en imágenes seleccionadas por el CodeBuild servicio. Sigue un

proceso de compilación de Docker que usa la imagen de tiempo de ejecución administrada de App Runner correspondiente como imagen base.

Los pasos generales son los siguientes:

1. Ejecute `pre-build` los comandos en una imagen CodeBuild seleccionada.

   Los `pre-build` comandos son opcionales. Solo se pueden especificar en el archivo `apprunner.yaml` de configuración.
2. Ejecute los `build` comandos utilizando CodeBuild la misma imagen del paso anterior.

   Los `build` comandos son obligatorios. Se pueden especificar en la consola de App Runner, en la API de App Runner o en el archivo `apprunner.yaml` de configuración.
3. Ejecuta una compilación de Docker para generar una imagen basada en la imagen de tiempo de ejecución gestionada por App Runner para tu plataforma y versión de ejecución específicas.
4. Copia el `/app` directorio de la imagen que generamos en el paso 2. El destino es la imagen basada en la imagen de tiempo de ejecución gestionada por App Runner, que generamos en el paso 3.
5. Vuelva a ejecutar los `build` comandos en la imagen de tiempo de ejecución gestionada por App Runner generada. Volvemos a ejecutar los comandos de compilación para generar artefactos de compilación a partir del código fuente del `/app` directorio que copiamos en él en el paso 4. App Runner implementará esta imagen más adelante para ejecutar el servicio web en un contenedor.

   Los `build` comandos son obligatorios. Se pueden especificar en la consola de App Runner, en la API de App Runner o en el archivo `apprunner.yaml` de configuración.
6. Ejecute `post-build` los comandos de la CodeBuild imagen del paso 2.

   Los `post-build` comandos son opcionales. Solo se pueden especificar en el archivo `apprunner.yaml` de configuración.

Una vez completada la compilación, App Runner implementa la imagen de tiempo de ejecución gestionada por App Runner generada en el paso 5 para ejecutar el servicio web en un contenedor.

La versión revisada de App Runner

El proceso de creación revisado es más rápido y eficiente que el proceso de creación original descrito en la sección anterior. Elimina la duplicación de los comandos de compilación que se producía en la compilación de la versión anterior. También crea una imagen final con un tamaño más

reducido que solo contiene el código fuente, los artefactos de compilación y los tiempos de ejecución necesarios para ejecutar la aplicación.

Este proceso de compilación utiliza una compilación de varias etapas de Docker. Los pasos generales del proceso son los siguientes:

1. Etapa de compilación: inicia un proceso de compilación de docker que ejecute `pre-build` y `build` comanda sobre las imágenes de compilación de App Runner.
   a. Copia el código fuente de la aplicación en el `/app` directorio.

      > **Note**
      >
      > Este `/app` directorio se designa como directorio de trabajo en todas las etapas de la compilación de Docker.

   b. Run Command `pre-build`.

      Los `pre-build` comandos son opcionales. Solo se pueden especificar en el archivo `apprunner.yaml` de configuración.

   c. Ejecute los `build` comandos.

      Los `build` comandos son obligatorios. Se pueden especificar en la consola de App Runner, en la API de App Runner o en el archivo `apprunner.yaml` de configuración.
2. Etapa de empaquetado: genera la imagen final del contenedor del cliente, que también se basa en la imagen de ejecución de App Runner.
   a. Copia el `/app` directorio de la etapa de compilación anterior a la nueva imagen de ejecución. Esto incluye el código fuente de la aplicación y los artefactos de compilación de la etapa anterior.
   b. Ejecute los `pre-run` comandos. Si necesita modificar la imagen en tiempo de ejecución fuera del `/app` directorio mediante los `build` comandos, añada los mismos comandos o los necesarios a este segmento del archivo de `apprunner.yaml` configuración.

      Se trata de un parámetro nuevo que se introdujo para admitir la versión revisada de App Runner.

      Los `pre-run` comandos son opcionales. Solo se pueden especificar en el archivo `apprunner.yaml` de configuración.

> **Notas**
>
> - Los `pre-run` comandos solo son compatibles con la versión revisada. No los añada al archivo de configuración si el servicio utiliza versiones en tiempo de ejecución que utilizan la compilación original.
> - Si no necesita modificar nada fuera del `/app` directorio con los `build` comandos, no necesita `pre-run` especificarlos.

3. Etapa posterior a la compilación: esta etapa se reanuda desde la etapa de compilación y ejecuta `post-build` los comandos.
   a. Ejecute los `post-build` comandos dentro del `/app` directorio.

      Los `post-build` comandos son opcionales. Solo se pueden especificar en el archivo `apprunner.yaml` de configuración.

Una vez completada la compilación, App Runner implementa la imagen de ejecución para ejecutar el servicio web en un contenedor.

> **Note**
>
> `apprunner.yaml`Al configurar el proceso de compilación, no se deje engañar por las `env` entradas de la sección Ejecutar. Aunque el parámetro de `pre-run` comando, al que se hace referencia en el paso 2 (b), reside en la sección Ejecutar, no utilices el `env` parámetro de la sección Ejecutar para configurar la compilación. Los `pre-run` comandos solo hacen referencia a las `env` variables definidas en la sección Compilación del archivo de configuración. Para obtener más información, consulte Sección de ejecución el capítulo del archivo de configuración de App Runner.

Requisitos de servicio para tener en cuenta la migración

Si el entorno de su aplicación tiene alguno de estos dos requisitos, tendrá que revisar la configuración de compilación añadiendo `pre-run` comandos.

- Si necesitas modificar algo fuera del `/app` directorio con los `build` comandos.

- Si necesita ejecutar los `build` comandos dos veces para crear el entorno requerido. Se trata de un requisito muy inusual. La gran mayoría de las versiones no lo harán.

Modificaciones fuera del **`/app`** directorio

- La versión revisada de App Runner asume que la aplicación no tiene dependencias fuera del `/app` directorio.
- Los comandos que proporciones con el `apprunner.yaml` archivo, la API de App Runner o la consola de App Runner deben generar artefactos de compilación en el `/app` directorio.
- Puedes modificar los `post-build` comandos `pre-buildbuild`, y para asegurarte de que todos los artefactos de compilación estén en el `/app` directorio.
- Si la aplicación requiere que la compilación modifique aún más la imagen generada para el servicio, fuera del `/app` directorio, puede usar los nuevos `pre-run` comandos del`apprunner.yaml`. Para obtener más información, consulte Configuración de las opciones de servicio de App Runner mediante un archivo de configuración.

Ejecutar los **`build`** comandos dos veces

- La versión original de App Runner ejecuta los `build` comandos dos veces, primero en el paso 2 y luego nuevamente en el paso 5. La versión revisada de App Runner corrige esta redundancia y solo ejecuta los `build` comandos una vez. Si tu aplicación requiere que los `build` comandos se ejecuten dos veces, algo inusual, en la versión revisada de App Runner se ofrece la opción de especificar y volver a ejecutar los mismos comandos mediante el `pre-run` parámetro. Al hacerlo, se conserva el mismo comportamiento de compilación doble.

Uso de la plataforma Python de

> **Important**
>
> App Runner dejará de ser compatible con Python 3.7 y Python 3.8 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La plataforma AWS App Runner Python proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una

versión de Python. Cuando utilizas un entorno de ejecución de Python, App Runner comienza con una imagen de tiempo de ejecución de Python gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de Python y algunas herramientas y paquetes de dependencias populares. App Runner usa esta imagen de tiempo de ejecución administrado como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

Para obtener nombres y versiones de tiempo de ejecución de Python válidos, consultethe section called “Información de lanzamiento”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de versión para tiempos de ejecución de Python: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Por ejemplo: `3.8.5`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `3.8`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `3.8.5`— Bloquear una versión de parche específica. App Runner no actualiza tu versión de ejecución.

Temas

- Configuración del tiempo de ejecución de Python
- Llamadas para versiones de tiempo de ejecución específicas

- Ejemplos de tiempo de ejecución de Python
- Información sobre la versión del motor de ejecución de Python

Configuración del tiempo de ejecución de Python

Al elegir un tiempo de ejecución administrado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.
- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Llamadas para versiones de tiempo de ejecución específicas

> **Note**
>
> App Runner ahora ejecuta un proceso de compilación actualizado para aplicaciones basado en las siguientes versiones de tiempo de ejecución: Python 3.11, Node.js 22 y Node.js 18. Si tu aplicación se ejecuta en alguna de estas versiones en tiempo de ejecución, consulta Versiones administradas en tiempo de ejecución y compilación de App Runner para obtener más información sobre el proceso de compilación revisado. Las aplicaciones que utilizan todas las demás versiones en tiempo de ejecución no se ven afectadas y siguen utilizando el proceso de compilación original.

Python 3.11 (versión revisada de App Runner)

Usa la siguiente configuración en apprunner.yaml para el tiempo de ejecución administrado de Python 3.11.

- Establece la clave de la sección superior en `runtime` `python311`

  Example

  ```
  runtime: python311
  ```

- Use el `pip3` en lugar de `pip` para instalar las dependencias.
- Utilice el `python3` intérprete en lugar de. `python`
- Ejecute el `pip3` instalador como un `pre-run` comando. Python instala las dependencias fuera del `/app` directorio. Dado que App Runner ejecuta la versión revisada de App Runner para Python 3.11, se perderá todo lo que se instale fuera del `/app` directorio mediante los comandos de la sección Compilación del `apprunner.yaml` archivo. Para obtener más información, consulte La versión revisada de App Runner.

  Example

  ```
  run:
    runtime-version: 3.11
    pre-run:
      - pip3 install pipenv
      - pipenv install
      - python3 copy-global-files.py
    command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  ```

Para obtener más información, consulte también el ejemplo de un archivo de configuración extendido para Python 3.11 más adelante en este tema.

Ejemplos de tiempo de ejecución de Python

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio de Python. El último ejemplo es el código fuente de una aplicación Python completa que se puede implementar en un servicio de tiempo de ejecución de Python.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es `3.7.7` y`3.11`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Python compatible, consultethe section called “Información de lanzamiento”.

Archivo de configuración mínimo de Python

En este ejemplo se muestra un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por Python. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consultethe section called “Ejemplos de archivos de configuración”.

Python 3.11 usa los `python3` comandos `pip3` y. Para obtener más información, consulte el ejemplo de un archivo de configuración extendido para Python 3.11 más adelante en este tema.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

Archivo de configuración de Python extendido

Este ejemplo muestra el uso de todas las claves de configuración con un entorno de ejecución gestionado por Python.

> **Note**
>
> La versión de tiempo de ejecución que se utiliza en estos ejemplos es`3.7.7`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Python compatible, consultethe section called “Información de lanzamiento”.

Python 3.11 usa los `python3` comandos `pip3` y. Para obtener más información, consulte el ejemplo de un archivo de configuración extendido para Python 3.11 más adelante en este tema.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Archivo de configuración de Python extendido: Python 3.11 (usa una compilación revisada)

En este ejemplo se muestra el uso de todas las claves de configuración con un entorno de ejecución gestionado por Python 3.11 en. `apprunner.yaml` En este ejemplo se incluye una `pre-run` sección, ya que esta versión de Python usa la versión revisada de App Runner.

El `pre-run` parámetro solo es compatible con la versión revisada de App Runner. No insertes este parámetro en el archivo de configuración si la aplicación usa versiones en tiempo de ejecución compatibles con la compilación original de App Runner. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

> **Note**
>
> La versión en tiempo de ejecución que se usa en estos ejemplos es*3.11*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Python compatible, consultethe section called “Información de lanzamiento”.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
```

```
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Fuente completa de la aplicación Python

En este ejemplo, se muestra el código fuente de una aplicación de Python completa que se puede implementar en un servicio de tiempo de ejecución de Python.

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
```

```
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
    server.serve_forever()
```

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
run:
  command: python server.py
```

Información sobre la versión del motor de ejecución de Python

> **Important**
>
> App Runner dejará de ser compatible con Python 3.7 y Python 3.8 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

En este tema se enumeran todos los detalles de las versiones de tiempo de ejecución de Python compatibles con App Runner.

Versiones de tiempo de ejecución compatibles: versión revisada de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Python 3.11 (python311)	3.11.14	SQLite 3.50,2
	3.11.13	SQLite 3.50,2
	3.11.13	SQLite 3.50,1
	3.11.12	SQLite 3.50,0

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	3.11.11	SQLite 3.49,1
	3.11.10	SQLite 3.46.1
	3.11.9	SQLite 3.46.1
	3.11.8	SQLite 3.45,2
	3.11.7	SQLite 3.44.2

Notas

- Python 3.11: tenemos recomendaciones específicas para la configuración de compilación de los servicios que utilizan el tiempo de ejecución gestionado de Python 3.11. Para obtener más información, consulte Llamadas para versiones de tiempo de ejecución específicas el tema de la plataforma Python.
- App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Versiones de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Python 3 (python3)	3.8.20	SQLite 3.50,2
	3.8.20	SQLite 3.50,1
	3.8.20	SQLite 3,5,0
	3.8.16	SQLite 3.46,1

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	3.8.15	SQLite 3.40,0
	3.7,16	SQLite 3.50,2
	3.7,16	SQLite 3.50,0
	3.7,15	SQLite 3.40,0
	3.7,10	SQLite 3.40,0

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Uso de la plataforma Node.js de

> **Important**
>
> App Runner dejará de ser compatible con Node.js 12, Node.js 14, Node.js 16 y Node.js 18 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La plataforma AWS App Runner Node.js proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una versión de Node.js. Cuando usas un entorno de ejecución de Node.js, App Runner comienza con una imagen de tiempo de ejecución de Node.js administrada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de Node.js y algunas herramientas. App Runner usa esta imagen de tiempo de ejecución administrada como

imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

Para obtener nombres y versiones de tiempo de ejecución de Node.js válidos, consultethe section called “Información de publicación”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de versión para los tiempos de ejecución de Node.js: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Por ejemplo: `22.14.0`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `22.14`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `22.14.0`— Bloquear una versión de parche específica. App Runner no actualiza tu versión en tiempo de ejecución.

Temas

- Configuración de tiempo de ejecución de Node.js
- Llamadas para versiones de ejecución específicas
- Ejemplos de tiempo de ejecución de Node.js
- Información de versión en tiempo de ejecución de Node.js

Configuración de tiempo de ejecución de Node.js

Al elegir un tiempo de ejecución administrado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.
- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

El suministro de un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

En el caso específico de los tiempos de ejecución de Node.js, también puedes configurar la compilación y el tiempo de ejecución mediante un archivo JSON cuyo nombre aparece `package.json` en la raíz del repositorio de origen. Con este archivo, puedes configurar la versión del motor Node.js, los paquetes de dependencias y varios comandos (aplicaciones de línea de comandos). Los administradores de paquetes como npm o yarn interpretan este archivo como entrada para sus comandos.

Por ejemplo:

- npm installinstala los paquetes definidos por el `devDependencies` nodo `dependencies` and en. `package.json`
- npm starto npm run start ejecuta el comando definido por el `scripts/start` nodo en`package.json`.

A continuación se muestra un ejemplo de un archivo `package.json`.

package.json

```
{
```

```
  "name": "node-js-getting-started",
  "version": "0.3.0",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "22.14.0"
  },
  "scripts": {
    "start": "node index.js",
    "test": "node test.js"
  },
  "dependencies": {
    "cool-ascii-faces": "^1.3.4",
    "ejs": "^2.5.6",
    "express": "^4.15.2"
  },
  "devDependencies": {
    "got": "^11.3.0",
    "tape": "^4.7.0"
  }
}
```

Para obtener más información`package.json`, consulte Crear un archivo package.json en el sitio web de npm Docs.

Consejos

- Si tu `package.json` archivo define un start comando, puedes usarlo como un run comando en el archivo de configuración de App Runner, como se muestra en el siguiente ejemplo.

  Example

  package.json

  ```
  {
    "scripts": {
      "start": "node index.js"
    }
  }
  ```

  apprunner.yaml

```
run:
  command: npm start
```

- Cuando ejecutas npm install en tu entorno de desarrollo, npm crea el archivo. `package-lock.json` Este archivo contiene una instantánea de las versiones del paquete que npm acaba de instalar. A partir de entonces, cuando npm instala dependencias, usa exactamente estas versiones. Si instalas yarn, se crea un archivo. `yarn.lock` Guarde estos archivos en su repositorio de código fuente para asegurarse de que su aplicación esté instalada con las versiones de las dependencias que desarrolló y con las que la probó.
- También puedes usar un archivo de configuración de App Runner para configurar la versión y el comando start de Node.js. Al hacer esto, estas definiciones anulan las que figuran en`package.json`. Un conflicto entre la `node` versión original `package.json` y el `runtime-version` valor del archivo de configuración de App Runner provoca un error en la fase de compilación de App Runner.

Llamadas para versiones de ejecución específicas

Node.js 22 y Node.js 18 (versión revisada de App Runner)

App Runner ahora ejecuta un proceso de compilación actualizado para aplicaciones basado en las siguientes versiones de tiempo de ejecución: Python 3.11, Node.js 22 y Node.js 18. Si tu aplicación se ejecuta en alguna de estas versiones en tiempo de ejecución, consulta Versiones administradas en tiempo de ejecución y compilación de App Runner para obtener más información sobre el proceso de compilación revisado. Las aplicaciones que utilizan todas las demás versiones en tiempo de ejecución no se ven afectadas y siguen utilizando el proceso de compilación original.

Ejemplos de tiempo de ejecución de Node.js

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio Node.js.

> **Note**
>
> La versión en tiempo de ejecución que se usa en estos ejemplos es`22.14.0`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Node.js compatible, consultethe section called “Información de publicación”.

Fichero de configuración mínimo de Node.js

En este ejemplo, se muestra un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por Node.js. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consultethe section called “Ejemplos de archivos de configuración”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

Archivo de configuración extendido de Node.js

En este ejemplo, se muestra el uso de todas las claves de configuración con un entorno de ejecución gestionado por Node.js.

> **Note**
>
> La versión de tiempo de ejecución que se utiliza en estos ejemplos es`22.14.0`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Node.js compatible, consultethe section called “Información de publicación”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
```

```
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Archivo de configuración extendido de Node.js: Node.js 22 (usa una versión revisada)

En este ejemplo se muestra el uso de todas las claves de configuración con un tiempo de ejecución gestionado por Node.js en `apprunner.yaml`. En este ejemplo se incluye una `pre-run` sección, ya que esta versión de Node.js usa la versión revisada de App Runner.

El `pre-run` parámetro solo es compatible con la versión revisada de App Runner. No insertes este parámetro en el archivo de configuración si la aplicación usa versiones en tiempo de ejecución compatibles con la compilación original de App Runner. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

> **Note**
>
> La versión en tiempo de ejecución que se usa en estos ejemplos es *22.14.0*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Node.js compatible, consultethe section called “Información de publicación”.

Example apprunner.yaml

```
version: 1.0
```

```
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 22.14.0
  pre-run:
    - node copy-global-files.js
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Aplicación Node.js con Grunt

En este ejemplo se muestra cómo configurar una aplicación Node.js desarrollada con Grunt. Grunt es un ejecutor de JavaScript tareas de línea de comandos. Ejecuta tareas repetitivas y gestiona la automatización de los procesos para reducir los errores humanos. Los complementos de Grunt y Grunt se instalan y administran mediante npm. Para configurar Grunt, debe incluir el `Gruntfile.js` archivo en la raíz de su repositorio de código fuente.

Example package.json

```
{
  "scripts": {
    "build": "grunt uglify",
    "start": "node app.js"
  },
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
```

```
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  },
  "dependencies": {
    "express": "^4.15.2"
  },
}
```

Example Gruntfile.js

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Load the plugin that provides the "uglify" task.
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).
  grunt.registerTask('default', ['uglify']);

};
```

Example apprunner.yaml

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es. *22.14.0* Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Node.js compatible, consultethe section called “Información de publicación”.

```
version: 1.0
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install grunt grunt-cli
      - npm install --only=dev
      - npm run build
    build:
      - npm install --production
run:
  runtime-version: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Información de versión en tiempo de ejecución de Node.js

Important

App Runner dejará de ser compatible con Node.js 12, Node.js 14, Node.js 16 y Node.js 18 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

Note

La política de obsolescencia estándar de App Runner consiste en dejar obsoleto un entorno de ejecución cuando algún componente importante del mismo llegue al final del soporte a largo plazo de la comunidad (LTS) y las actualizaciones de seguridad ya no estén disponibles. En algunos casos, App Runner puede retrasar la obsolescencia de un tiempo de ejecución durante un período limitado, más allá de la end-of-support fecha de la versión lingüística compatible con el tiempo de ejecución. Un ejemplo de este tipo de casos podría ser ampliar la compatibilidad con un entorno de ejecución para que los clientes tengan tiempo de migrar.

En este tema se enumeran todos los detalles de las versiones en tiempo de ejecución de Node.js compatibles con App Runner.

Versiones en tiempo de ejecución compatibles: versión revisada de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Node.js 22 (nodejs22)	22.21.1	npm 10.9.4, hilo 1.22.22
	22.20.0	npm 10.9.3, hilo 1.22.22
	22.17.0	npm 10.9.2, hilo 1.22.22
	22.16.0	npm 10.9.2, hilo 1.22.22
	22.14.0	npm 10.9.2, hilo 1.22.22

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Versiones de ejecución compatibles: versión revisada de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Node.js 18 (nodejs18)	18.20.8	npm 10.8.2, hilo 1.22.22
	18.20.7	npm 10.8.2, hilo 1.22.22
	18.20.6	npm 10.8.2, hilo 1.22.22
	18.20.5	npm 10.8.2, hilo 1.22.22

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	18.20.4	npm 10.7.0, hilo 1.22.22
	18.20.3	npm 10.7.0, hilo 1.22.22
	18.20.2	npm 10, hilo *
	18.19.1	npm 10, hilo *
	18.19.0	npm 10, hilo *

Versiones de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Node.js 16 (nodejs16)	16.20.2	npm 8.19.4, hilo 1.22.22
	16.20.1	npm 8.19.4, hilo *
	16.20.0	npm 8.19.4, hilo *
	16.19.1	npm 8.19.4, hilo *
	16.19.0	npm 8.19.4, hilo *
	16.18.1	npm 8.19.4, hilo *
	16.17.1	npm 8.19.4, hilo *
	16.17.0	npm 8.19.4, hilo *
Node.js 14 (nodejs14)	14.21.3	npm 6.14.18, hilo 1.22.22
	14.21.2	npm 6.14.18, hilo *
	14.21.1	npm 6.14.18, hilo *

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	14.20.1	npm 6.14.18, hilo *
	14.19.0	npm 6.14.18, hilo *
Node.js 12 (nodejs12)	12.22.12	npm 6.14.16, hilo 1.22.22
	12.21.0	npm 6.14.16, hilo *

Uso de la plataforma Java

La plataforma AWS App Runner Java proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una versión de Java. Cuando usas un entorno de ejecución de Java, App Runner comienza con una imagen de tiempo de ejecución de Java gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de Java y algunas herramientas. App Runner usa esta imagen de tiempo de ejecución administrada como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

En este momento, todos los tiempos de ejecución de Java compatibles se basan en Amazon Corretto. Para ver los nombres y las versiones de los tiempos de ejecución de Java válidos, consulte. the section called “Información de lanzamiento”

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de la versión para los tiempos de ejecución de Amazon Corretto:

Tiempo de ejecución	Sintaxis	Ejemplo
corretto11	`11.0[.`*`openjdk-update`* `[.`*`openjdk-build`* `[.`*`corretto-specific-revision`* `]]]`	`11.0.13.08.1`
corretto8	`8[.`*`openjdk-update`* `[.`*`openjdk-build`* `[.`*`corretto-specific-revision`* `]]]`	`8.312.07.1`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `11.0.13`— Bloquee la versión de actualización de Open JDK. App Runner actualiza solo las versiones de nivel inferior de Open JDK y Amazon Corretto.
- `11.0.13.08.1`— Bloquear una versión específica. App Runner no actualiza tu versión en tiempo de ejecución.

Temas

- Configuración del tiempo de ejecución de Java
- Ejemplos de entornos de ejecución de Java
- Información sobre la versión de Java Runtime

Configuración del tiempo de ejecución de Java

Al elegir un tiempo de ejecución gestionado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.

- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Ejemplos de entornos de ejecución de Java

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio Java. El último ejemplo es el código fuente de una aplicación Java completa que puede implementar en un servicio de tiempo de ejecución de Corretto 11.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es*`11.0.13.08.1`*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de ejecución de Java compatible, consultethe section called “Información de lanzamiento”.

Archivo de configuración Minimal Corretto 11

Este ejemplo muestra un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por Corretto 11. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consulte.

Example apprunner.yaml

```
version: 1.0
runtime: corretto11
build:
  commands:
    build:
      - mvn clean package
run:
```

```
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
```

Archivo de configuración de Corretto 11 extendido

Este ejemplo muestra cómo puede utilizar todas las claves de configuración con un tiempo de ejecución gestionado por Corretto 11.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es*11.0.13.08.1*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de ejecución de Java compatible, consultethe section called “Información de lanzamiento”.

Example apprunner.yaml

```
version: 1.0
runtime: corretto11
build:
  commands:
    pre-build:
      - yum install some-package
      - scripts/prebuild.sh
    build:
      - mvn clean package
    post-build:
      - mvn clean test
  env:
    - name: M2
      value: "/usr/local/apache-maven/bin"
    - name: M2_HOME
      value: "/usr/local/apache-maven/bin"
run:
  runtime-version: 11.0.13.08.1
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Fuente completa de la aplicación Corretto 11

Este ejemplo muestra el código fuente de una aplicación Java completa que puede implementar en un servicio de tiempo de ejecución de Corretto 11.

Example src/main/java/com/HelloWorld/HelloWorld.java

```
package com.HelloWorld;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorld {

    @RequestMapping("/")
    public String index(){
        String s = "Hello World";
        return s;
    }
}
```

Example src/main/java/com/HelloWorld/Main.java

```
package com.HelloWorld;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Main {

    public static void main(String[] args) {

        SpringApplication.run(Main.class, args);
    }
}
```

Example aprunner.yaml

```
version: 1.0
runtime: corretto11
```

```
build:
  commands:
    build:
      - mvn clean package
run:
  command: java -Xms256m -jar target/HelloWorldJavaApp-1.0-SNAPSHOT.jar .
  network:
    port: 8080
```

Example pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
    <relativePath/>
  </parent>
  <groupId>com.HelloWorld</groupId>
  <artifactId>HelloWorldJavaApp</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
```

```
          <groupId>org.junit.vintage</groupId>
          <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Información sobre la versión de Java Runtime

En este tema se enumeran todos los detalles de las versiones de ejecución de Java compatibles con App Runner.

Versiones de tiempo de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Corretto 11 (correcto 11)	11.0.28.6.1	Maven 3.9.11, Gradle 6.9.4
	11.0.27.6.1	Maven 3.9.10, Gradle 6.9.4
	11.0.27.6.1	Maven 3.9.9, Gradle 6.9.4
	11.0.26.4.1	Maven 3.9.9, Gradle 6.9.4

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	11.0.25.9.1	Maven 3.9.9, Gradle 6.9.4
	11.0.24.8.1	Maven 3.9.9, Gradle 6.9.4
	11.0.23.9.1	Maven 3.9.8, Gradle 6.9.4
	11.0.22.7.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.5, Gradle 6.9.4
	11.0.20.8.1	Maven 3.9.3, Gradle 6.9.4
	11.0.19.7.1	Maven 3.9.3, Gradle 6.9.4
	11.0.18.10.1	Maven 3.9.1, Gradle 6.9.4
	11.0.17.8.1	Maven 3.8.6, Gradle 6.9.3
	11.0.16.9.1	Maven 3.8.6, Gradle 6.9.2
	11.0.13.08.1	Maven 3.6.3, Gradle 6.5
Corretto 8 (correcto 8)	8.472,08.1	Maven 3.9.11, Gradle 6.9.4
	8.462.08.1	Maven 3.9.11, Gradle 6.9.4
	8.452.09.2	Maven 3.9.10, Gradle 6.9.4
	8.452.09.2	Maven 3.9.9, Gradle 6.9.4
	8.452.09.1	Maven 3.9.9, Gradle 6.9.4
	8.442.06.1	Maven 3.9.9, Gradle 6.9.4
	8.432.06.1	Maven 3.9.9, Gradle 6.9.4
	8.422.05.1	Maven 3.9.9, Gradle 6.9.4

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	8.412.08.1	Maven 3.9.8, Gradle 6.9.4
	8.402.08.1	Maven 3.9.6, Gradle 6.9.4
	8.392.08.1	Maven 3.9.6, Gradle 6.9.4
	8.382.05.1	Maven 3.9.4, Gradle 6.9.4
	8.372.07.1	Maven 3.9.3, Gradle 6.9.4
	8.362.08.1	Maven 3.9.1, Gradle 6.9.4
	8.352.08.1	Maven 3.8.6, Gradle 6.9.3
	8.342.07.4	Maven 3.8.6, Gradle 6.9.2
	8.312.07.1	Maven 3.6.3, Gradle 6.5

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Uso de la plataforma .NET de

> **Important**
>
> App Runner dejará de ser compatible con .NET 6 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La AWS App Runner plataforma.NET proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una versión.NET. Cuando usas un entorno de ejecución de.NET, App Runner comienza con una imagen de tiempo de ejecución de.NET gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de.NET y algunas herramientas y paquetes de dependencias populares. App Runner usa esta imagen de tiempo de ejecución administrado como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

Para obtener nombres y versiones de tiempo de ejecución de .NET válidos, consultethe section called “Información sobre la versión”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de versión para los tiempos de ejecución de.NET: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Por ejemplo: `6.0.9`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `6.0`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `6.0.9`— Bloquear una versión de parche específica. App Runner no actualiza tu versión en tiempo de ejecución.

Temas

- Configuración de tiempo de ejecución de.NET

- Ejemplos de tiempo de ejecución de.NET
- Información sobre la versión de.NET Runtime

Configuración de tiempo de ejecución de.NET

Al elegir un tiempo de ejecución administrado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.
- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Ejemplos de tiempo de ejecución de.NET

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio.NET. El último ejemplo es el código fuente de una aplicación.NET completa que se puede implementar en un servicio de ejecución de.NET.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es`6.0.9`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de .NET compatible, consultethe section called “Información sobre la versión”.

Archivo de configuración de .NET mínimo

En este ejemplo se muestra un archivo de configuración mínimo que se puede utilizar con un entorno de ejecución gestionado por .NET. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consultethe section called “Ejemplos de archivos de configuración”.

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
```

Archivo de configuración de.NET extendido

En este ejemplo se muestra el uso de todas las claves de configuración en un entorno de ejecución gestionado por .NET.

> **Note**
>
> La versión de tiempo de ejecución que se utiliza en estos ejemplos es`6.0.9`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de .NET compatible, consultethe section called “Información sobre la versión”.

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
  commands:
    pre-build:
      - scripts/prebuild.sh
    build:
      - dotnet publish -c Release -o out
    post-build:
```

```
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 6.0.9
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

Fuente completa de la aplicación.NET

En este ejemplo se muestra el código fuente de una aplicación.NET completa que se puede implementar en un servicio de tiempo de ejecución de.NET.

> **Note**
>
> - Ejecute el siguiente comando para crear una aplicación web sencilla de.NET 6: `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - `apprunner.yaml`Añádala a la aplicación web .NET 6 creada.

Example HelloWorldDotNetApp

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
```

```
      value: "http://*:5000"
```

Información sobre la versión de.NET Runtime

> **Important**
>
> App Runner dejará de ser compatible con .NET 6 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

En este tema se enumeran todos los detalles de las versiones en tiempo de ejecución de.NET compatibles con App Runner.

Versiones en tiempo de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
.NET 6 (dotnet6)	6.0.36	.NET SDK 6.0.428
	6.0.33	.NET SDK 6.0.425
	6.0.32	.NET SDK 6.0.424
	6.0.31	.NET SDK 6.0.423
	6.0.30	.NET SDK 6.0.422
	6.0.29	.NET SDK 6.0.421
	6.0.28	.NET SDK 6.0.420
	6.0.26	.NET SDK 6.0.418
	6.0.25	.NET SDK 6.0.417
	6.0.24	.NET SDK 6.0.416
	6.0.22	.NET SDK 6.0.414

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	6.0.21	.NET SDK 6.0.413
	6.0.20	.NET SDK 6.0.412
	6.0.19	.NET SDK 6.0.411
	6.0.16	.NET SDK 6.0.408
	6.0.15	.NET SDK 6.0.407
	6.0.14	.NET SDK 6.0.406
	6.0.13	.NET SDK 6.0.405
	6.0.12	.NET SDK 6.0.404
	6.0.11	.NET SDK 6.0.403
	6.0.10	.NET SDK 6.0.402
	6.0.9	.NET SDK 6.0.401

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Uso de la plataforma PHP de

> **Important**
>
> App Runner dejará de ser compatible con PHP 8.1 el 31 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La plataforma AWS App Runner PHP proporciona tiempos de ejecución gestionados. Puede usar cada tiempo de ejecución para crear y ejecutar contenedores con aplicaciones web basadas en una versión de PHP. Cuando utilizas un entorno de ejecución de PHP, App Runner comienza con una imagen de tiempo de ejecución de PHP gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de PHP y algunas herramientas. App Runner usa esta imagen de tiempo de ejecución administrada como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

Para obtener nombres y versiones de tiempo de ejecución de PHP válidos, consultethe section called “Información sobre la versión”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución del servicio.

Sintaxis de la versión para los tiempos de ejecución de PHP: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Por ejemplo: `8.1.10`

Los siguientes son ejemplos de bloqueo de versiones:

- `8.1`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `8.1.10`— Bloquear una versión de parche específica. App Runner no actualiza tu versión de ejecución.

> **Important**
>
> Si quieres especificar el directorio fuente del repositorio de código para tu servicio de App Runner en una ubicación distinta del directorio raíz del repositorio predeterminado, la versión de ejecución gestionada por PHP debe ser PHP `8.1.22` o posterior. Las versiones de tiempo de ejecución de PHP anteriores a solo `8.1.22` pueden usar el directorio fuente raíz predeterminado.

Temas

Configuración del tiempo de ejecución de PHP

Al elegir un tiempo de ejecución gestionado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.
- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Compatibilidad

Puedes ejecutar tus servicios de App Runner en la plataforma PHP mediante uno de los siguientes servidores web:

- Apache HTTP Server
- NGINX

Apache HTTP Servery NGINX son compatibles con PHP-FPM. Puede iniciar Apache HTTP Servery NGINXmediante una de las siguientes opciones:

- Supervisord: para obtener más información sobre la ejecución de un supervisord, consulte Ejecutar supervisord.
- Script de inicio

Para ver ejemplos sobre cómo configurar el servicio App Runner con la plataforma PHP mediante el servidor HTTP Apache o NGINX, consulte. the section called “Fuente completa de la aplicación PHP”

Estructura de archivos

`index.php`Debe estar instalado en la `public` carpeta del `root` directorio del servidor web.

> **Note**
>
> Se recomienda almacenar los `supervisord.conf` archivos `startup.sh` o en el directorio raíz del servidor web. Asegúrese de que el `start` comando apunte a la ubicación en la que están almacenados `supervisord.conf` los archivos `startup.sh` o.

A continuación se muestra un ejemplo de la estructura de archivos que está utilizando supervisord.

```
/
## public/
#  ## index.php
```

```
## apprunner.yaml
## supervisord.conf
```

El siguiente es un ejemplo de la estructura de archivos si utiliza un script de inicio.

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Se recomienda almacenar estas estructuras de archivos en el directorio fuente del repositorio de código designado para el servicio App Runner.

```
/<sourceDirectory>/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

> **Important**
>
> Si quieres especificar el directorio fuente del repositorio de código para tu servicio de App Runner en una ubicación que no sea el directorio raíz del repositorio predeterminado, tu versión de tiempo de ejecución gestionado por PHP debe ser PHP `8.1.22` o posterior. Las versiones de tiempo de ejecución de PHP anteriores a solo `8.1.22` pueden usar el directorio fuente raíz predeterminado.
>
> App Runner actualiza el tiempo de ejecución de tu servicio a la última versión en cada implementación o actualización del servicio. Su servicio utilizará los tiempos de ejecución más recientes de forma predeterminada, a menos que haya especificado el bloqueo de versiones con la `runtime-version` palabra clave del archivo de configuración de App Runner.

Ejemplos de tiempos de ejecución de PHP

Los siguientes son ejemplos de archivos de configuración de App Runner que se utilizan para crear y ejecutar un servicio PHP.

Archivo de configuración PHP mínimo

El siguiente ejemplo es un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por PHP. Para obtener más información sobre un archivo de configuración mínima, consultethe section called “Ejemplos de archivos de configuración”.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
```

Archivo de configuración PHP extendido

El siguiente ejemplo utiliza todas las claves de configuración con un tiempo de ejecución gestionado por PHP.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es*8.1.10*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de ejecución de PHP compatible, consultethe section called “Información sobre la versión”.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - echo example build command for PHP
    post-build:
      - scripts/postbuild.sh
```

```
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 8.1.10
  command: ./startup.sh
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Fuente completa de la aplicación PHP

Los siguientes ejemplos son del código fuente de una aplicación PHP que puede usar para implementar en un servicio de tiempo de ejecución de PHP usando Apache HTTP Servero NGINX. En estos ejemplos se supone que se utiliza la estructura de archivos predeterminada.

Ejecutar la plataforma PHP con Apache HTTP Server el uso supervisord

Example Estructura de archivos

> **Note**
>
> - El `supervisord.conf` archivo se puede almacenar en cualquier parte del repositorio. Asegúrese de que el `start` comando apunta al lugar donde está almacenado el `supervisord.conf` archivo.
> - `index.php`Debe estar instalado en la `public` carpeta situada debajo del `root` directorio.

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example supervisord.conf

```
[supervisord]
```

```
nodaemon=true

[program:httpd]
command=httpd -DFOREGROUND
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
```

```
?>
</body>
</html>
```

Ejecutar la plataforma PHP con Apache HTTP Server el uso startup script

Example Estructura de archivos

> **Note**
>
> - El `startup.sh` archivo se puede almacenar en cualquier parte del repositorio. Asegúrese de que el `start` comando apunta al lugar donde está almacenado el `startup.sh` archivo.
> - `index.php`Debe estar instalado en la `public` carpeta situada debajo del `root` directorio.

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start apache
httpd -DFOREGROUND &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Asegúrate de guardar el `startup.sh` archivo como ejecutable antes de enviarlo a un repositorio de Git. Se usa `chmod +x startup.sh` para establecer el permiso de ejecución en `startup.sh` el archivo.
> - Si no guarda el `startup.sh` archivo como ejecutable, `chmod +x startup.sh` introdúzcalo como `build` comando en el `apprunner.yaml` archivo.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Ejecutar la plataforma PHP con NGINX el uso supervisord

Example Estructura de archivos

> **Note**
>
> - El `supervisord.conf` archivo se puede almacenar en cualquier parte del repositorio. Asegúrese de que el `start` comando apunta al lugar donde está almacenado el `supervisord.conf` archivo.
> - `index.php`Debe estar instalado en la `public` carpeta situada debajo del `root` directorio.

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example supervisord.conf

```
[supervisord]
nodaemon=true

[program:nginx]
command=nginx -g "daemon off;"
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Ejecutar la plataforma PHP con NGINX el uso startup script

Example Estructura de archivos

> **Note**
>
> - El `startup.sh` archivo se puede almacenar en cualquier parte del repositorio. Asegúrese de que el `start` comando apunta al lugar donde está almacenado el `startup.sh` archivo.
> - `index.php`Debe estar instalado en la `public` carpeta situada debajo del `root` directorio.

```
/
```

```
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start nginx
nginx -g 'daemon off;' &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Asegúrate de guardar el `startup.sh` archivo como ejecutable antes de enviarlo a un repositorio de Git. Se usa `chmod +x startup.sh` para establecer el permiso de ejecución en `startup.sh` el archivo.
> - Si no guarda el `startup.sh` archivo como ejecutable, `chmod +x startup.sh` introdúzcalo como `build` comando en el `apprunner.yaml` archivo.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
```

```
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Información sobre la versión de PHP Runtime

> **Important**
>
> App Runner dejará de ser compatible con PHP 8.1 el 31 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

En este tema se enumeran todos los detalles de las versiones de PHP en tiempo de ejecución compatibles con App Runner.

Versiones de tiempo de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
PHP 8.1 (php81)	8.1.33	
	8.1,32	
	8.1,31	
	8.1,29	

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	8.1,28	
	8.1,27	
	8.1,26	
	8.1,24	
	8.1.22	
	8.1.21	
	8.1,20	
	8.1,19	
	8.1,17	
	8.1,16	
	8.1,14	
	8.1,13	
	8.1.12	
	8.1,10	

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Uso de la plataforma Ruby de

> **Important**
>
> App Runner dejará de ser compatible con Ruby 3.1 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La plataforma AWS App Runner Ruby proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una versión de Ruby. Cuando usas un entorno de ejecución de Ruby, App Runner comienza con una imagen de tiempo de ejecución de Ruby gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de Ruby y algunas herramientas. App Runner usa esta imagen de tiempo de ejecución administrada como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es*`<language-name><major-version>`*.

Para ver los nombres y las versiones de los tiempos de ejecución de Ruby válidos, consultethe section called “Información de lanzamiento”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de la versión para los tiempos de ejecución de Ruby: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Por ejemplo: `3.1.2`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `3.1`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `3.1.2`— Bloquear una versión de parche específica. App Runner no actualiza tu versión en tiempo de ejecución.

Temas

Configuración del tiempo de ejecución de Ru

Al elegir un tiempo de ejecución administrado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.
- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Ejemplos de ejecución de Ruby

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio de Ruby.

Archivo de configuración mínimo de Ruby

En este ejemplo, se muestra un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por Ruby. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consultethe section called “Ejemplos de archivos de configuración”.

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 8080
```

Archivo de configuración de Ruby extendido

Este ejemplo muestra el uso de todas las claves de configuración con un entorno de ejecución gestionado por Ruby.

> **Note**
>
> La versión de tiempo de ejecución que se utiliza en estos ejemplos es `3.1.2`. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Ruby compatible, consultethe section called “Información de lanzamiento”.

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - bundle install
    post-build:
      - scripts/postbuild.sh
```

```
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.1.2
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Fuente completa de la aplicación Ruby

Estos ejemplos muestran el código fuente de una aplicación de Ruby completa que se puede implementar en un servicio de tiempo de ejecución de Ruby.

Example servidor.rb

```
# server.rb
require 'sinatra'

get '/' do
  'Hello World!'
end
```

Example config.ru

```
# config.ru

require './server'

run Sinatra::Application
```

Example Archivo Gemfile

```
# Gemfile
source 'https://rubygems.org (https://rubygems.org/)'

gem 'sinatra'
gem 'puma'
```

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
```

Información sobre la versión de Ruby Runtime

> **Important**
>
> App Runner dejará de ser compatible con Ruby 3.1 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

En este tema se enumeran todos los detalles de las versiones de Ruby en tiempo de ejecución compatibles con App Runner.

Versiones de tiempo de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Ruby 3.1 (ruby31)	3.1.7	SQLite 3.50.2
	3.1.7	SQLite 3.50,1
	3.1.7	SQLite 3.50,0
	3.1.6	SQLite 3.49,1
	3.1.4	SQLite 3.46,0

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
	3.1.3	SQLite 3.41.0
	3.1.2	SQLite 3.39,4

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Uso de la plataforma Go de

> **Important**
>
> App Runner dejará de ser compatible con Go 1.18 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

La plataforma AWS App Runner Go proporciona tiempos de ejecución gestionados. Cada tiempo de ejecución facilita la creación y ejecución de contenedores con aplicaciones web basadas en una versión de Go. Cuando utilizas un motor de ejecución de Go, App Runner comienza con una imagen de tiempo de ejecución de Go gestionada. Esta imagen se basa en la imagen de Docker de Amazon Linux y contiene el paquete de tiempo de ejecución de una versión de Go y algunas herramientas. App Runner usa esta imagen de tiempo de ejecución administrada como imagen base y agrega el código de la aplicación para crear una imagen de Docker. A continuación, implementa esta imagen para ejecutar el servicio web en un contenedor.

Usted especifica un tiempo de ejecución para su servicio de App Runner al crear un servicio mediante la consola de App Runner o la operación de la CreateServiceAPI. También puedes especificar un tiempo de ejecución como parte de tu código fuente. Usa la `runtime` palabra clave en

un archivo de configuración de App Runner que incluyas en tu repositorio de código. La convención de nomenclatura de un tiempo de ejecución gestionado es`<language-name><major-version>`.

Para ver los nombres y las versiones de los tiempos de ejecución de Go válidos, consultethe section called “Información sobre el lanzamiento”.

App Runner actualiza el tiempo de ejecución del servicio a la versión más reciente en cada implementación o actualización del servicio. Si su aplicación requiere una versión específica de un tiempo de ejecución administrado, puede especificarla mediante la `runtime-version` palabra clave del archivo de configuración de App Runner. Puedes bloquearla en cualquier nivel de versión, incluida una versión principal o secundaria. App Runner solo realiza actualizaciones de nivel inferior en el tiempo de ejecución de tu servicio.

Sintaxis de versión para los tiempos de ejecución de Go: `major[.minor[.patch]]`

Por ejemplo: `1.18.7`

Los siguientes ejemplos muestran el bloqueo de versiones:

- `1.18`— Bloquee las versiones principales y secundarias. App Runner actualiza solo las versiones con parches.
- `1.18.7`— Bloquear una versión de parche específica. App Runner no actualiza tu versión de ejecución.

Temas

- Configuración de tiempo de ejecución de Go
- Ejemplos de tiempo de ejecución de Go
- Información sobre la versión de Go Runtime

Configuración de tiempo de ejecución de Go

Al elegir un tiempo de ejecución gestionado, también debe configurar, como mínimo, los comandos de compilación y ejecución. Los configuras al crear o actualizar tu servicio App Runner. Puede hacerlo mediante uno de los siguientes métodos:

- Mediante la consola de App Runner: especifique los comandos en la sección Configurar compilación del proceso de creación o en la pestaña de configuración.

- Uso de la API de App Runner: llame a la operación CreateServiceo UpdateServiceAPI. Especifique los comandos mediante los `StartCommand` miembros `BuildCommand` y del tipo de CodeConfigurationValuesdatos.
- Uso de un archivo de configuración: especifique uno o más comandos de compilación en un máximo de tres fases de compilación y un solo comando de ejecución que sirva para iniciar la aplicación. Hay opciones de configuración opcionales adicionales.

Proporcionar un archivo de configuración es opcional. Al crear un servicio de App Runner mediante la consola o la API, debe especificar si App Runner obtiene los ajustes de configuración directamente al crearlos o de un archivo de configuración.

Ejemplos de tiempo de ejecución de Go

Los siguientes ejemplos muestran los archivos de configuración de App Runner para crear y ejecutar un servicio Go.

Archivo de configuración mínimo de Go

En este ejemplo, se muestra un archivo de configuración mínimo que puede utilizar con un entorno de ejecución gestionado por Go. Para ver las suposiciones que App Runner hace con un archivo de configuración mínimo, consultethe section called “Ejemplos de archivos de configuración”.

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
```

Archivo de configuración Go extendido

En este ejemplo, se muestra el uso de todas las claves de configuración con un entorno de ejecución gestionado por Go.

> **Note**
>
> La versión en tiempo de ejecución que se utiliza en estos ejemplos es*1.18.7*. Puede sustituirla por la versión que desee utilizar. Para ver la última versión de tiempo de ejecución de Go compatible, consultethe section called “Información sobre el lanzamiento”.

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - go build main.go
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 1.18.7
  command: ./main
  network:
    port: 3000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Fuente completa de la aplicación Go

En estos ejemplos se muestra el código fuente de una aplicación Go completa que se puede implementar en un servicio de tiempo de ejecución de Go.

Example main.go

```
package main
import (
```

```
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprint(w, "<h1>Welcome to App Runner</h1>")
    })
    fmt.Println("Starting the server on :3000...")
    http.ListenAndServe(":3000", nil)
}
```

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
  network:
    port: 3000
    env: APP_PORT
```

Información sobre la versión de Go Runtime

> **Important**
>
> App Runner dejará de ser compatible con Go 1.18 el 1 de diciembre de 2025. Para obtener recomendaciones y más información, consultethe section called “Fin del soporte para las versiones de tiempo de ejecución gestionado”.

En este tema se enumeran todos los detalles de las versiones de tiempo de ejecución de Go compatibles con App Runner.

Versiones de tiempo de ejecución compatibles: versión original de App Runner

Nombre del entorno de tiempo de ejecución	Versiones secundarias	Paquetes incluidos
Go 1 (go1)	1.18.10	
	1.18,9	
	1.18,8	
	1.18,7	

> **Note**
>
> App Runner proporciona un proceso de compilación revisado para los principales tiempos de ejecución específicos que se publicaron más recientemente. Por este motivo, verás referencias a la versión revisada de App Runner y a la versión original de App Runner en determinadas secciones de este documento. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

Desarrollo de código de aplicación para App Runner

En este capítulo se analiza la información sobre el tiempo de ejecución y las pautas de desarrollo que debe tener en cuenta al desarrollar o migrar el código de la aplicación para su despliegue. AWS App Runner

Información de tiempo de ejecución

Ya sea que proporciones una imagen de contenedor o que App Runner cree una por ti, App Runner ejecuta el código de tu aplicación en una instancia de contenedor. Estos son algunos aspectos clave del entorno de ejecución de la instancia de contenedor.

- Compatibilidad con el marco: App Runner es compatible con cualquier imagen que implemente una aplicación web. Es independiente del lenguaje de programación que elijas y del servidor o marco de aplicaciones web que utilices, si lo utilizas. Para su comodidad, ofrecemos tiempos de ejecución gestionados específicos de cada plataforma para diversas plataformas de programación, a fin de agilizar el proceso de creación de aplicaciones y la creación de imágenes abstractas.
- Solicitudes web: App Runner proporciona soporte para HTTP 1.0 y HTTP 1.1 a las instancias contenedoras. Para obtener más información sobre la configuración del servicio, consultethe section called “Configuración”. No es necesario implementar la gestión del tráfico seguro HTTPS. App Runner redirige todas las solicitudes HTTP entrantes a los puntos finales HTTPS correspondientes. No es necesario configurar ningún ajuste para permitir la redirección de las solicitudes web HTTP. App Runner finaliza el TLS antes de pasar las solicitudes a la instancia del contenedor de la aplicación.

> **Note**
>
> - Hay un límite de tiempo de espera total de 120 segundos para las solicitudes HTTP. Los 120 segundos incluyen el tiempo que tarda la aplicación en leer la solicitud, incluido el cuerpo, y en terminar de escribir la respuesta HTTP.
> - El límite de tiempo de espera de lectura y respuesta de la solicitud depende de las aplicaciones que utilices. Estas aplicaciones pueden tener sus propios tiempos de espera internos, por ejemplo, el servidor HTTP para Python, Gunicorn, tiene un límite de tiempo de espera predeterminado de 30 segundos. En esos casos, el límite de tiempo de espera de la aplicación anula el límite de 120 segundos de App Runner.

- No necesitas configurar los conjuntos de cifrado TLS ni ningún otro parámetro, ya que App Runner, al ser un servicio totalmente gestionado, gestiona la terminación de TLS por ti.

- Aplicaciones sin estado: actualmente, App Runner no admite aplicaciones con estado. Por lo tanto, App Runner no garantiza la persistencia del estado más allá de la duración del procesamiento de una sola solicitud web entrante.
- Almacenamiento: App Runner amplía o reduce automáticamente las instancias de tu aplicación App Runner en función del volumen de tráfico entrante. Puede configurar las opciones de escalado automático para su aplicación App Runner. Dado que el número de instancias actualmente activas que procesan las solicitudes web se basa en el volumen de tráfico entrante, App Runner no puede garantizar que los archivos se conserven más allá del procesamiento de una sola solicitud. Por lo tanto, App Runner implementa el sistema de archivos en la instancia contenedora como almacenamiento efímero, lo que implica que los archivos son transitorios. Por ejemplo, los archivos no se conservan cuando pausas y reanudas el servicio de App Runner.

  App Runner te proporciona 3 GB de almacenamiento efímero y utiliza una parte de los 3 GB de almacenamiento efímero para la imagen de contenedor extraída, comprimida y descomprimida de la instancia. El servicio App Runner puede usar el almacenamiento efímero restante. Sin embargo, no se trata de un almacenamiento permanente debido a su naturaleza apátrida.

  > **Note**
  >
  > Puede haber situaciones en las que los archivos de almacenamiento persistan en todas las solicitudes. Por ejemplo, si la siguiente solicitud llega a la misma instancia, los archivos de almacenamiento se conservarán. La persistencia de los archivos de almacenamiento en todas las solicitudes puede resultar útil en determinadas situaciones. Por ejemplo, al gestionar una solicitud, puede almacenar en caché los archivos que la aplicación descarga si es posible que las solicitudes futuras los necesiten. Esto podría acelerar la gestión de solicitudes en el futuro, pero no puede garantizar que la velocidad aumente. El código no debe dar por sentado que aún existe un archivo que se descargó en una solicitud anterior. Para garantizar el almacenamiento en caché mediante un almacén de datos en memoria de alto rendimiento y baja latencia, utilice un servicio como Amazon. ElastiCache
- Variables de entorno: de forma predeterminada, App Runner hace que la variable de `PORT` entorno esté disponible en tu instancia de contenedor. Puede configurar el valor de la variable con la información del puerto y agregar variables y valores de entorno personalizados. También puede

hacer referencia a los datos confidenciales almacenados en el AWS Secrets Manageralmacén de AWS Systems Manager parámetros como variables de entorno. Para obtener más información sobre la creación de variables de entorno, consulteVariables de entorno de referencia.

- Función de instancia: si el código de la aplicación realiza llamadas a algún AWS servicio, mediante el servicio APIs o uno de ellos AWS SDKs, cree una función de instancia mediante AWS Identity and Access Management (IAM). A continuación, adjúntelo a su servicio de App Runner cuando lo cree. Incluye todos los permisos de acción de AWS servicio que tu código requiere en tu rol de instancia. Para obtener más información, consulte the section called “Rol de instancia”.

Directrices de desarrollo de código

Tenga en cuenta estas pautas al desarrollar código para una aplicación web de App Runner.

- Aplicar parches a las imágenes de los contenedores: al proporcionar imágenes de contenedores, usted es responsable de actualizarlas y aplicarles parches con regularidad. Mientras App Runner administra la infraestructura, debes garantizar la seguridad y el up-to-date estado de las imágenes de contenedor proporcionadas. Para obtener más información, consulte la documentación de AWS App Runner
- Diseñe código sin estado: diseñe la aplicación web que implemente en su servicio App Runner para que no tenga estado. El código debe suponer que ningún estado persiste más allá de la duración del procesamiento de una sola solicitud web entrante.
- Elimine archivos temporales: cuando crea archivos, se almacenan en un sistema de archivos y ocupan parte de la asignación de almacenamiento de su servicio. Para evitar out-of-storage errores, no guardes los archivos temporales durante períodos prolongados. Equilibre el tamaño del almacenamiento con la velocidad de gestión de las solicitudes al tomar decisiones sobre el almacenamiento en caché de los archivos.
- Inicio de instancias: App Runner proporciona cinco minutos de tiempo de inicio de la instancia. La instancia debe escuchar las solicitudes en sus puertos de escucha configurados y estar en buen estado a los cinco minutos de su inicio. Durante el tiempo de inicio, a las instancias de App Runner se les asigna una CPU virtual (vCPU) en función de la configuración de la vCPU. Para obtener más información sobre la configuración de vCPU disponible, consulte. the section called “Configuraciones compatibles con App Runner”

  Una vez que la instancia se inicia correctamente, pasa a un estado inactivo y espera las solicitudes. El pago se basa en la duración del inicio de la instancia, con un cargo mínimo de un

minuto por inicio de la instancia. Para obtener información sobre precios, consulte Precios de AWS App Runner.

Uso de la consola de App Runner

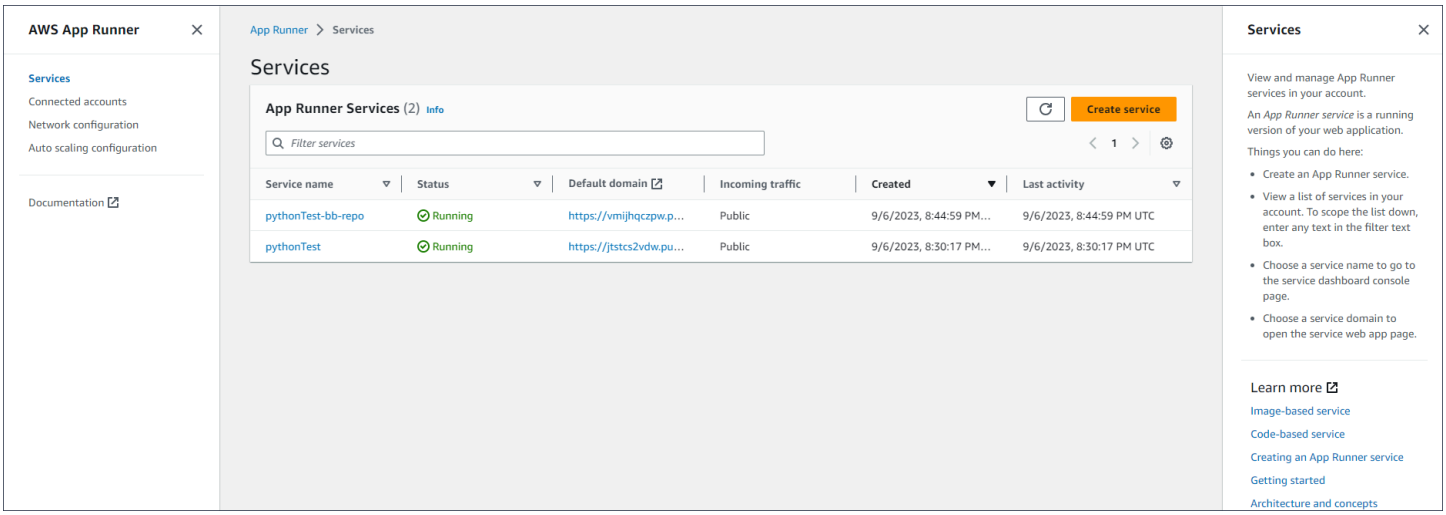
Usa la AWS App Runner consola para crear, administrar y monitorear tus servicios de App Runner y los recursos relacionados, como las cuentas conectadas. Puede ver los servicios existentes, crear otros nuevos y configurar un servicio. Puede ver el estado de un servicio de App Runner, así como ver los registros, supervisar la actividad y realizar un seguimiento de las métricas. También puedes navegar al sitio web de tu servicio o al repositorio de origen.

En las siguientes secciones se describe el diseño y la funcionalidad de la consola y se proporciona información relacionada.

Diseño general de la consola

La consola de App Runner tiene tres áreas. De izquierda a derecha:

- Panel de navegación: panel lateral que se puede contraer o expandir. Úselo para elegir la página de consola de nivel superior que desee usar.
- Panel de contenido: la parte principal de la página de la consola. Úselo para ver información y realizar sus tareas.
- Panel de ayuda: un panel lateral para obtener más información. Amplíelo para obtener ayuda sobre la página en la que se encuentra. O elige cualquier enlace de información de una página de consola para obtener ayuda contextual.



La página de servicios

La página de servicios muestra los servicios de App Runner de su cuenta. Puede reducir el alcance de la lista mediante el cuadro de texto del filtro.

Para llegar al Services page

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, elija Servicios.

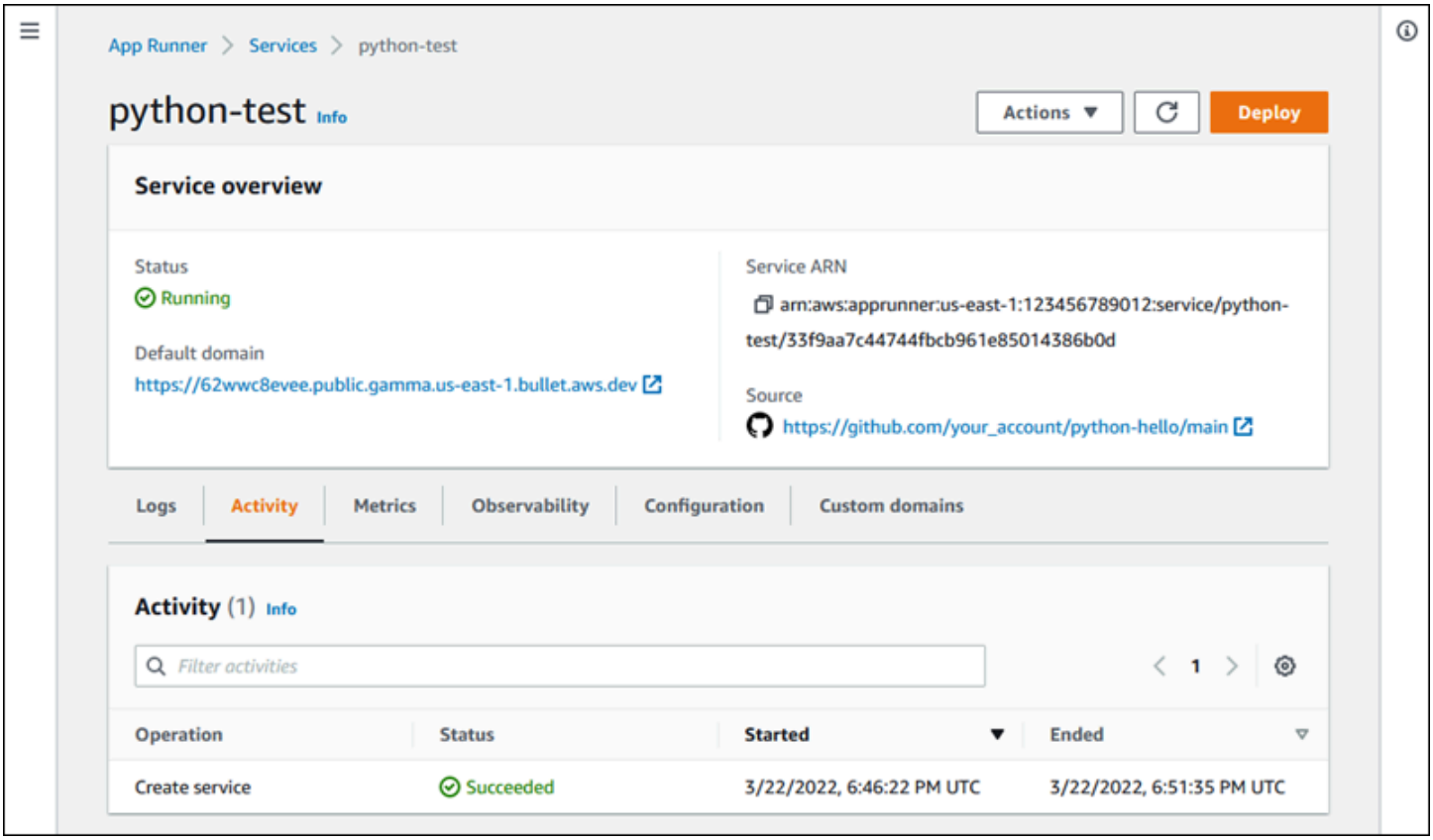
Cosas que puedes hacer aquí:

- Crea un servicio de App Runner. Para obtener más información, consulte the section called “Creación”.
- Elija un nombre de servicio para ir a la página de la consola del panel de servicios.
- Elija un dominio de servicio para abrir la página de la aplicación web del servicio.

La página del panel de control del servicio

Puede ver la información sobre un servicio de App Runner y administrarlo desde la página del panel de control del servicio. En la parte superior de la página, puedes ver el nombre del servicio.

Para acceder al panel de servicios, dirígete a la página de servicios (consulta la sección anterior) y, a continuación, selecciona tu servicio de App Runner.



La sección de descripción general del servicio proporciona detalles básicos sobre el servicio App Runner y su aplicación. Cosas que puede hacer aquí:

- Vea los detalles del servicio, como el estado, el estado y el ARN.
- Navegue hasta el dominio predeterminado, el dominio que App Runner proporciona para la aplicación web que se ejecuta en su servicio. Se trata de un subdominio del `awsapprunner.com` dominio propiedad de App Runner.
- Navegue hasta el repositorio de origen implementado en el servicio.
- Inicie una implementación del repositorio de origen en su servicio.
- Pausa, reanuda y elimina tu servicio.

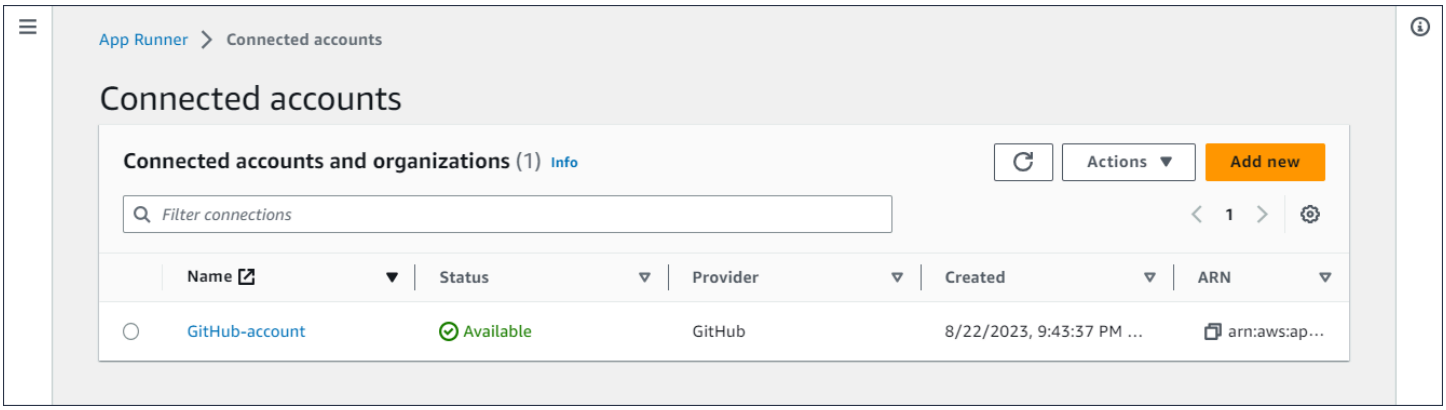
Las pestañas que se encuentran debajo de la descripción general del servicio son para la gestión y la observabilidad del servicio.

La página de cuentas conectadas

La página de cuentas conectadas muestra las conexiones de App Runner con los proveedores de repositorios de código fuente de su cuenta. Puede reducir el alcance de la lista mediante el cuadro de texto del filtro. Para obtener más información sobre las cuentas conectadas, consultethe section called “Conexiones”.

Para ir al Cuentas conectadas page

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Cuentas conectadas.



Cosas que puedes hacer aquí:

- Consulta una lista de conexiones de proveedores de repositorios en tu cuenta. Para reducir el alcance de la lista, introduce cualquier texto en el cuadro de texto del filtro.
- Elija un nombre de conexión para ir a la cuenta u organización del proveedor correspondiente.
- Seleccione una conexión para completar el apretón de manos de una conexión que acaba de establecer (como parte de la creación de un servicio) o para eliminarla.

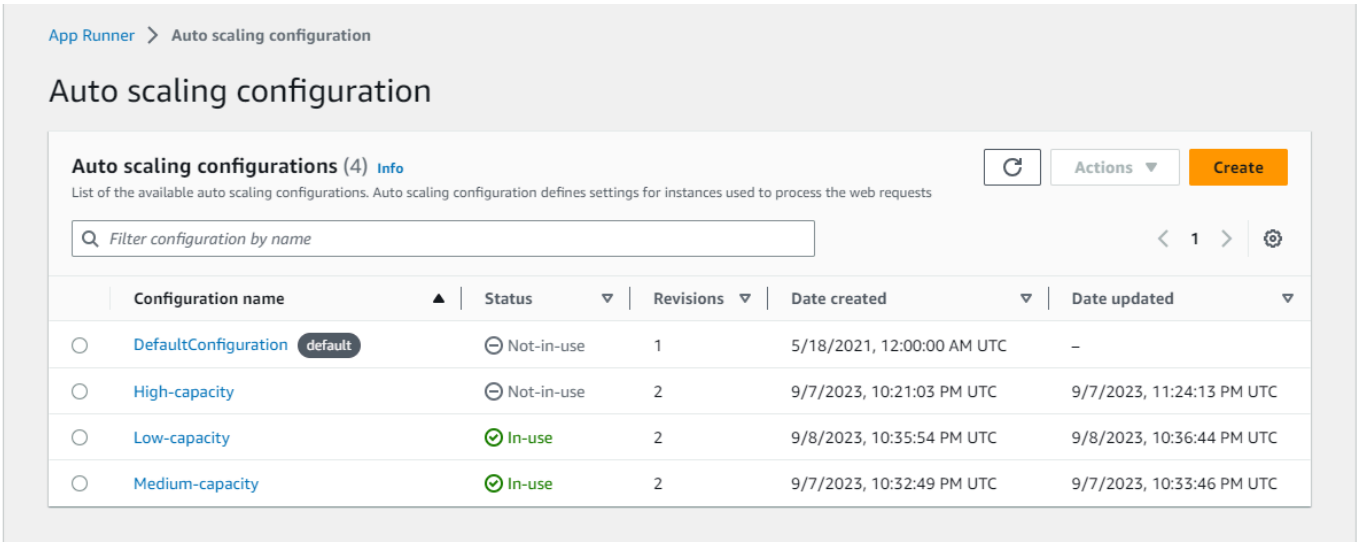
La página de configuraciones de escalado automático

La página de configuraciones de autoescalado muestra las configuraciones de autoescalado que ha configurado en su cuenta. Puede configurar algunos parámetros para ajustar el comportamiento del autoescalado y guardarlos en diferentes configuraciones que luego podrá asignar a uno o más servicios de App Runner. Puede reducir el alcance de la lista mediante el cuadro de texto del filtro.

Para obtener más información sobre las configuraciones de autoescalado, consulteAdministre el escalado automático de un servicio.

Para llegar a la Configuración de escalado automático page

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Configuración de escalado automático.



Cosas que puede hacer aquí:

- Consulta la lista de configuraciones de escalado automático existentes en tu cuenta.
- Cree una nueva configuración de autoescalado o una revisión de una existente.
- Establezca una configuración de escalado automático como predeterminada para los nuevos servicios que cree.
- Elimine una configuración.
- Seleccione el nombre de una configuración para acceder al panel de revisiones de escalado automático para gestionar las revisiones.

Administrar el servicio App Runner

En este capítulo se describe cómo gestionar el AWS App Runner servicio. En este capítulo, aprenderá a gestionar el ciclo de vida de su servicio: crear, configurar y eliminar un servicio, implementar nuevas versiones de aplicaciones en su servicio y controlar la disponibilidad de su servicio web pausando y reanudando el servicio. También aprenderá a administrar otros aspectos de su servicio, como las conexiones y el escalado automático.

Temas

Creación de un servicio App Runner

AWS App Runner automatiza la transición de una imagen de contenedor o un repositorio de código fuente a un servicio web en ejecución que se escala automáticamente. Apuntas App Runner a tu imagen o código fuente y especificas solo una pequeña cantidad de configuraciones obligatorias. App Runner crea la aplicación si es necesario, aprovisiona recursos informáticos e implementa la aplicación para que se ejecute en ellos.

Al crear un servicio, App Runner crea un recurso de servicio. En algunos casos, es posible que tengas que proporcionar un recurso de conexión. Si utilizas la consola de App Runner, la consola crea implícitamente el recurso de conexión. Para obtener más información sobre los tipos de recursos de App Runner, consultethe section called “Recursos de App Runner”. Estos tipos de recursos tienen cuotas asociadas a tu cuenta en cada uno de ellos Región de AWS. Para obtener más información, consulte the section called “Cuotas de recursos de App Runner”.

Existen pequeñas diferencias en el procedimiento de creación de un servicio según el tipo de fuente y el proveedor. En este tema se describen diferentes procedimientos para crear estos tipos de fuentes, de modo que pueda seguir el que mejor se adapte a su situación. Para iniciar un procedimiento básico con un ejemplo de código, consulteIntroducción.

Requisitos previos

Antes de crear el servicio App Runner, asegúrate de completar las siguientes acciones:

- Complete los pasos de configuración que se indican enConfiguración.
- Asegúrese de que la fuente de la aplicación esté lista. Puedes usar un repositorio de código en GitHubBitbucket o una imagen de contenedor en Amazon Elastic Container Registry (Amazon ECR) para crear un servicio de App Runner.

Crear un servicio

En esta sección, se explica el proceso de creación de los dos tipos de servicios de App Runner: los basados en el código fuente y los basados en una imagen de contenedor.

> **Note**
>
> Si crea un conector de VPC de tráfico saliente para un servicio, el proceso de inicio del servicio que sigue experimentará una latencia única. Puedes establecer esta configuración para un servicio nuevo al crearlo o, posteriormente, con una actualización del servicio. Para obtener más información, consulte Latencia única el capítulo Redes con App Runner de esta guía.

Cree un servicio a partir de un repositorio de código

En las siguientes secciones, se muestra cómo crear un servicio de App Runner cuando tu fuente es un repositorio de código en GitHubBitbucket. Cuando utilizas un repositorio de código, App Runner debe conectarse a la organización o cuenta del proveedor. Por lo tanto, necesitas ayudar a establecer esta conexión. Para obtener más información sobre las conexiones de App Runner, consultethe section called “Conexiones”.

Al crear el servicio, App Runner crea una imagen de Docker que contiene el código y las dependencias de la aplicación. A continuación, lanza un servicio que ejecuta una instancia contenedora de esta imagen.

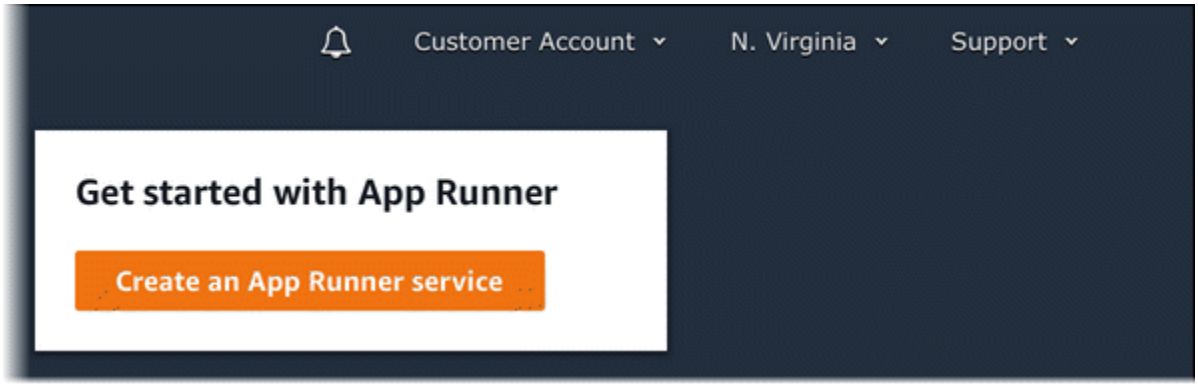
Crear un servicio a partir del código mediante la consola de App Runner

Para crear un servicio de App Runner mediante la consola

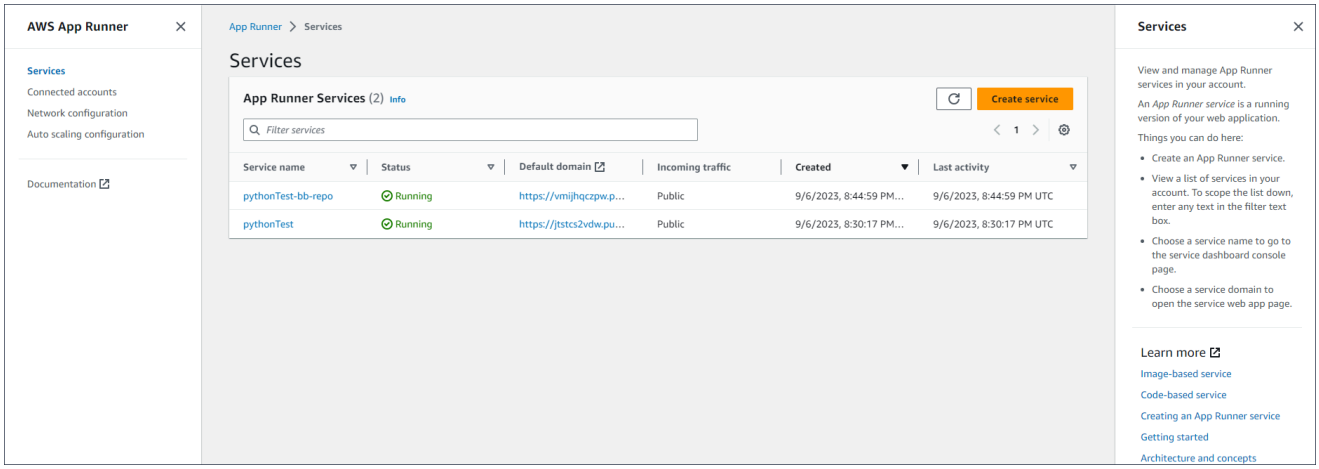
1. Configure el código fuente.

   a. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.

   b. Si aún Cuenta de AWS no tiene ningún servicio de App Runner, aparecerá la página de inicio de la consola. Selecciona Crear un servicio de App Runner.



   Si Cuenta de AWS tiene servicios existentes, se muestra la página de servicios con una lista de sus servicios. Elija Crear servicio.



   c. En la página Origen e implementación, en la sección Fuente, para Tipo de repositorio, elija Repositorio de código fuente.

   d. Seleccione un tipo de proveedor. Elige entre Bitbucket GitHubo Bitbucket.

   e. A continuación, selecciona una cuenta u organización para el proveedor que hayas utilizado anteriormente o selecciona Añadir nueva. A continuación, sigue el proceso de proporcionar

las credenciales del repositorio de código y elegir una cuenta u organización a la que conectarte.

f. En Repositorio, selecciona el repositorio que contiene el código de tu aplicación.

g. En Branch, selecciona la rama que deseas implementar.

h. En el directorio de origen, introduzca el directorio del repositorio de origen que almacena el código de la aplicación y los archivos de configuración.

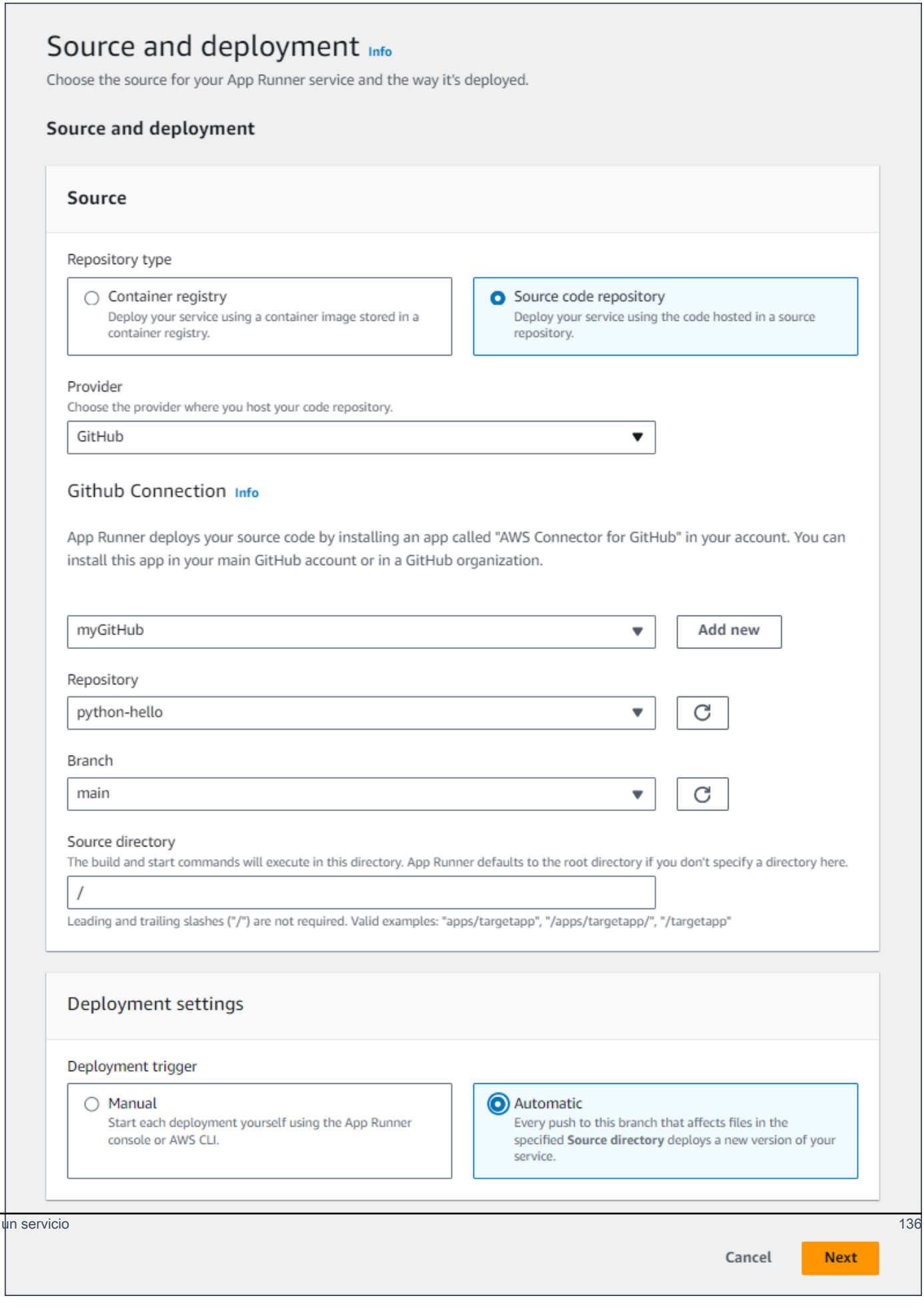
> **Note**
>
> Los comandos build e start se ejecutan desde el directorio de origen que especifique. App Runner trata la ruta como absoluta desde la raíz. Si no especificas ningún valor aquí, el directorio toma como valor predeterminado la raíz del repositorio.

2. Configure sus despliegues.

a. En la sección Configuración de despliegue, elija Manual o Automático.

Para obtener más información sobre los métodos de despliegue, consultethe section called “Métodos de implementación”.

b. Elija Siguiente.


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source and deployment
Source
Repository type
Container registry
Deploy your service using a container image stored in a container registry.
Source code repository
Deploy your service using the code hosted in a source repository.
Provider
Choose the provider where you host your code repository.
GitHub
Github Connection Info
App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.
myGitHub
Add new
Repository
python-hello
Branch
main
Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.
/
Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
Every push to this branch that affects files in the specified Source directory deploys a new version of your service.
Cancel
Next

3. Configure la compilación de la aplicación.

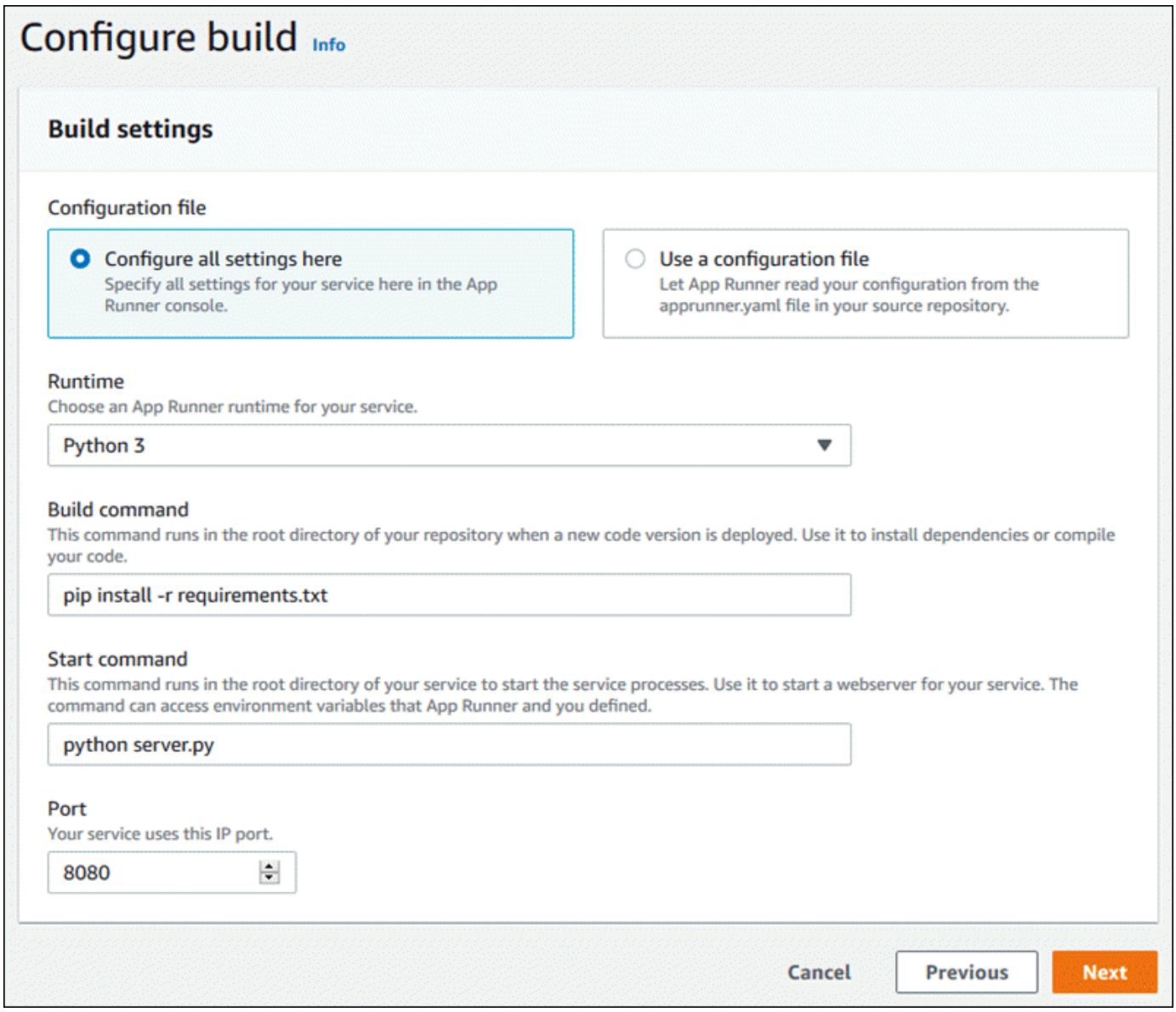
   a. En la página Configurar la compilación, en Archivo de configuración, selecciona Configurar todos los ajustes aquí si tu repositorio no contiene un archivo de configuración de App Runner o Usar un archivo de configuración si lo tiene.

   > **Note**
   >
   > Un archivo de configuración de App Runner es una forma de mantener la configuración de compilación como parte del código fuente de la aplicación. Cuando proporcionas uno, App Runner lee algunos valores del archivo y no te permite configurarlos en la consola.

   b. Proporcione la siguiente configuración de compilación:

   - Tiempo de ejecución: elija un tiempo de ejecución gestionado específico para su aplicación.
   - Comando de compilación: introduzca un comando que cree la aplicación a partir de su código fuente. Puede ser una herramienta específica del idioma o un script incluido con el código.
   - Comando de inicio: introduzca el comando que inicia el servicio web.
   - Puerto: introduzca el puerto IP que escucha su servicio web.

   c. Elija Siguiente.



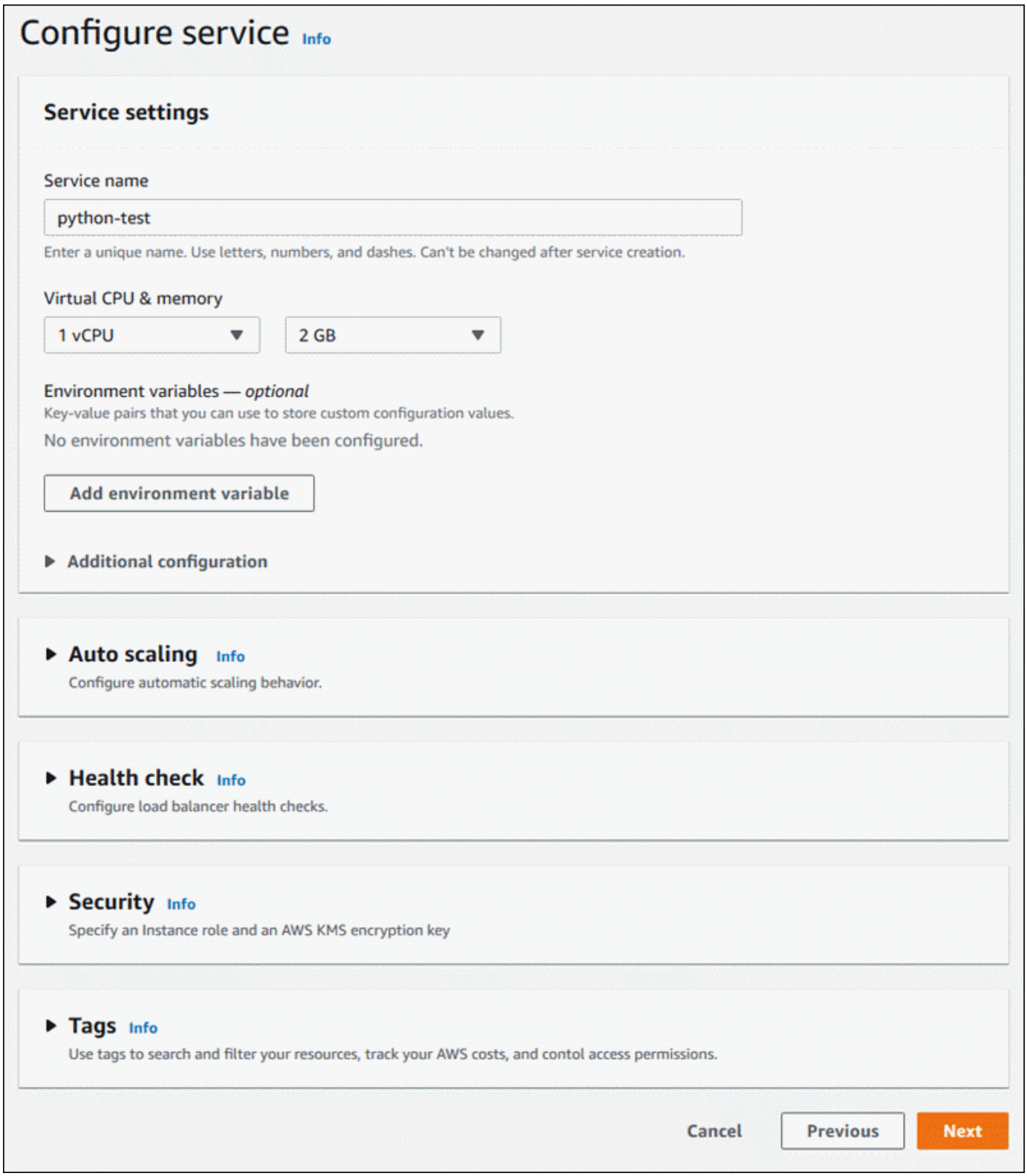
4. Configure su servicio.

   a. En la página Configurar el servicio, en la sección Configuración del servicio, introduzca un nombre de servicio.

   > **Note**
   >
   > Todos los demás ajustes del servicio son opcionales o tienen los valores predeterminados proporcionados por la consola.

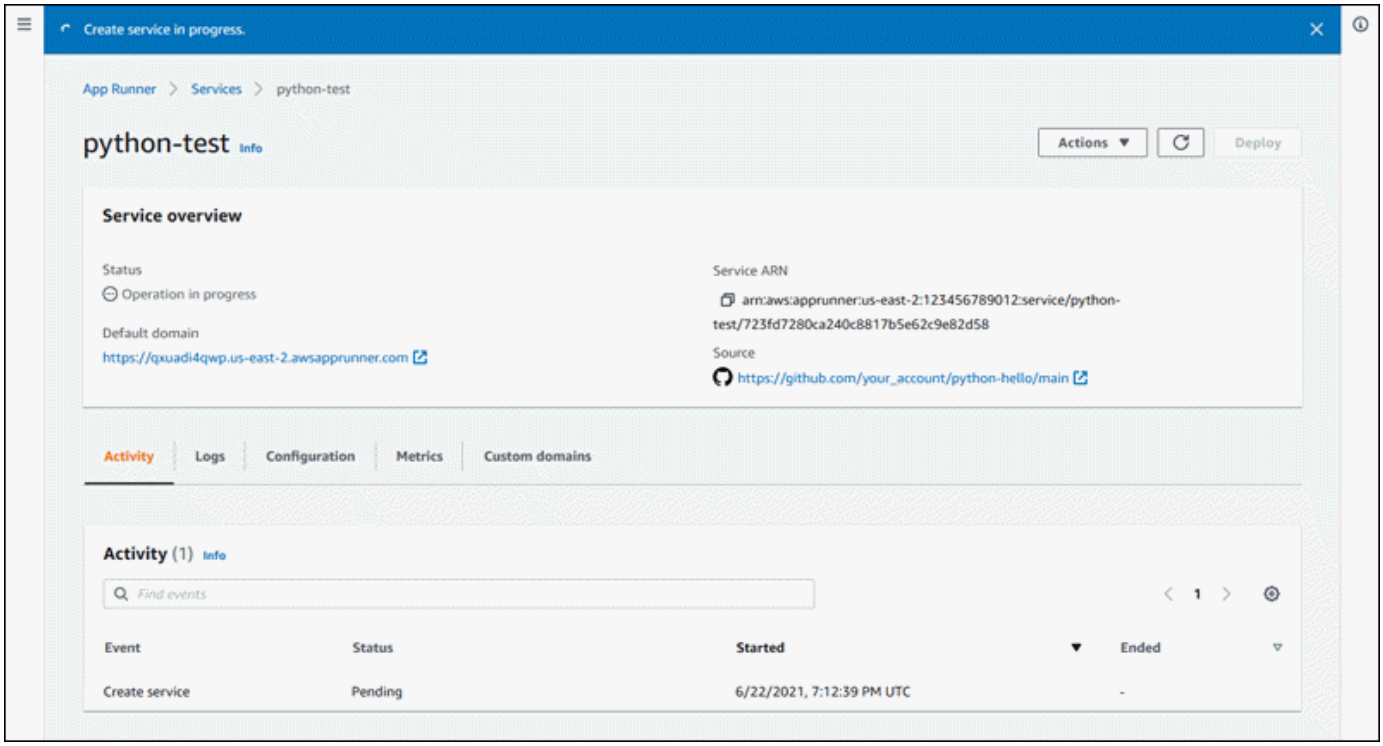
   b. Si lo desea, cambie o añada otros ajustes para cumplir con los requisitos de su aplicación.

c. Elija Siguiente.



5. En la página Revisar y crear, compruebe todos los detalles que ha introducido y, a continuación, seleccione Crear e implementar.

   Resultado: si el servicio se ha creado correctamente, la consola muestra el panel de servicio con una descripción general del nuevo servicio.



6. Verifica que el servicio esté en ejecución.

   a. En la página del panel de control del servicio, espere hasta que el estado del servicio sea En ejecución.

   b. Elija el valor de dominio predeterminado. Es la URL del sitio web de tu servicio.

   c. Usa tu sitio web y comprueba que funciona correctamente.

Crear un servicio a partir del código mediante la API de App Runner o AWS CLI

Para crear un servicio mediante la API de App Runner AWS CLI, llame a la acción de la `CreateService` API. Para obtener más información y un ejemplo, consulte CreateService. Si es la primera vez que creas un servicio con una organización o cuenta específica para un repositorio de código fuente (GitHub o Bitbucket), comienza por llamar CreateConnection. Esto establece una conexión entre App Runner y la organización o cuenta del proveedor del repositorio. Para obtener más información sobre las conexiones de App Runner, consultethe section called “Conexiones”.

Si la llamada devuelve una respuesta correcta y muestra un objeto de servicio`"Status": "CREATING"`, el servicio comienza a crearse.

Para ver un ejemplo de llamada, consulta Cómo crear un servicio de repositorio de código fuente en la referencia de la AWS App Runner API

Crear un servicio a partir de una imagen de Amazon ECR

En las siguientes secciones se muestra cómo crear un servicio de App Runner cuando la fuente es una imagen de contenedor almacenada en Amazon ECR. Amazon ECR es un Servicio de AWS. Por lo tanto, para crear un servicio basado en una imagen de Amazon ECR, debes proporcionar a App Runner un rol de acceso que contenga los permisos de acción de Amazon ECR necesarios.

> **Note**
>
> Las imágenes almacenadas en Amazon ECR Public están disponibles públicamente. Por lo tanto, si su imagen está almacenada en Amazon ECR Public, no se requiere un rol de acceso.

Cuando se crea el servicio, App Runner lanza un servicio que ejecuta una instancia contenedora de la imagen que usted proporciona. En este caso, no hay ninguna fase de creación.

Para obtener más información, consulte Servicio basado en imágenes.

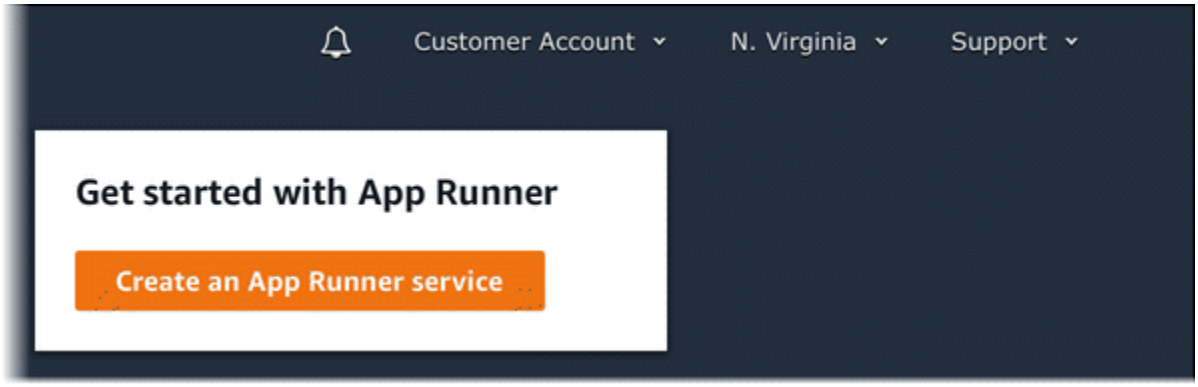
Crear un servicio a partir de una imagen mediante la consola de App Runner

Para crear un servicio de App Runner mediante la consola

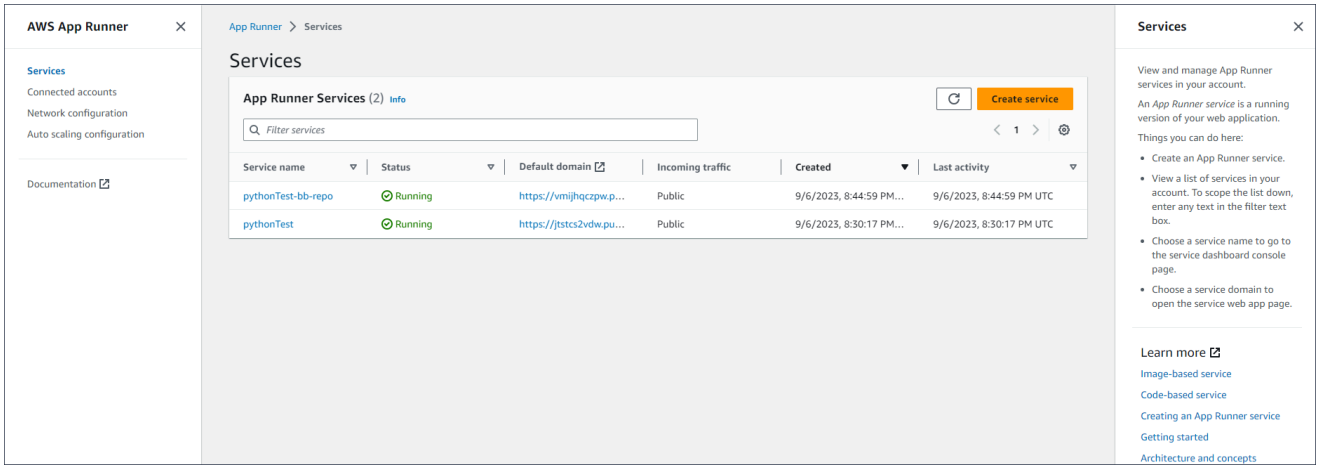
1. Configure el código fuente.

    a. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.

    b. Si aún Cuenta de AWS no tiene ningún servicio de App Runner, aparecerá la página de inicio de la consola. Selecciona Crear un servicio de App Runner.



Si Cuenta de AWS tiene servicios existentes, se muestra la página de servicios con una lista de sus servicios. Elija Crear servicio.



c. En la página Origen e implementación, en la sección Origen, para Tipo de repositorio, elija Registro de contenedores.

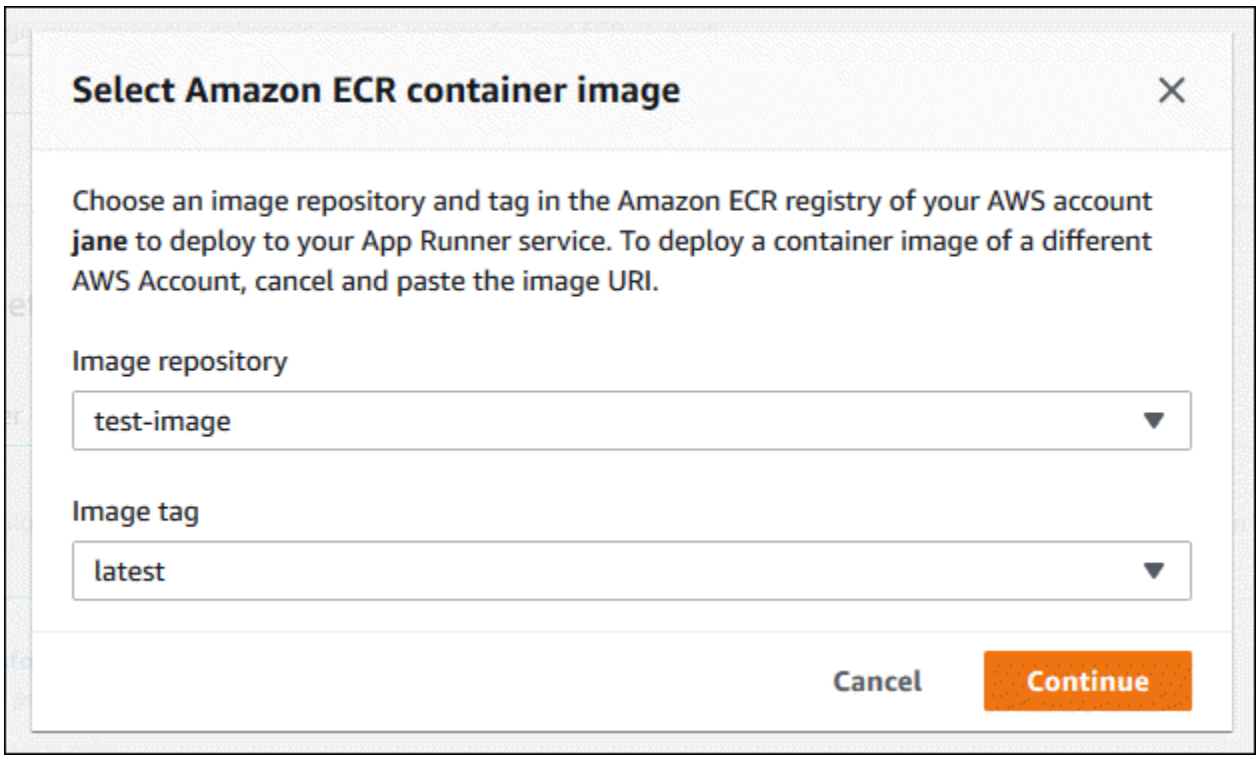
d. En Proveedor, elija el proveedor en el que se almacena la imagen:

- Amazon ECR: imagen privada que se almacena en Amazon ECR.
- Amazon ECR Public: imagen legible públicamente que se almacena en Amazon ECR Public.

e. Para el URI de la imagen del contenedor, seleccione Browse.

f. En el cuadro de diálogo Select Amazon ECR container image, en Repositorio de imágenes, seleccione el repositorio que contiene la imagen.

g. En Etiqueta de imagen, seleccione la etiqueta de imagen específica que desee implementar (por ejemplo, la más reciente) y, a continuación, elija Continuar.



2. Configure sus despliegues.

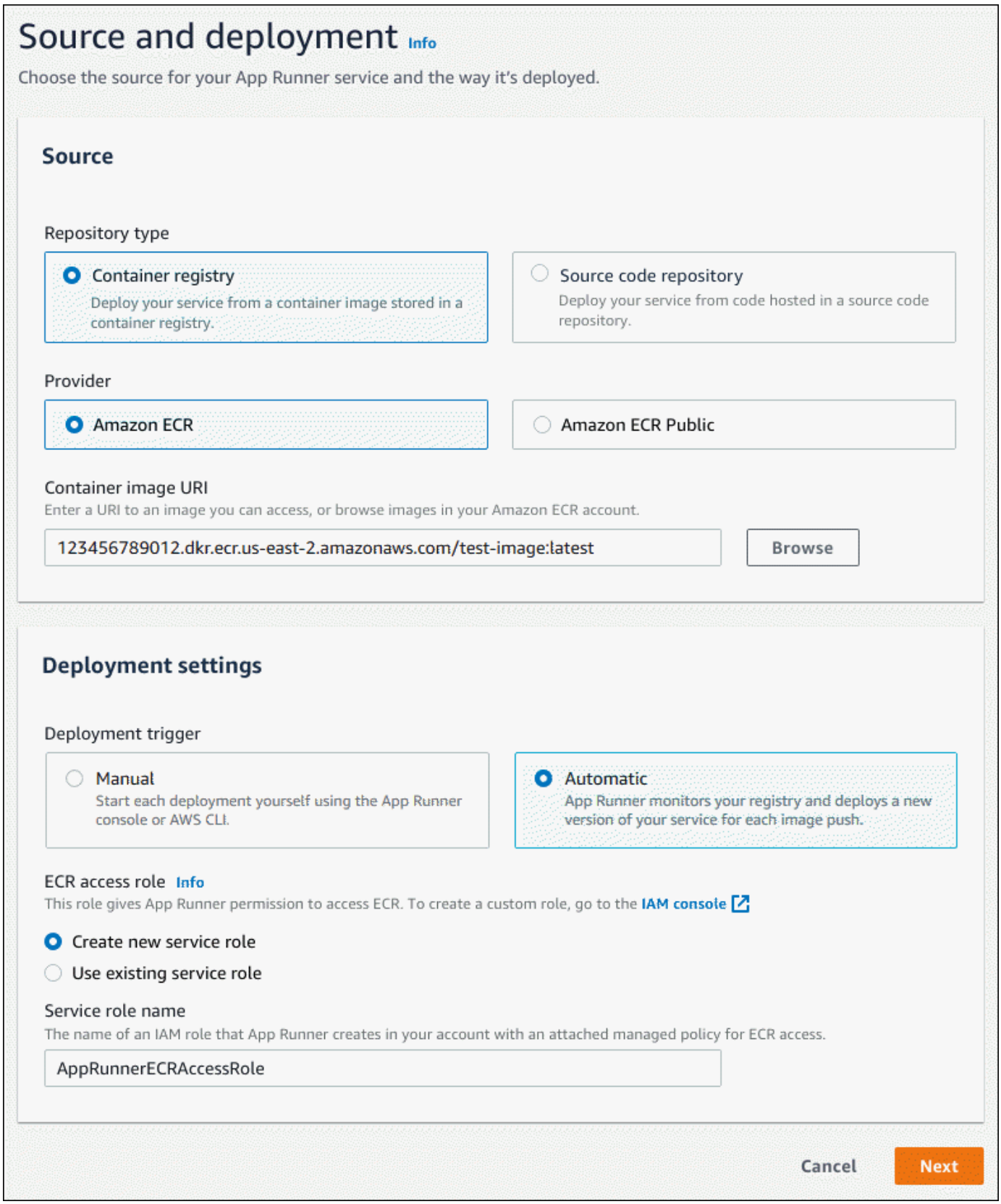
   a. En la sección Configuración de despliegue, elija Manual o Automático.

   > **Note**
   >
   > App Runner no admite la implementación automática de las imágenes públicas de Amazon ECR ni de las imágenes de un repositorio de Amazon ECR que pertenezca a una AWS cuenta diferente de la cuenta en la que se encuentra su servicio.

   Para obtener más información sobre los métodos de implementación, consulte. the section called “Métodos de implementación”

   b. [Proveedor de Amazon ECR] Para el rol de acceso a ECR, elija un rol de servicio existente en su cuenta o cree uno nuevo. Si utiliza el despliegue manual, también puede optar por utilizar el rol de usuario de IAM en el momento del despliegue.

   c. Elija Siguiente.


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source
Repository type
Container registry
Deploy your service from a container image stored in a container registry.
Source code repository
Deploy your service from code hosted in a source code repository.
Provider
Amazon ECR
Amazon ECR Public
Container image URI
Enter a URI to an image you can access, or browse images in your Amazon ECR account.
123456789012.dkr.ecr.us-east-2.amazonaws.com/test-image:latest
Browse
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
App Runner monitors your registry and deploys a new version of your service for each image push.
ECR access role Info
This role gives App Runner permission to access ECR. To create a custom role, go to the IAM console
Create new service role
Use existing service role
Service role name
The name of an IAM role that App Runner creates in your account with an attached managed policy for ECR access.
AppRunnerECRAccessRole
Cancel
Next

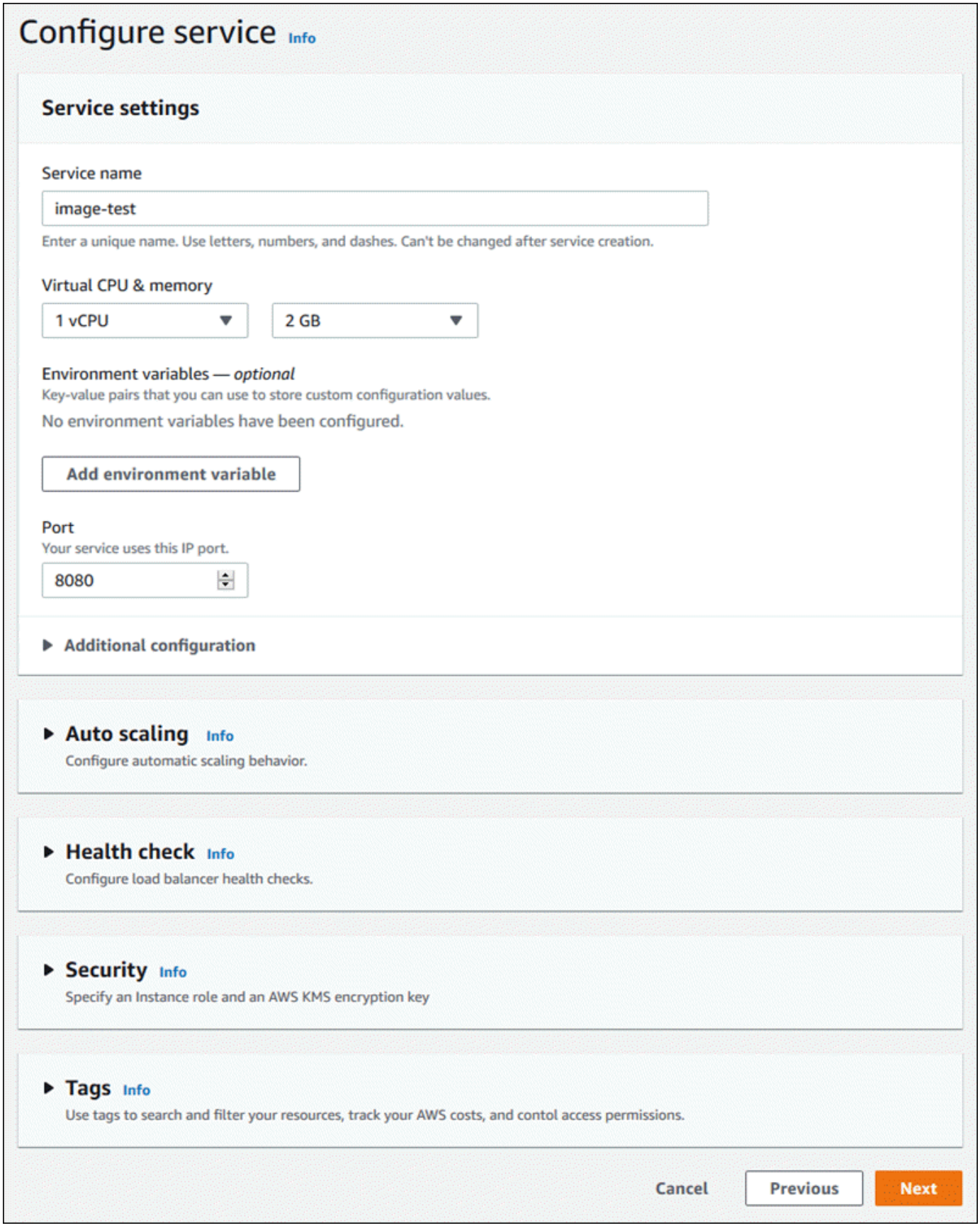
3. Configure su servicio.

   a. En la página Configurar el servicio, en la sección Configuración del servicio, introduzca el nombre del servicio y el puerto IP que escucha el sitio web del servicio.

      > **Note**
      >
      > Todos los demás ajustes del servicio son opcionales o tienen valores predeterminados proporcionados por la consola.

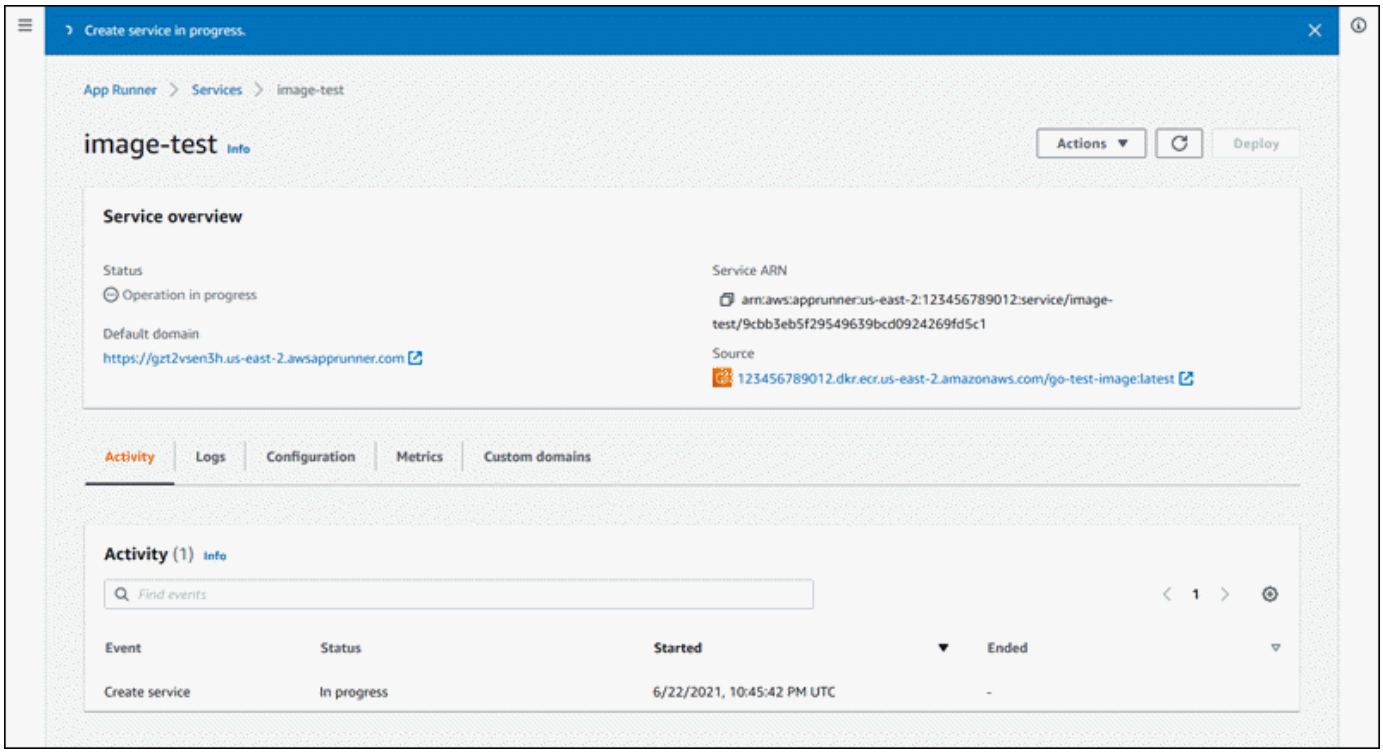
   b. (Opcional) Cambie o añada otras configuraciones para adaptarlas a las necesidades de la aplicación.

   c. Elija Siguiente.


Configure service Info
Service settings
Service name
image-test
Enter a unique name. Use letters, numbers, and dashes. Can't be changed after service creation.
Virtual CPU & memory
1 vCPU
2 GB
Environment variables — optional
Key-value pairs that you can use to store custom configuration values.
No environment variables have been configured.
Add environment variable
Port
Your service uses this IP port.
8080
Additional configuration
Auto scaling Info
Configure automatic scaling behavior.
Health check Info
Configure load balancer health checks.
Security Info
Specify an Instance role and an AWS KMS encryption key
Tags Info
Use tags to search and filter your resources, track your AWS costs, and contol access permissions.
Cancel
Previous
Next

4. En la página Revisar y crear, compruebe todos los detalles que ha introducido y, a continuación, seleccione Crear e implementar.

   Resultado: si el servicio se ha creado correctamente, la consola muestra el panel de control del servicio, con una descripción general del nuevo servicio.



5. Verifica que el servicio esté en ejecución.

   a. En la página del panel de control del servicio, espere hasta que el estado del servicio sea En ejecución.

   b. Elija el valor de dominio predeterminado. Es la URL del sitio web de tu servicio.

   c. Usa tu sitio web y comprueba que funciona correctamente.

Crear un servicio a partir de una imagen mediante la API de App Runner o AWS CLI

Para crear un servicio mediante la API de App Runner o AWS CLI, llama a la acción de la CreateServiceAPI.

La creación del servicio comienza si la llamada devuelve una respuesta correcta y muestra un objeto de servicio`"Status": "CREATING"`.

Para ver un ejemplo de llamada, consulta Cómo crear un servicio de repositorio de imágenes de origen en la referencia de la AWS App Runner API

Reconstrucción de un servicio de App Runner fallido

Si aparece un error al crear un servicio de App Runner, puede realizar una de las siguientes acciones.

- Siga los pasos que se indican the section called “No se pudo crear el servicio” para identificar la causa del error.
- Si encuentra un error en el origen o en la configuración, realice los cambios necesarios y, a continuación, reconstruya el servicio.
- Si un problema temporal con App Runner provocó un error en el servicio, reconstruya el servicio defectuoso sin realizar ningún cambio en la fuente o la configuración.

Puede reconstruir el servicio fallido mediante la consola de App Runner o la API de App Runner o AWS CLI.

Reconstrucción de un servicio de App Runner fallido mediante la consola de App Runner

Rebuild with updates

La creación de un servicio puede fallar por varios motivos. Cuando esto ocurre, es importante identificar y corregir la causa raíz del problema antes de reconstruir el servicio. Para obtener más información, consulte the section called “No se pudo crear el servicio”.

Para reconstruir un servicio fallido con actualizaciones

1. Ve a la pestaña Configuraciones de la página de tu servicio y selecciona Editar.

   La página abre un panel de resumen que muestra una lista de todas las actualizaciones.
2. Realice los cambios necesarios y revíselos en el panel de resumen.
3. Seleccione Guardar y reconstruir.

   Puede supervisar el progreso en la pestaña Registros de la página de su servicio.

Rebuild without updates

Si un problema temporal provoca un error en la creación del servicio, puede volver a crearlo sin modificar su fuente ni sus ajustes de configuración.

Para reconstruir un servicio fallido sin actualizaciones

- Selecciona Reconstruir en la esquina superior derecha de la página de servicio.

  Puedes supervisar el progreso en la pestaña Registros de la página de tu servicio.
- Si el servicio no se puede volver a crear, sigue las instrucciones de solución de problemas que se indican enthe section called “No se pudo crear el servicio”. Realice los cambios necesarios y, a continuación, reconstruya el servicio.

Reconstruir el servicio de App Runner fallido mediante la API de App Runner o AWS CLI

Rebuild with updates

Para reconstruir un servicio fallido:

1. Siga las instrucciones the section called “No se pudo crear el servicio” para encontrar la causa del error.
2. Realice los cambios necesarios en la rama o la imagen del repositorio de origen o en la configuración que causó el error.
3. Realice la reconstrucción mediante una llamada a la acción de la UpdateServiceAPI con los parámetros del nuevo repositorio de código fuente o repositorio de imágenes fuente. App Runner recupera la última confirmación del repositorio de código fuente.

Example Reconstrucción con actualizaciones

En el siguiente ejemplo, se actualiza la configuración de origen de un servicio basado en imágenes. El valor de `Port` se cambia a`80`.

Actualización del `input.json` archivo para el servicio App Runner basado en imágenes

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageConfiguration": {
        "Port": "80"
```

```
      }
    }
  }
}
```

Llamar a la acción `UpdateService` de la API.

```
aws apprunner update-service
--cli-input-json file://input.json
```

Rebuild without updates

Para reconstruir el servicio que ha fallado mediante la API de App Runner o bien AWS CLI, llama a la acción de la UpdateServiceAPI sin realizar ningún cambio en el origen o la configuración del servicio. Elige realizar la reconstrucción sin realizar actualizaciones solo si la creación del servicio ha fallado debido a un problema temporal con App Runner.

Implementación de una nueva versión de la aplicación en App Runner

Al crear un servicio en AWS App Runner, se configura una fuente de aplicación: una imagen de contenedor o un repositorio de fuentes. App Runner aprovisiona recursos para ejecutar el servicio e implementa la aplicación en ellos.

En este tema se describen las formas de volver a implementar la fuente de la aplicación en el servicio de App Runner cuando haya una nueva versión disponible. Puede ser una nueva versión de imagen en el repositorio de imágenes o una nueva confirmación en el repositorio de código. App Runner proporciona dos métodos de implementación en un servicio: automático y manual.

Métodos de implementación

App Runner proporciona los siguientes métodos para controlar cómo se inician las implementaciones de las aplicaciones.

Despliegue automático

Utilice el despliegue automático cuando desee un comportamiento continuo de integración y despliegue (CI/CD) para su servicio. App Runner supervisa el repositorio de imágenes o códigos para detectar cambios.

Repositorio de imágenes: cada vez que insertas una nueva versión de una imagen en tu repositorio de imágenes o una nueva confirmación en tu repositorio de código, App Runner la despliega automáticamente en tu servicio sin que tengas que hacer nada más.

Repositorio de código: cada vez que insertas una nueva confirmación en tu repositorio de código que realiza cambios en el directorio de origen, App Runner despliega todo el repositorio. Como solo los cambios en el directorio de origen activan una implementación automática, es importante entender cómo afecta la ubicación del directorio de origen al alcance de una implementación automática.

- Top-level directorio (raíz del repositorio): este es el valor predeterminado que se establece para el directorio de origen al crear un servicio. Si tu directorio de origen está establecido en este valor, significa que todo el repositorio está dentro del directorio de origen. Por lo tanto, todas las confirmaciones que envíes al repositorio de origen activarán una implementación en este caso.
- Cualquier ruta de directorio que no sea la raíz del repositorio (no predeterminada): dado que solo los cambios que se inserten en el directorio de origen activarán una implementación automática, cualquier cambio que se inserte en tu repositorio que no esté en el directorio de origen no activará una implementación automática. Por lo tanto, debes usar un despliegue manual para implementar los cambios que insertes fuera del directorio de origen.

> **Note**
>
> App Runner no admite la implementación automática de las imágenes públicas de Amazon ECR ni de las imágenes de un repositorio de Amazon ECR que pertenezca a una AWS cuenta diferente de la cuenta en la que se encuentra su servicio.

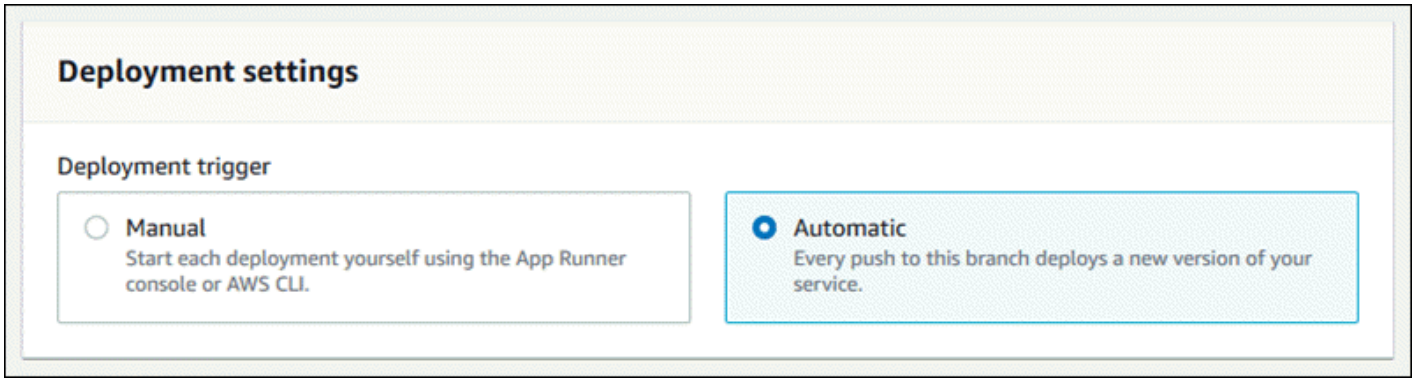
Implementación manual

Utilice el despliegue manual cuando desee iniciar de forma explícita cada despliegue en su servicio. Usted inicia una implementación si el repositorio que configuró para su servicio tiene una nueva versión que desee implementar. Para obtener más información, consulte the section called “Implementación manual”.

> **Note**
>
> Cuando ejecutas una implementación manual, App Runner implementa la fuente desde el repositorio completo.

Puede configurar el método de implementación de su servicio de las siguientes maneras:

- Consola: para un servicio nuevo que esté creando o para un servicio existente, en la sección Configuración de despliegue de la página de configuración de origen e implementación, seleccione Manual o Automático.



- API o AWS CLI: en una llamada a la UpdateServiceacción CreateServiceo, defina el `AutoDeploymentsEnabled` miembro del SourceConfigurationparámetro como `False` para el despliegue manual o `True` automático.

> Comparación de despliegues automáticos y manuales
>
> Tanto las implementaciones automáticas como las manuales producen el mismo resultado: ambos métodos implementan el repositorio completo.
> La diferencia entre los dos métodos es el mecanismo de activación:
>
> - Las implementaciones manuales se activan mediante una implementación desde la consola, una llamada a la API de App Runner o una llamada a la API de App Runner. AWS CLIImplementación manualEn la siguiente sección, se proporcionan los procedimientos correspondientes.
> - Los despliegues automáticos se activan cuando se produce un cambio en el contenido del directorio de origen.

Implementación manual

En el caso de la implementación manual, debe iniciar de forma explícita cada implementación en su servicio. Cuando tenga una nueva versión de la imagen o el código de la aplicación lista para su

implementación, puede consultar las siguientes secciones para obtener información sobre cómo realizar una implementación mediante la consola y la API.

> **Note**
>
> Cuando ejecutas una implementación manual, App Runner implementa el código fuente del repositorio completo.

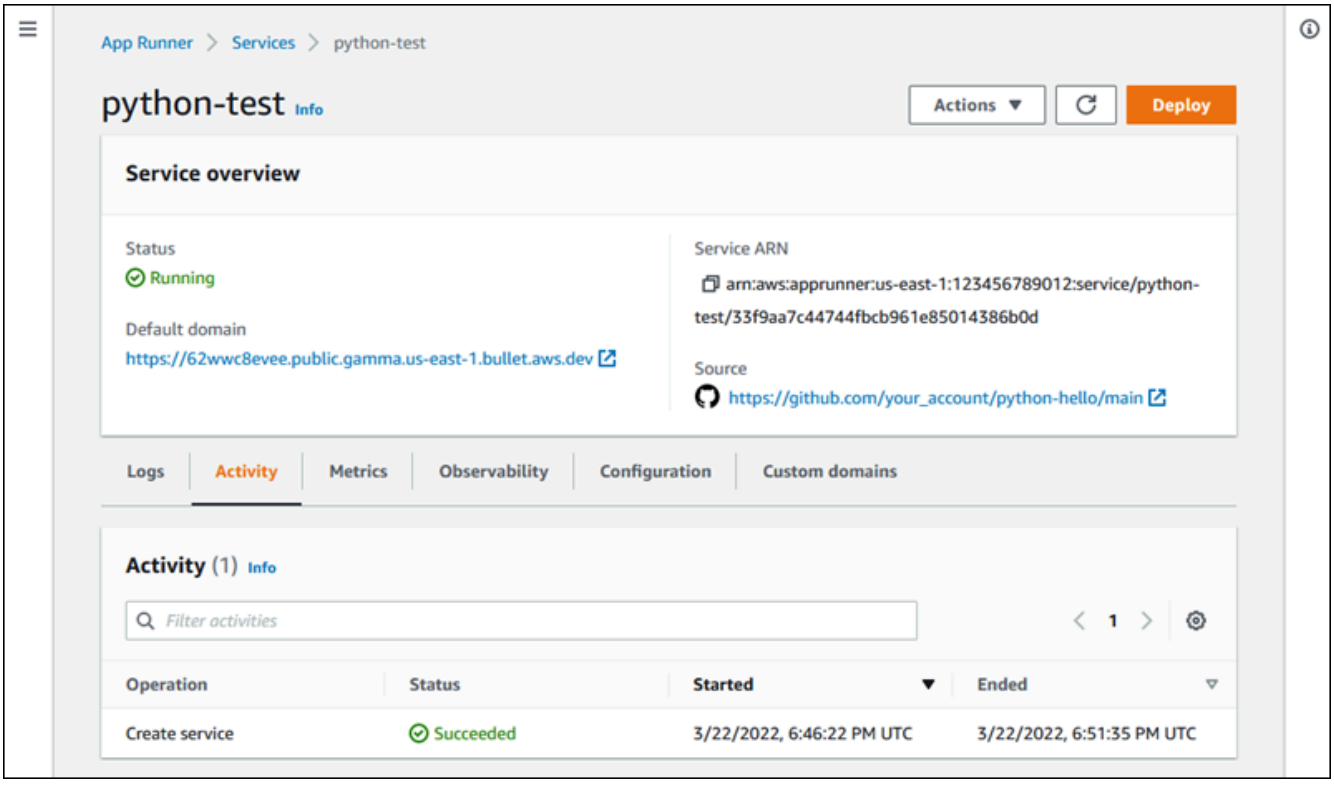
Implementa una versión de tu aplicación mediante uno de los siguientes métodos:

App Runner console

Para realizar la implementación mediante la consola de App Runner

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.



3. Elija Implementar.

Resultado: se inicia el despliegue de la nueva versión. En la página del panel de control del servicio, el estado del servicio cambia a Funcionamiento en curso.

4. Espere a que finalice la implementación. En la página del panel de control del servicio, el estado del servicio debería volver a ser En ejecución.
5. Para comprobar que la implementación se ha realizado correctamente, en la página del panel de control del servicio, elige el valor de dominio predeterminado, que es la URL del sitio web del servicio. Inspecciona tu aplicación web o interactúa con ella y verifica el cambio de versión.

> **Note**
>
> Para aumentar la seguridad de sus aplicaciones de App Runner, el dominio*.awsapprunner.com está registrado en la lista pública de sufijos (PSL). Para mayor seguridad, te recomendamos que utilices cookies con un `__Host-` prefijo si alguna vez necesitas configurar cookies confidenciales en el nombre de dominio predeterminado de tus aplicaciones de App Runner. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulta la Set-Cookiepágina de la Red de desarrolladores de Mozilla.

App Runner API or AWS CLI

Para realizar la implementación mediante la API de App Runner o AWS CLI, llame a la acción de la StartDeploymentAPI. El único parámetro que se debe pasar es el ARN de su servicio. Ya configuró la ubicación de origen de la aplicación cuando creó el servicio y App Runner puede encontrar la nueva versión. La implementación comienza si la llamada devuelve una respuesta correcta.

Configuración de un servicio de App Runner

Al crear un AWS App Runner servicio, se establecen varios valores de configuración. Puede cambiar algunos de estos valores de configuración después de crear el servicio. Otros ajustes solo se pueden aplicar al crear el servicio y no se pueden cambiar posteriormente. En este tema se describe la configuración del servicio mediante la API de App Runner, la consola de App Runner y un archivo de configuración de App Runner.

Temas

Configure su servicio mediante la API de App Runner o AWS CLI

La API define qué ajustes se pueden cambiar después de la creación del servicio. En la siguiente lista se describen las acciones, los tipos y las limitaciones relevantes.

- UpdateServiceacción: se puede ejecutar después de la creación para actualizar algunos valores de configuración.
  - Se puede actualizar: puede actualizar la configuración de los `HealthCheckConfiguration` parámetros `SourceConfiguration` `InstanceConfiguration`, y. Sin embargo`SourceConfiguration`, en, no puedes cambiar el tipo de fuente de código a imagen o viceversa. Debes proporcionar el mismo parámetro de repositorio que proporcionaste cuando creaste el servicio. Es una u otra. `CodeRepository` `ImageRepository`

    También puede actualizar los siguientes ARN de los distintos recursos de configuración asociados al servicio:

    - `AutoScalingConfigurationArn`
    - `VpcConnectorArn`
  - No se puede actualizar: no se pueden cambiar `ServiceName` los `EncryptionConfiguration` parámetros que están disponibles en la CreateServiceacción. No se pueden cambiar una vez creados. La UpdateServiceacción no incluye estos parámetros.
  - API o archivo: puedes establecer el `ConfigurationSource` parámetro del CodeConfigurationtipo (que se utiliza para los repositorios de código fuente como parte de él`SourceConfiguration`) en. `Repository` En este caso, App Runner ignora las opciones de `CodeConfigurationValues` configuración y las lee de un archivo de configuración del repositorio. Si lo `ConfigurationSource` establece`API`, App Runner obtiene todos los ajustes de configuración de la llamada a la API e ignora el archivo de configuración, incluso si existe alguno.

- TagResourceacción: se puede llamar después de crear el servicio para añadir etiquetas al servicio o actualizar los valores de las etiquetas existentes.
- UntagResourceacción: se puede llamar después de crear el servicio para eliminar las etiquetas del servicio.

> **Note**
>
> Si crea un conector de VPC de tráfico saliente para un servicio, el proceso de inicio del servicio que sigue experimentará una latencia única. Puedes establecer esta configuración para un servicio nuevo al crearlo o, posteriormente, con una actualización del servicio. Para obtener más información, consulte Latencia única el capítulo Redes con App Runner de esta guía.

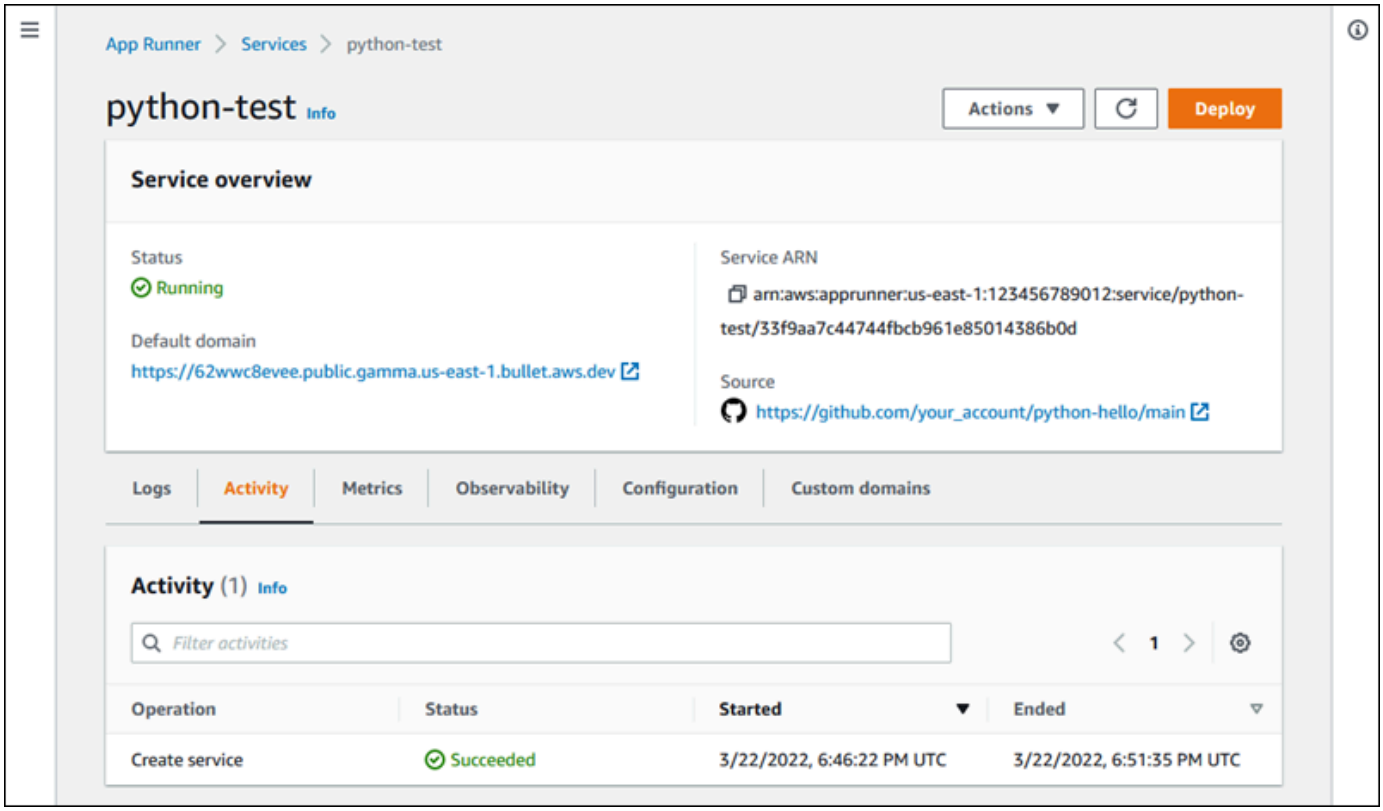
Configure su servicio mediante la consola de App Runner

La consola usa la API de App Runner para aplicar las actualizaciones de configuración. Las reglas de actualización que impone la API, tal como se definieron en la sección anterior, determinan lo que se puede configurar con la consola. Algunos ajustes que estaban disponibles durante la creación del servicio no están disponibles para modificarlos más adelante. Además, si decides usar un archivo de configuración, los ajustes adicionales se ocultarán en la consola y App Runner los leerá del archivo.

Para configurar el servicio

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, seleccione la pestaña Configuración.

   Resultado: la consola muestra los ajustes de configuración actuales del servicio en varias secciones: Origen e implementación, Configuración de compilación y Configuración del servicio.

4. Para actualizar los ajustes de cualquier categoría, seleccione Editar.

5. En la página de edición de la configuración, realice los cambios que desee y, a continuación, seleccione Guardar cambios.

> **Note**
>
> Si crea un conector de VPC de tráfico saliente para un servicio, el proceso de inicio del servicio que sigue experimentará una latencia única. Puedes establecer esta configuración para un servicio nuevo al crearlo o, posteriormente, con una actualización del servicio. Para obtener más información, consulte Latencia única el capítulo Redes con App Runner de esta guía.

Configure su servicio mediante un archivo de configuración de App Runner

Al crear o actualizar un servicio de App Runner, puede indicarle a App Runner que lea algunos ajustes de configuración de un archivo de configuración que proporcione como parte de su repositorio de origen. De este modo, puede administrar los ajustes relacionados con su código fuente bajo el control de código fuente, junto con el propio código. El archivo de configuración también proporciona algunos ajustes avanzados que no se pueden configurar mediante la consola o la API. Para obtener más información, consulte Archivo de configuración de App Runner.

> **Note**
>
> Si crea un conector de VPC de tráfico saliente para un servicio, el proceso de inicio del servicio que sigue experimentará una latencia única. Puedes establecer esta configuración para un servicio nuevo al crearlo o, posteriormente, con una actualización del servicio. Para obtener más información, consulte Latencia única el capítulo Redes con App Runner de esta guía.

Configuración de la observabilidad para su servicio

AWS App Runner se integra con varios AWS servicios para proporcionarte un amplio conjunto de herramientas de observabilidad para tu servicio App Runner. Para obtener más información, consulte Observabilidad.

App Runner permite habilitar algunas funciones de observabilidad y configurar su comportamiento mediante un recurso compartible llamado. ObservabilityConfiguration Puedes proporcionar un recurso de configuración de observabilidad al crear o actualizar un servicio. La consola de App Runner crea uno para ti cuando creas un nuevo servicio de App Runner. Proporcionar una configuración de observabilidad es opcional. Si no la proporcionas, App Runner proporciona una configuración de observabilidad predeterminada.

Puedes compartir una única configuración de observabilidad en varios servicios de App Runner para asegurarte de que tengan el mismo comportamiento de observabilidad. Para obtener más información, consulte the section called “Recursos de configuración”.

Puedes configurar las siguientes funciones de observabilidad mediante configuraciones de observabilidad:

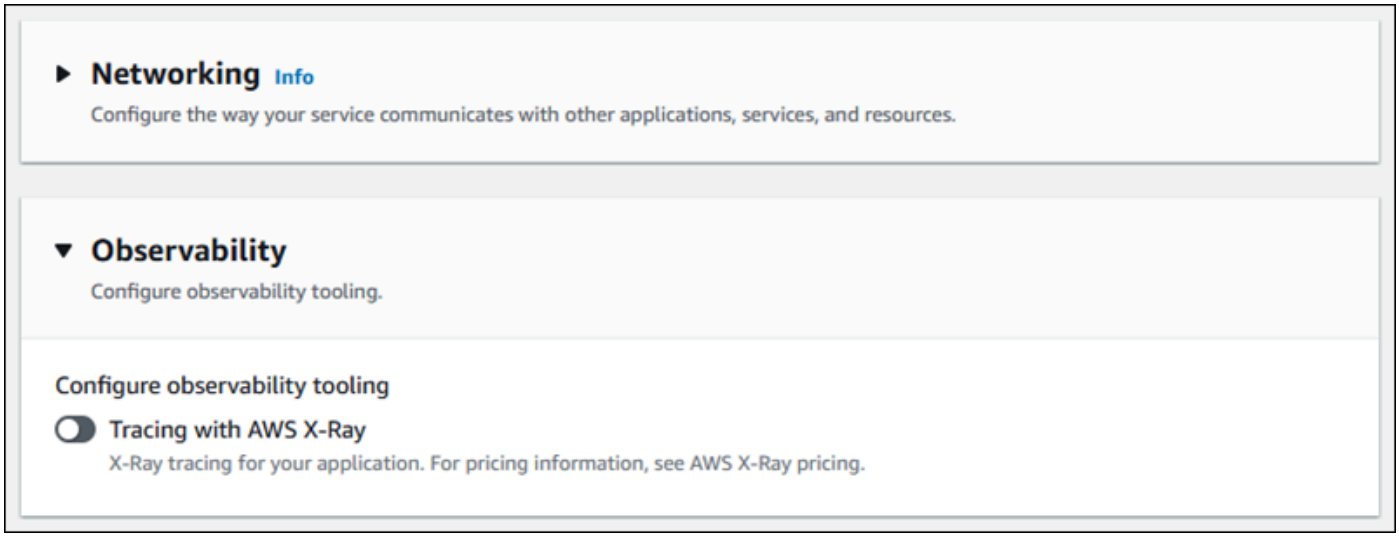
- Configuración de rastreo: ajustes para rastrear las solicitudes que atiende su aplicación y las llamadas posteriores que realiza. Para obtener más información sobre el seguimiento, consulte the section called “Rastreo (X-Ray)”.

Gestione la observabilidad

Administra la observabilidad de tus servicios de App Runner mediante uno de los siguientes métodos:

App Runner console

Al crear un servicio mediante la consola de App Runner o al actualizar su configuración más adelante, puede configurar las funciones de observabilidad de su servicio. Busca la sección de configuración de observabilidad en la página de la consola.



App Runner API or AWS CLI

Cuando llamas a las acciones de la API CreateServiceo a la API de UpdateServiceApp Runner, puedes usar el objeto de `ObservabilityConfiguration` parámetro para habilitar las funciones de observabilidad y especificar un recurso de configuración de observabilidad para tu servicio.

Usa las siguientes acciones de la API de App Runner para administrar tus recursos de configuración de observabilidad.

- CreateObservabilityConfiguration— Crea una nueva configuración de observabilidad o una revisión de una existente.

- ListObservabilityConfigurations— Devuelve una lista de las configuraciones de observabilidad asociadas a la suya Cuenta de AWS, con información resumida.
- DescribeObservabilityConfiguration— Devuelve una descripción completa de una configuración de observabilidad.
- DeleteObservabilityConfiguration— Elimina una configuración de observabilidad. Puede eliminar una revisión específica o la última revisión activa. Es posible que tengas que eliminar las configuraciones de observabilidad innecesarias si alcanzas la cuota de configuración de observabilidad correspondiente a tu. Cuenta de AWS

Configurar los ajustes del servicio mediante recursos que se pueden compartir

Para algunas funciones, tiene sentido compartir la configuración entre AWS App Runner los servicios. Por ejemplo, es posible que desee que un conjunto de servicios tenga el mismo comportamiento de escalado automático. O puede que desee una configuración de observabilidad idéntica para todos sus servicios. App Runner te permite compartir la configuración mediante el uso de recursos compartibles independientes. Se crea un recurso que define un conjunto de ajustes de configuración para una función y, a continuación, se proporciona el nombre de recurso de Amazon (ARN) de este recurso de configuración a uno o más servicios de App Runner.

App Runner implementa recursos de configuración que se pueden compartir para las siguientes funciones:

- Escalado automático
- Observabilidad
- Acceso mediante VPC

La página del documento de cada una de estas funciones proporciona información sobre la configuración disponible y los procedimientos de administración.

Las funciones que utilizan recursos de configuración independientes comparten algunas características y consideraciones de diseño.

- Revisiones: algunos recursos de configuración pueden tener revisiones. El escalado automático y la observabilidad son ejemplos de dos recursos de configuración que utilizan revisiones. En estos casos, cada configuración tiene un nombre y una revisión numérica. Varias revisiones de una

configuración tienen el mismo nombre y números de revisión diferentes. Puede usar diferentes nombres de configuración para diferentes escenarios. Para cada nombre, puede añadir varias revisiones para ajustar la configuración de un escenario específico.

La primera configuración que cree con un nombre recibe el número de revisión 1. Las configuraciones posteriores con el mismo nombre obtienen números de revisión consecutivos (empezando por 2). Puedes asociar tu servicio App Runner a una revisión de configuración específica o a la última revisión de la configuración.

- Compartido: puedes compartir un único recurso de configuración entre varios servicios de App Runner. Esto resulta útil si desea mantener configuraciones idénticas en todos estos servicios. En concreto, si sus recursos admiten revisiones, puede configurar varios servicios para usar la última revisión de una configuración. Para ello, especifique solo el nombre de la configuración, pero no una revisión. Todos los servicios que haya configurado de esta manera reciben actualizaciones de configuración al actualizar el servicio. Para obtener más información sobre los cambios de configuración, consultethe section called “Configuración”.
- Administración de recursos: puede usar App Runner para crear y eliminar configuraciones. No puedes actualizar una configuración directamente. En su lugar, en el caso de los recursos que admiten revisiones, puede crear una nueva revisión del nombre de una configuración existente para actualizar la configuración de manera efectiva.

> **Note**
>
> Para el escalado automático, puedes crear configuraciones y múltiples revisiones tanto con la consola de App Runner como con la API de App Runner. Tanto la consola de App Runner como la API de App Runner también pueden eliminar configuraciones y revisiones. Para obtener más información, consulte Administrar el escalado automático de App Runner.
> Para otros tipos de configuración, como las configuraciones de observabilidad, solo puedes crear una configuración con una única revisión con la consola de App Runner. Para crear más revisiones y eliminar configuraciones, debes usar la API de App Runner.

- Cuota de recursos: hay cuotas establecidas para la cantidad de nombres y revisiones de configuración únicos que puede tener para sus recursos de configuración en cada uno Región de AWS. Si alcanza estas cuotas, debe eliminar el nombre de una configuración o al menos algunas de sus revisiones antes de poder crear más. Para las revisiones de las configuraciones de escalado automático, puedes usar la consola de App Runner o la API de App Runner para eliminarlas. Para obtener más información, consulte Administrar el escalado automático de

App Runner. Debe usar la API de App Runner para eliminar otros recursos. Para obtener más información sobre las cuotas, consulte the section called “Cuotas de recursos de App Runner”.

- Sin coste de recursos: no se incurre en costes adicionales por crear un recurso de configuración. Es posible que incurra en costes por la función en sí (por ejemplo, se le cobra el AWS X-Ray coste normal al activar el X-Ray seguimiento), pero no por el recurso de configuración de App Runner que configura la función para su servicio de App Runner.

Configurar las comprobaciones de estado de su servicio

AWS App Runner supervisa el estado de su servicio mediante la realización de comprobaciones de estado. El protocolo de comprobación de estado predeterminado es TCP. App Runner hace ping al dominio asignado a tu servicio. También puedes configurar el protocolo de verificación de estado en HTTP. App Runner envía las solicitudes HTTP de verificación de estado a tu aplicación web.

Puede configurar algunos ajustes relacionados con las comprobaciones de estado. En la siguiente tabla se describen los ajustes de las comprobaciones de estado y sus valores predeterminados.

Opción	Descripción	Predeterminado
Protocolo	El protocolo IP que App Runner utiliza para realizar las comprobaciones de estado de su servicio. Si configuras el protocolo en`TCP`, App Runner hace ping al dominio predeterminado asignado a tu servicio en el puerto que escucha tu aplicación. Si configuras el protocolo en`HTTP`, App Runner envía las solicitudes de comprobación de estado a la ruta configurada.	`TCP`
Ruta	La URL a la que App Runner envía las solicitudes de comprobación de estado HTTP. Aplicable solo a las comprobaciones HTTP.	`/`
Interval	El intervalo de tiempo, en segundos, que transcurre entre las comprobaciones de estado.	5
Timeout (Tiempo de espera)	El tiempo, en segundos, para esperar una respuesta de comprobación de estado antes de decidir que ha fallado.	2

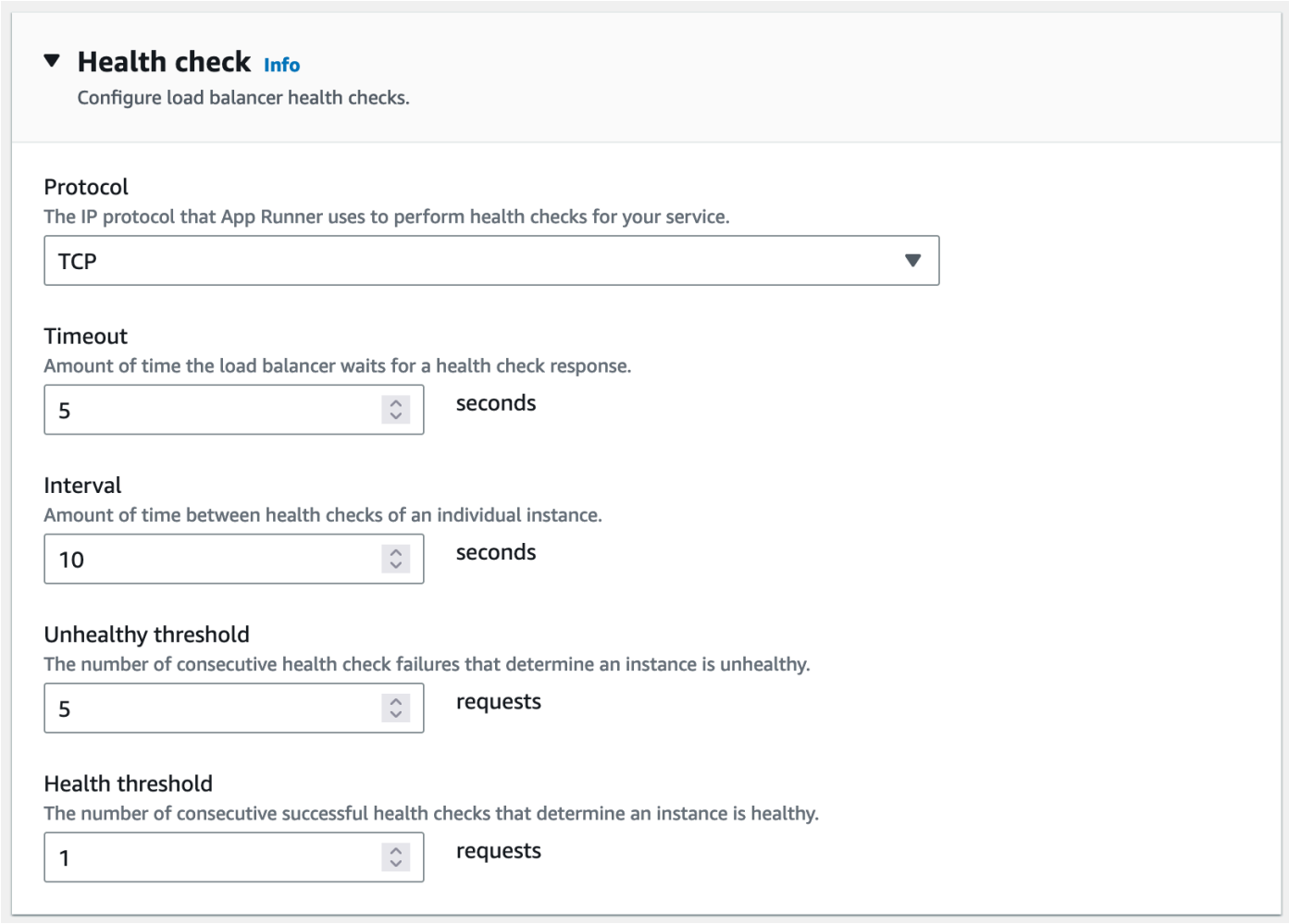
Opción	Descripción	Predeterminado
Umbral saludable	El número de comprobaciones consecutivas que deben tener éxito antes de que App Runner decida que el servicio es saludable.	1
Umbral poco saludable	El número de comprobaciones consecutivas que deben fallar antes de que App Runner decida que el servicio no es saludable.	5

Configurar comprobaciones de estado

Configura las comprobaciones de estado de tu servicio de App Runner mediante uno de los siguientes métodos:

App Runner console

Al crear el servicio de App Runner mediante la consola de App Runner o al actualizar su configuración más adelante, puede configurar los ajustes de comprobación de estado. Para ver todos los procedimientos de la consola, consulte the section called “Creación” ythe section called “Configuración”. En ambos casos, busque la sección de configuración de Health Check en la página de la consola.



App Runner API or AWS CLI

Cuando llamas a las acciones CreateServiceo a las de la UpdateServiceAPI, puedes usar el `HealthCheckConfiguration` parámetro para especificar la configuración de las comprobaciones de estado.

Para obtener información sobre la estructura del parámetro, consulta HealthCheckConfigurationla referencia de la AWS App Runner API.

Administrar las conexiones de App Runner

Al crear un servicio en AWS App Runner, se configura una fuente de aplicación: una imagen de contenedor o un repositorio de fuentes que se almacena en un proveedor. App Runner debe establecer una conexión autenticada y autorizada con el proveedor. Luego, App Runner puede leer tu repositorio e implementarlo en tu servicio. App Runner no requiere el establecimiento de una conexión cuando creas un servicio que accede al código almacenado en tu Cuenta de AWS.

App Runner mantiene la información de conexión en un recurso denominado conexión. La consola de App Runner y esta guía también se refieren a las conexiones como cuentas conectadas. App Runner requiere un recurso de conexión cuando creas un servicio que necesita información de conexión de terceros. La siguiente es información importante sobre las conexiones:

- Proveedores: actualmente, App Runner requiere recursos de conexión con Bitbucket GitHubo Bitbucket.
- Compartido: puedes usar un recurso de conexión para crear varios servicios de App Runner que usen la misma cuenta de proveedor de repositorios.
- Administración de recursos: en App Runner, puede crear y eliminar conexiones. Sin embargo, no puedes modificar una conexión existente.
- Cuota de recursos: los recursos de conexión tienen una cuota establecida asociada a la tuya Cuenta de AWS en cada uno de ellos Región de AWS. Si alcanzas esta cuota, es posible que tengas que eliminar una conexión antes de poder conectarte a una nueva cuenta de proveedor. Puedes eliminar una conexión mediante la consola o la API de App Runner, tal y como se describe en la siguiente sección,the section called “Administra las conexiones”. Para obtener más información, consulte the section called “Cuotas de recursos de App Runner”.

Administra las conexiones

Administra tus conexiones de App Runner mediante uno de los siguientes métodos:

App Runner console

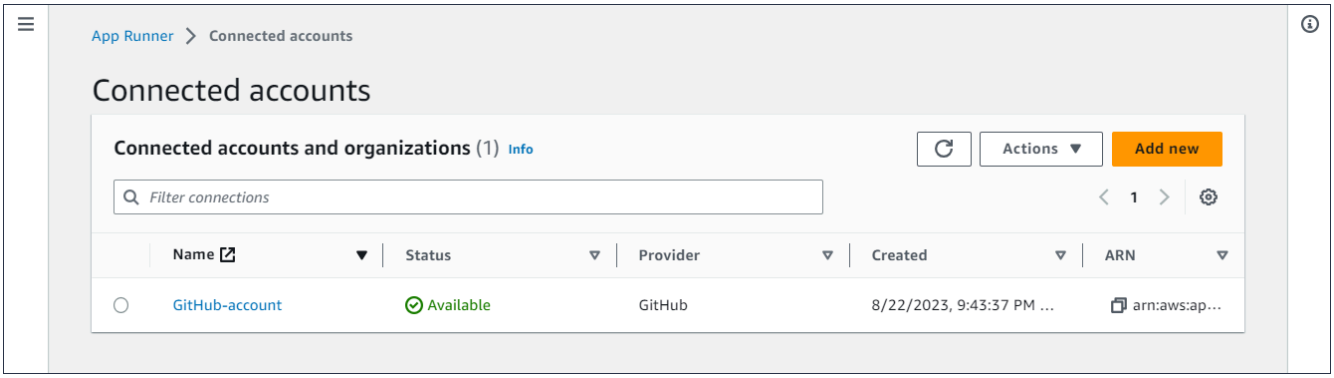
Cuando utilizas la consola de App Runner para crear un servicio, proporcionas los detalles de la conexión. No tienes que crear un recurso de conexión de forma explícita. En la consola, puedes elegir conectarte a una cuenta GitHub o a una cuenta de Bitbucket a la que te hayas conectado anteriormente, o conectarte a una cuenta nueva. Cuando sea necesario, App Runner crea un recurso de conexión para ti. Para una conexión nueva, algunos proveedores requieren que completes un protocolo de autenticación antes de poder usar la conexión. La consola le guiará por este proceso.

La consola también tiene una página para administrar las conexiones existentes. Puedes completar el protocolo de autenticación de una conexión si no lo hiciste cuando creaste el servicio. También puedes eliminar las conexiones que ya no utilices. El siguiente procedimiento muestra cómo administrar las conexiones de los proveedores de repositorios.

Para administrar las conexiones de su cuenta

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Cuentas conectadas.

   A continuación, la consola muestra una lista de las conexiones de proveedores de repositorios de su cuenta.



3. Ahora puedes realizar una de las siguientes acciones con cualquier conexión de la lista:

   - Abrir GitHub/Bitbucket cuenta u organización: elige el nombre de la conexión.
   - Protocolo de autenticación completo: selecciona la conexión y, a continuación, en el menú Acciones, selecciona Protocolo de enlace completo. La consola le guiará por el proceso de protocolo de autenticación.
   - Eliminar conexión: seleccione la conexión y, a continuación, en el menú Acciones, elija Eliminar. Siga las instrucciones de la solicitud de eliminación.

App Runner API or AWS CLI

Puedes usar las siguientes acciones de la API de App Runner para administrar tus conexiones.

- CreateConnection— Crea una conexión a una cuenta de proveedor de repositorios. Una vez creada la conexión, debes completar manualmente el protocolo de autenticación mediante la consola de App Runner. Este proceso se explica en la sección anterior.
- ListConnections— Devuelve una lista de las conexiones de App Runner asociadas a tu Cuenta de AWS.
- DeleteConnection— Elimina una conexión. Es posible que tengas que eliminar las conexiones innecesarias si alcanzas la cuota de conexión de tu Cuenta de AWS.

Administrar el escalado automático de App Runner

AWS App Runner escala automáticamente los recursos informáticos, específicamente las instancias, hacia arriba o hacia abajo para su aplicación App Runner. El escalado automático proporciona una gestión adecuada de las solicitudes cuando hay mucho tráfico y reduce los costes cuando el tráfico se ralentiza.

Configuración de escalado automático

Puede configurar algunos parámetros para ajustar el comportamiento de escalado automático de su servicio. App Runner mantiene la configuración de escalado automático en un recurso que se puede compartir llamado AutoScalingConfiguration. Puede crear y mantener configuraciones de escalado automático independientes antes de asignarlas a los servicios. Una vez que se hayan asociado a un servicio, puede seguir manteniendo las configuraciones. También puede optar por crear una nueva configuración de autoescalado mientras está en el proceso de crear un nuevo servicio o configurar uno existente. Una vez creada la nueva configuración de autoescalado, puede asociarla al servicio y continuar con el proceso de creación o configuración del servicio.

Asignación de nombres y revisiones

Una configuración de autoescalado tiene un nombre y una revisión numérica. Varias revisiones de una configuración tienen el mismo nombre y números de revisión diferentes. Puede usar diferentes nombres de configuración para diferentes escenarios de escalado automático, como alta disponibilidad o bajo costo. Para cada nombre, puede añadir varias revisiones para ajustar la configuración de un escenario específico. Puede tener hasta 10 nombres de configuración de escalado automático únicos y hasta 5 revisiones para cada configuración. Si alcanza el límite y necesita crear más, puede eliminar una y, a continuación, crear otra. App Runner no te permitirá eliminar una configuración que esté establecida como predeterminada o que esté siendo utilizada por un servicio activo. Para obtener más información sobre las cuotas, consulte the section called “Cuotas de recursos de App Runner”.

Establecer una configuración predeterminada

Al crear o actualizar un servicio de App Runner, puede proporcionar un recurso de configuración de autoescalado. Proporcionar una configuración de escalado automático es opcional. Si no lo proporcionas, App Runner proporciona una configuración de escalado automático predeterminada con los valores recomendados. La función de configuración de autoescalado te ofrece la opción de establecer tu propia configuración de escalado automático predeterminada en lugar de

usar la configuración predeterminada que proporciona App Runner. Una vez que especifique otra configuración de autoescalado como predeterminada, esa configuración se asignará automáticamente como predeterminada a los nuevos servicios que cree en el futuro. La nueva designación predeterminada no afecta a las asociaciones que se establecieron anteriormente para los servicios existentes.

Configuración de servicios con escalado automático

Puede compartir una única configuración de escalado automático entre varios servicios de App Runner para garantizar que los servicios tengan el mismo comportamiento de escalado automático. Para obtener más información sobre cómo configurar las configuraciones de autoescalado con la consola de App Runner o la API de App Runner, consulte las secciones siguientes de este tema. Para obtener más información general sobre los recursos que se pueden compartir, consultethe section called “Recursos de configuración”.

Parámetros configurables

Puede configurar los siguientes ajustes de escalado automático:

- Simultaneidad máxima: el número máximo de solicitudes simultáneas que procesa una instancia. Cuando el número de solicitudes simultáneas supera esta cuota, App Runner amplía el servicio.
- Tamaño máximo: la cantidad máxima de instancias a las que puede ampliarse tu servicio. Es el número máximo de instancias que pueden gestionar el tráfico de tu servicio de forma simultánea.
- Tamaño mínimo: el número mínimo de instancias que App Runner puede aprovisionar para tu servicio. El servicio siempre tiene al menos este número de instancias aprovisionadas. Algunas de estas instancias gestionan el tráfico de forma activa. El resto forma parte de la rentable reserva de capacidad informática, que está lista para activarse rápidamente. Usted paga por el uso de memoria de todas las instancias aprovisionadas. Solo paga por el uso de la CPU del subconjunto activo.

> **Note**
>
> El recuento de recursos de vCPU determina la cantidad de instancias que App Runner puede proporcionar a su servicio. Se trata de un valor de cuota ajustable para el recuento de recursos de On-DemandvCPU de Fargate que reside en el servicio. AWS Fargate Para ver la configuración de la cuota de vCPU de su cuenta o para solicitar un aumento de la cuota, utilice la consola Service Quotas del. Consola de administración de AWS Para obtener más

información, consulte las cuotas AWS Fargate de servicio en la Guía para desarrolladores de Amazon Elastic Container Service.

Administre el escalado automático de un servicio

Administra el escalado automático de tus servicios de App Runner mediante uno de los siguientes métodos:

App Runner console

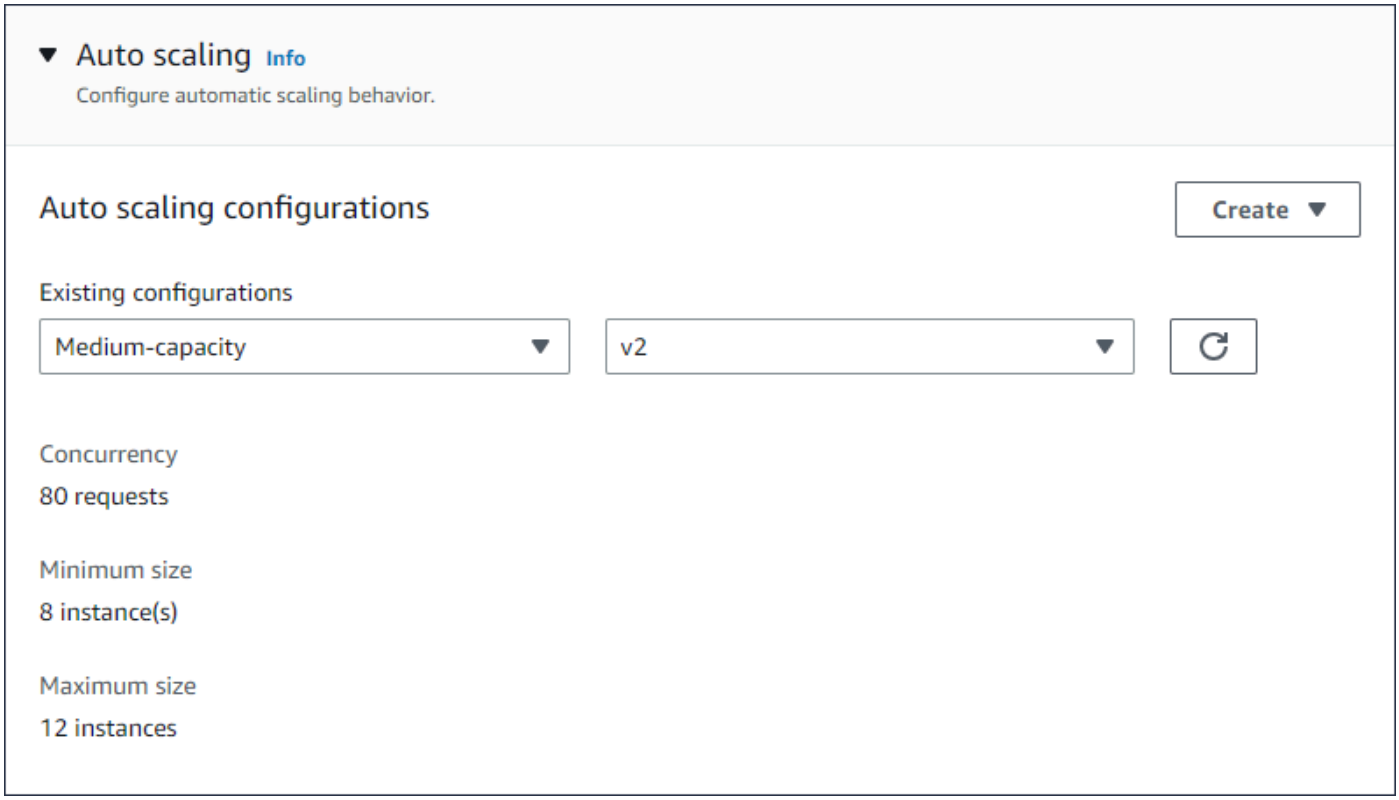
Al crear un servicio mediante la consola de App Runner o al actualizar la configuración de un servicio, puede especificar una configuración de autoescalado.

> **Note**
>
> Cuando cambias la configuración o revisión del escalado automático que está asociada a un servicio, el servicio se vuelve a implementar.

La página de configuración del autoescalado ofrece varias opciones para configurar el autoescalado del servicio.

- Para asignar una configuración y una revisión existentes, elija un valor en el menú desplegable Configuraciones existentes. La última versión de revisión aparecerá por defecto en el menú desplegable adyacente. Si existe una revisión diferente que prefiera seleccionar, hágalo desde el menú desplegable de revisiones. Se muestran los valores de configuración de la versión de revisión.
- Para crear y asignar una nueva configuración de autoescalado, seleccione Crear nuevo ASC en el menú Crear. Esto abre la página Agregar configuración de escalado automático personalizada. Introduzca un nombre y valores de configuración para los parámetros de escalado automático. Luego, elija Añadir. App Runner crea el nuevo recurso de configuración de escalado automático por usted y lo devuelve a la sección de escalado automático con la nueva configuración seleccionada y mostrada.
- Para crear y asignar una nueva revisión, primero seleccione el nombre de la configuración en el menú desplegable Configuraciones existentes. A continuación, seleccione Crear revisión ASC en el menú Crear. Esto abre la página Agregar configuración de escalado automático

personalizada. Introduzca valores para los parámetros de escalado automático. Luego, elija Añadir. App Runner crea una nueva revisión de la configuración de escalado automático para ti y te devuelve a la sección de escalado automático con la nueva revisión seleccionada y mostrada.



App Runner API or AWS CLI

Cuando llamas a las acciones de la API CreateServiceo a UpdateServiceApp Runner, puedes usar el `AutoScalingConfigurationArn` parámetro para especificar un recurso de configuración de autoescalado para tu servicio.

La siguiente sección proporciona instrucciones para administrar los recursos de configuración del escalado automático.

Administre los recursos de configuraciones de escalado automático

Administra las configuraciones y revisiones del escalado automático de App Runner para tu cuenta mediante uno de los siguientes métodos:

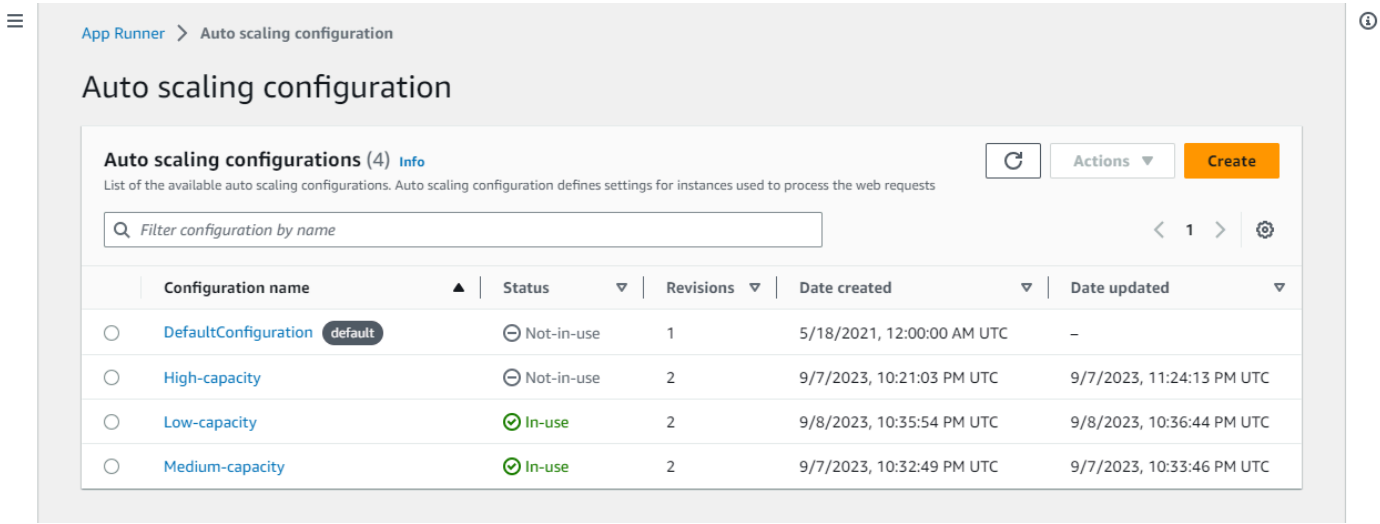
App Runner console

Administre las configuraciones de escalado automático

La página de configuraciones de autoescalado muestra las configuraciones de autoescalado que ha configurado en su cuenta. Puede crear y administrar sus configuraciones de escalado automático en esta página y luego asignarlas a uno o más servicios de App Runner.

Desde esta página, puede realizar cualquiera de las siguientes acciones:

- Cree una nueva configuración de autoescalado.
- Cree una nueva revisión para una configuración de autoescalado existente.
- Elimine una configuración de escalado automático.
- Establezca una configuración de escalado automático como predeterminada.



Para administrar las configuraciones de escalado automático en su cuenta

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Configuraciones de escalado automático. La consola muestra una lista de configuraciones de escalado automático en su cuenta.

   Ahora puede realizar cualquiera de las siguientes acciones.

   - Para crear una nueva configuración de autoescalado, sigue estos pasos.

     a. En la página de configuraciones de escalado automático, seleccione Crear.

Aparece la página Crear configuración de autoescalado.

b. Introduzca los valores para el nombre de la configuración, la simultaneidad, el tamaño mínimo y el tamaño máximo.

c. (Opcional) Si quieres añadir etiquetas, selecciona Nueva etiqueta automática. A continuación, en los campos que aparecen, introduce un nombre y un valor (opcional).

d. Seleccione Crear.

- Para crear una nueva revisión para una configuración de autoescalado existente, siga estos pasos.

  a. En la página de configuraciones de escalado automático, seleccione el botón de radio situado junto a la configuración que necesita la nueva revisión. A continuación, seleccione Crear revisión en el menú Acciones.

     Aparece la página Crear revisión.

  b. Activa, introduzca los valores de simultaneidad, tamaño mínimo y tamaño máximo.

  c. (Opcional) Si quieres añadir etiquetas, selecciona Nueva etiqueta automática. A continuación, en los campos que aparecen, introduce un nombre y un valor (opcional).

  d. Seleccione Crear.

- Para eliminar una configuración de escalado automático, sigue estos pasos.

  a. En la página de configuraciones de escalado automático, seleccione el botón de radio situado junto a la configuración que desee eliminar.

  b. Seleccione Eliminar en el menú Acciones.

  c. Para continuar con la eliminación, seleccione Eliminar en el cuadro de diálogo de confirmación. De lo contrario, selecciona Cancelar.

  > **Note**
  >
  > App Runner valida que la opción de eliminación no esté establecida como predeterminada o que esté siendo utilizada actualmente por algún servicio activo.

- Para establecer una configuración de escalado automático como predeterminada, sigue estos pasos.

  a. En la página de configuraciones de escalado automático, seleccione el botón de radio situado junto a la configuración que desee establecer como predeterminada.

  b. Seleccione Establecer como predeterminado en el menú Acciones.

  c. Aparece un cuadro de diálogo en el que se le informa de que App Runner utilizará la última revisión como configuración predeterminada para todos los nuevos servicios que cree. Selecciona Confirmar para continuar. De lo contrario, selecciona Cancelar.

  > **Note**
  >
  > - Cuando establece una configuración de escalado automático como predeterminada, se asigna automáticamente como configuración predeterminada a los nuevos servicios que cree en el futuro.
  > - La nueva designación predeterminada no afecta a las asociaciones que se establecieron anteriormente para los servicios existentes.
  > - Si la configuración de escalado automático predeterminada designada tiene revisiones, App Runner asigna su última revisión como predeterminada.

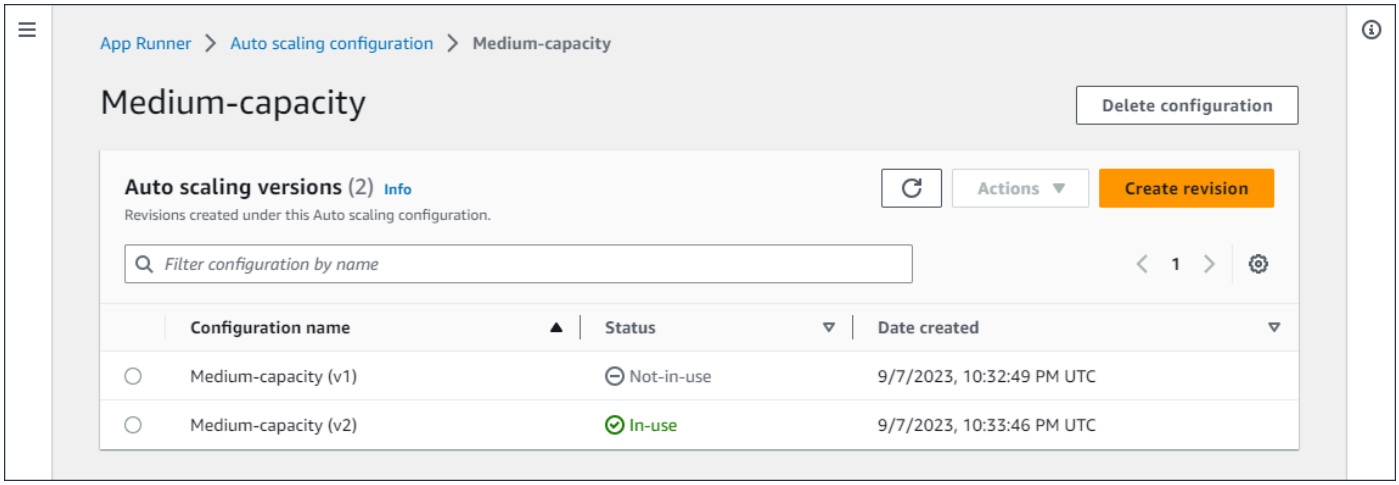
Administra las revisiones

La consola también tiene una página para crear y administrar las revisiones de escalado automático existentes, denominada revisiones de autoescalado. Para acceder a esta página, seleccione el nombre de una configuración en la página de configuraciones de escalado automático.

Puede realizar cualquiera de las siguientes acciones desde la página de revisiones del escalado automático:

- Cree una nueva revisión de escalado automático.
- Establezca una revisión de la configuración de escalado automático como predeterminada.
- Elimine una revisión.

- Elimine toda la configuración de autoescalado, incluidas todas las revisiones asociadas.
- Vea los detalles de configuración de una revisión.
- Vea una lista de los servicios asociados a una revisión.
- Cambie la revisión de un servicio de la lista.



Para administrar las revisiones de escalado automático en tu cuenta

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Configuraciones de escalado automático. La consola muestra una lista de configuraciones de escalado automático en su cuenta. El conjunto de procedimientos anterior de la ??? sección incluye una imagen de pantalla de esta página.
3. Ahora puede profundizar en una configuración de escalado automático específica para ver y administrar todas sus revisiones. En el panel de configuraciones de autoescalado, en la columna Nombre de configuración, elija un nombre de configuración de autoescalado. Seleccione el nombre real en lugar del botón de radio. Esto le llevará a una lista de todas las revisiones de esa configuración en la página de revisiones del escalado automático.
4. Ahora puede realizar cualquiera de las siguientes acciones.
   - Para crear una nueva revisión para una configuración de autoescalado existente, siga estos pasos.

     a. En la página de revisiones de escalado automático, seleccione Crear revisión.

        Aparece la página Crear revisión.

     b. Introduzca los valores de simultaneidad, tamaño mínimo y tamaño máximo.

c. (Opcional) Si quieres añadir etiquetas, selecciona Nueva etiqueta automática. A continuación, en los campos que aparecen, introduce un nombre y un valor (opcional).

d. Seleccione Crear.

- Para eliminar toda la configuración de autoescalado, incluidas todas las revisiones asociadas, sigue estos pasos.

  a. Seleccione Eliminar configuración en la parte superior derecha de la página.

  b. Para continuar con la eliminación, selecciona Eliminar en el cuadro de diálogo de confirmación. De lo contrario, selecciona Cancelar.

  > **Note**
  >
  > App Runner valida que la opción de eliminación no esté establecida como predeterminada o que esté siendo utilizada actualmente por algún servicio activo.

- Para establecer una revisión de escalado automático como predeterminada, sigue estos pasos.

  a. Seleccione el botón de radio situado junto a la revisión que desee establecer como predeterminada.

  b. Seleccione Establecer como predeterminado en el menú Acciones.

  > **Note**
  >
  > - Cuando establece una configuración de escalado automático como predeterminada, se asigna automáticamente como configuración predeterminada a los nuevos servicios que cree en el futuro.
  > - La nueva designación predeterminada no afecta a las asociaciones que se establecieron anteriormente para los servicios existentes.

- Para ver los detalles de configuración de una revisión, sigue estos pasos.

  - Seleccione el botón de radio situado junto a la revisión.

Los detalles de configuración de la revisión, incluido el ARN, se muestran en el panel de división inferior. Consulte la imagen de la pantalla al final de este procedimiento.

- Para ver una lista de los servicios asociados a una revisión, siga estos pasos.
  - Seleccione el botón de radio situado junto a la revisión.

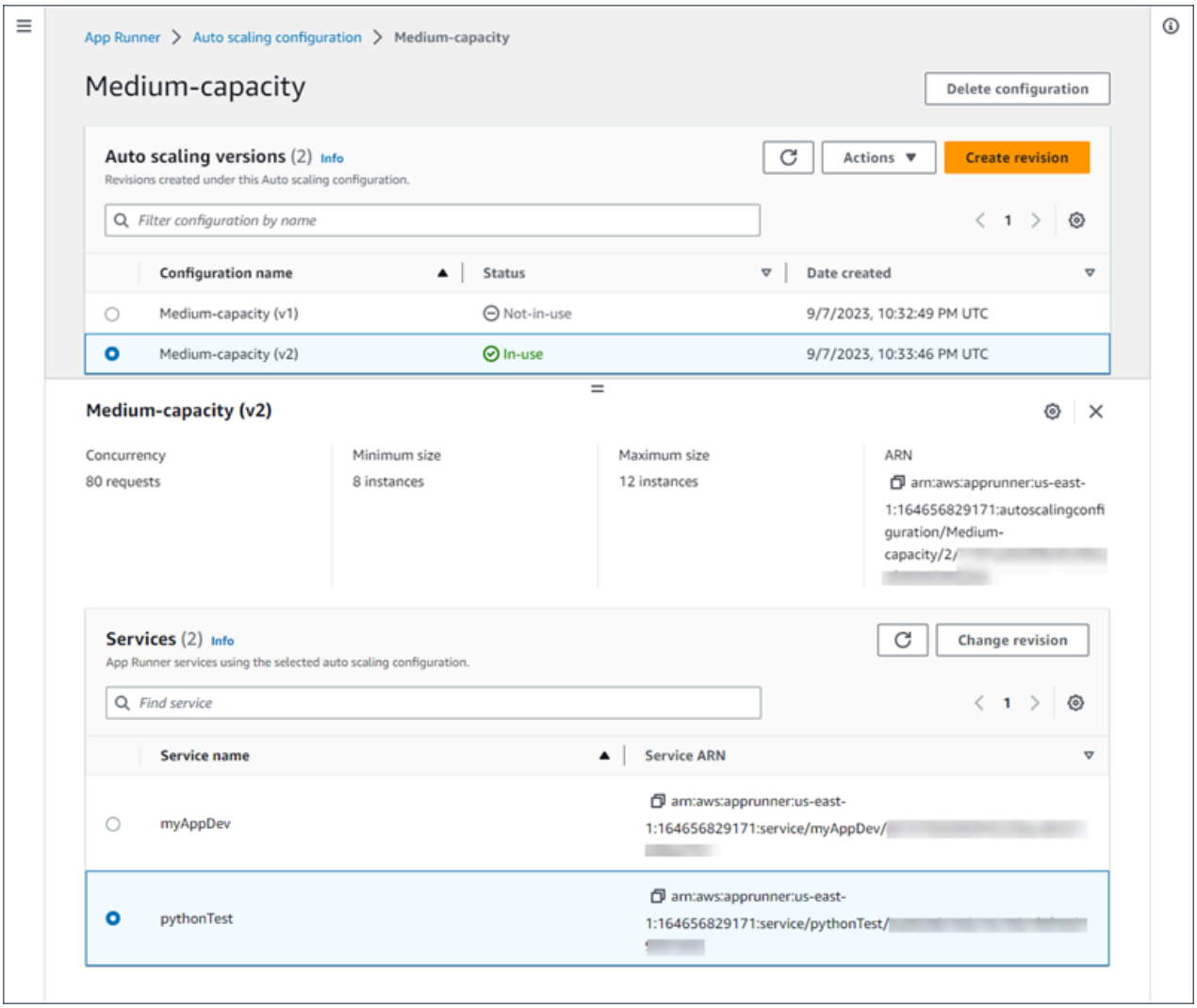
    El panel de servicios se muestra en el panel dividido inferior, debajo de los detalles de configuración de la revisión. El panel muestra todos los servicios que utilizan esta revisión de configuración de autoescalado. Consulte la imagen de la pantalla al final de este procedimiento.
- Para cambiar la revisión de un servicio de la lista, siga estos pasos.
  a. Selecciona el botón de radio situado junto a la revisión, si aún no lo has hecho.

     El panel de servicios se muestra en el panel dividido inferior, debajo de los detalles de configuración de la revisión. El panel muestra todos los servicios que utilizan esta revisión de configuración de autoescalado. Consulte la imagen de la pantalla al final de este procedimiento.
  b. En el panel de servicios, seleccione el botón de radio situado junto al servicio que desee modificar. A continuación, selecciona Cambiar revisiones.
  c. Aparece el panel Cambiar revisión de ASC. Elija una de las revisiones disponibles en el menú desplegable. Solo están disponibles las revisiones de la configuración de autoescalado que eligió anteriormente. Si necesita cambiar a una configuración de autoescalado diferente, siga los procedimientos de la sección anteriorthe section called “Administre el escalado automático de un servicio”.

     Seleccione Actualizar para continuar con el cambio. De lo contrario, selecciona Cancelar.

     > **Note**
     >
     > Cuando cambias una revisión que está asociada a un servicio, el servicio se vuelve a implementar.
     > Debe seleccionar actualizar en este panel para ver las asociaciones actualizadas.

Para ver la actividad en curso y el estado de la redistribución del servicio, utilice las rutas de navegación del panel para ir a App Runner > Servicios, seleccione el servicio y, a continuación, consulte la pestaña Registros en el panel de información general del servicio.



App Runner API or AWS CLI

Usa las siguientes acciones de la API de App Runner para administrar tus recursos de configuración de autoescalado.

- CreateAutoScalingConfiguration— Crea una nueva configuración de autoescalado o una revisión de una existente.

- UpdateDefaultAutoScalingConfiguration—Establece una configuración de escalado automático como predeterminada. La configuración de escalado automático predeterminada existente se establecerá automáticamente como no predeterminada.
- ListAutoScalingConfigurations— Devuelve una lista de las configuraciones de escalado automático asociadas a usted Cuenta de AWS, con información resumida.
- ListServicesForAutoScalingConfiguration— Devuelve una lista de los servicios de App Runner asociados mediante una configuración de escalado automático.
- DescribeAutoScalingConfiguration— Devuelve una descripción completa de una configuración de autoescalado.
- DeleteAutoScalingConfiguration— Elimina una configuración de autoescalado. Puede eliminar una configuración de autoescalado de nivel superior, una revisión específica de una o todas las revisiones asociadas a la configuración de nivel superior. Utilice el `DeleteAllRevisions` parámetro opcional para eliminar todas las revisiones. Si alcanza la cuota de recursos de configuración de autoescalado de su cuenta Cuenta de AWS, es posible que tenga que eliminar las configuraciones de autoescalado innecesarias.

Administrar nombres de dominio personalizados para un servicio de App Runner

Al crear un AWS App Runner servicio, App Runner le asigna un nombre de dominio. Se trata de un subdominio del `awsapprunner.com` dominio que es propiedad de App Runner. Puedes usar el nombre de dominio para acceder a la aplicación web que se ejecuta en tu servicio.

> **Note**
>
> Para aumentar la seguridad de las aplicaciones de App Runner, el dominio*.awsapprunner.com está registrado en la lista pública de sufijos (PSL). Para mayor seguridad, te recomendamos que utilices cookies con un `__Host-` prefijo si alguna vez necesitas configurar cookies confidenciales en el nombre de dominio predeterminado de tus aplicaciones de App Runner. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulta la Set-Cookiepágina de la Red de desarrolladores de Mozilla.

Si tienes un nombre de dominio, puedes asociarlo a tu servicio App Runner. Una vez que App Runner valide tu nuevo dominio, puedes usar tu dominio para acceder a tu aplicación además del dominio de App Runner. Puedes asociar hasta cinco dominios personalizados.

> **Note**
>
> Si lo desea, puede incluir el `www` subdominio de su dominio. Sin embargo, actualmente solo lo admite la API. La consola de App Runner no admite la inclusión de un `www` subdominio de tu dominio.

> **Note**
>
> AWS App Runner no admite el uso de zonas alojadas privadas de Route 53. Las zonas alojadas privadas personalizan la resolución de nombres de dominio para el tráfico de Amazon VPC. Para obtener más información sobre las zonas alojadas privadas, consulte Trabajar con zonas alojadas privadas en la documentación de Route 53.

Asocie (vincule) un dominio personalizado a su servicio

Cuando asocias un dominio personalizado a tu servicio, debes añadir los registros CNAME y los registros de destino del DNS a tu servidor DNS. En las siguientes secciones, se proporciona información sobre los registros CNAME y los registros de destino de DNS y sobre cómo utilizarlos.

> **Note**
>
> Si utilizas Amazon Route 53 como proveedor de DNS, App Runner configura automáticamente tu dominio personalizado con la validación de certificados y los registros de DNS necesarios para vincularlos a tu aplicación web de App Runner. Esto sucede cuando utilizas la consola de App Runner para vincular tu dominio personalizado a tu servicio. En el Administra dominios personalizados tema siguiente se proporciona más información.

Registros CNAME

Cuando asocias un dominio personalizado a tu servicio, App Runner te proporciona un conjunto de registros de validación de certificados para la validación de los certificados. Debe agregar estos

registros de validación de certificados a su servidor del Sistema de nombres de dominio (DNS). Agregue los registros de validación de certificados, proporcionados por App Runner, a su servidor DNS. De esta forma, App Runner puede validar que eres propietario o controlas el dominio.

> **Note**
>
> Para renovar automáticamente sus certificados de dominio personalizados, asegúrese de no eliminar los registros de validación de certificados de su servidor DNS. Para obtener información sobre cómo resolver los problemas relacionados con la renovación del certificado, consultethe section called “Renovación de certificados de dominio personalizados”.

App Runner usa ACM para verificar el dominio. Si utilizas registros CAA en tus registros DNS, asegúrate de que al menos un registro CAA haga referencia`amazon.com`. De lo contrario, ACM no podrá verificar el dominio y crearlo correctamente.

Si recibes errores relacionados con la CAA, consulta los siguientes enlaces para obtener información sobre cómo resolverlos:

- Problemas con la autorización de la autoridad de certificación (CAA)
- ¿Cómo resuelvo los errores de la CAA al emitir o renovar un certificado ACM?
- Nombres de dominio personalizados

> **Note**
>
> Si utilizas Amazon Route 53 como proveedor de DNS, App Runner configura automáticamente tu dominio personalizado con la validación de certificados y los registros de DNS necesarios para vincularlos a tu aplicación web de App Runner. Esto sucede cuando utilizas la consola de App Runner para vincular tu dominio personalizado a tu servicio. En el Administra dominios personalizados tema siguiente se proporciona más información.

Registros de destino de DNS

Agrega los registros de destino DNS a tu servidor DNS para dirigirlos al dominio de App Runner. Agrega un registro para el dominio personalizado y otro para el `www` subdominio, si eliges esta opción. Luego, espera a que el estado del dominio personalizado pase a ser Activo en la consola

de App Runner. Esto suele tardar varios minutos, pero puede tardar entre 24 y 48 horas (entre 1 y 2 días). Cuando se valida tu dominio personalizado, App Runner comienza a redirigir el tráfico de este dominio a tu aplicación web.

> **Note**
>
> Para una mejor compatibilidad con los servicios de App Runner, le recomendamos que utilice Amazon Route 53 como proveedor de DNS. Si no utiliza Amazon Route 53 para administrar sus registros de DNS públicos, póngase en contacto con su proveedor de DNS para obtener información sobre cómo añadir registros.
> Si utiliza Amazon Route 53 como proveedor de DNS, puede añadir un registro CNAME o un registro de alias para el subdominio. Para el dominio raíz, asegúrese de utilizar el registro de alias.

Puedes comprar un nombre de dominio en Amazon Route 53 o en otro proveedor. Para comprar un nombre de dominio con Amazon Route 53, consulte Registrar un nuevo dominio en la Guía para desarrolladores de Amazon Route 53.

Para obtener instrucciones sobre cómo configurar un destino de DNS en Route 53, consulte Enrutamiento del tráfico a sus recursos, en la Guía para desarrolladores de Amazon Route 53.

Para obtener instrucciones sobre cómo configurar un destino de DNS en otros registradores, como Shopify GoDaddy, Hover, etc., consulte su documentación específica sobre cómo agregar registros de DNS Target.

Especifica un dominio para asociarlo a tu servicio de App Runner

Puede especificar un dominio para asociarlo a su servicio de App Runner de las siguientes maneras:

- Un dominio raíz: el DNS tiene algunas limitaciones inherentes que pueden impedirle crear registros CNAME para el nombre de dominio raíz. Por ejemplo, si tu nombre de dominio lo es`example.com`, puedes crear un registro CNAME que dirija el tráfico `acme.example.com` a tu servicio de App Runner. Sin embargo, no puedes crear un registro CNAME que dirija el tráfico `example.com` a tu servicio de App Runner. Para crear un dominio raíz, asegúrate de añadir un registro de alias.

  Un registro de alias es específico de Route 53 y tiene las siguientes ventajas en comparación con los registros CNAME:

  - Route 53 le proporciona más flexibilidad, ya que se pueden crear registros de alias para el dominio raíz o el subdominio. Por ejemplo, si tu nombre de dominio es`example.com`, puedes crear un registro que dirija las solicitudes hacia `example.com` o `acme.example.com` hacia tu servicio de App Runner.
  - Es más rentable. Esto se debe a que Route 53 no cobra por las solicitudes que utilizan un registro de alias para enrutar el tráfico.
- Un subdominio, por ejemplo, `login.example.com` o`admin.login.example.com`. Si lo desea, también puede asociar el `www` subdominio como parte de la misma operación. Puedes añadir un registro CNAME o un registro de alias para el subdominio.
- Un comodín: por ejemplo,. `*.example.com` No puede usar la `www` opción en este caso. Puede especificar un comodín solo como subdominio inmediato de un dominio raíz y solo como tal. Estas especificaciones no son válidas:`login*.example.com`,. `*.login.example.com` Esta especificación comodín asocia todos los subdominios inmediatos y no asocia el dominio raíz en sí. El dominio raíz debe estar asociado en una operación independiente.

Una asociación de dominio más específica anula una menos específica. Por ejemplo, `login.example.com` `*.example.com` anulaciones. Se utilizan el certificado y el CNAME de la asociación más específica.

En el siguiente ejemplo, se muestra cómo se pueden utilizar varias asociaciones de dominios personalizadas:

1. `example.com`Asócielo a la página de inicio de su servicio. Active la opción `www` para asociarse`www.example.com`.
2. `login.example.com`Asócielo a la página de inicio de sesión de su servicio.
3. Asocie `*.example.com` a una página personalizada «no encontrada».

Desasociar (desvincular) un dominio personalizado

Puedes desasociar (desvincular) un dominio personalizado de tu servicio App Runner. Cuando desvinculas un dominio, App Runner deja de enrutar el tráfico de este dominio a tu aplicación web.

> **Note**
>
> Debes eliminar los registros del dominio que desasociaste de tu servidor DNS.

App Runner crea internamente certificados que rastrean la validez del dominio. Estos certificados se almacenan en AWS Certificate Manager (ACM). App Runner no elimina estos certificados durante 7 días después de que un dominio se desvincule de su servicio o después de que se elimine el servicio.

Administra dominios personalizados

Administre los dominios personalizados para su servicio de App Runner mediante uno de los siguientes métodos:

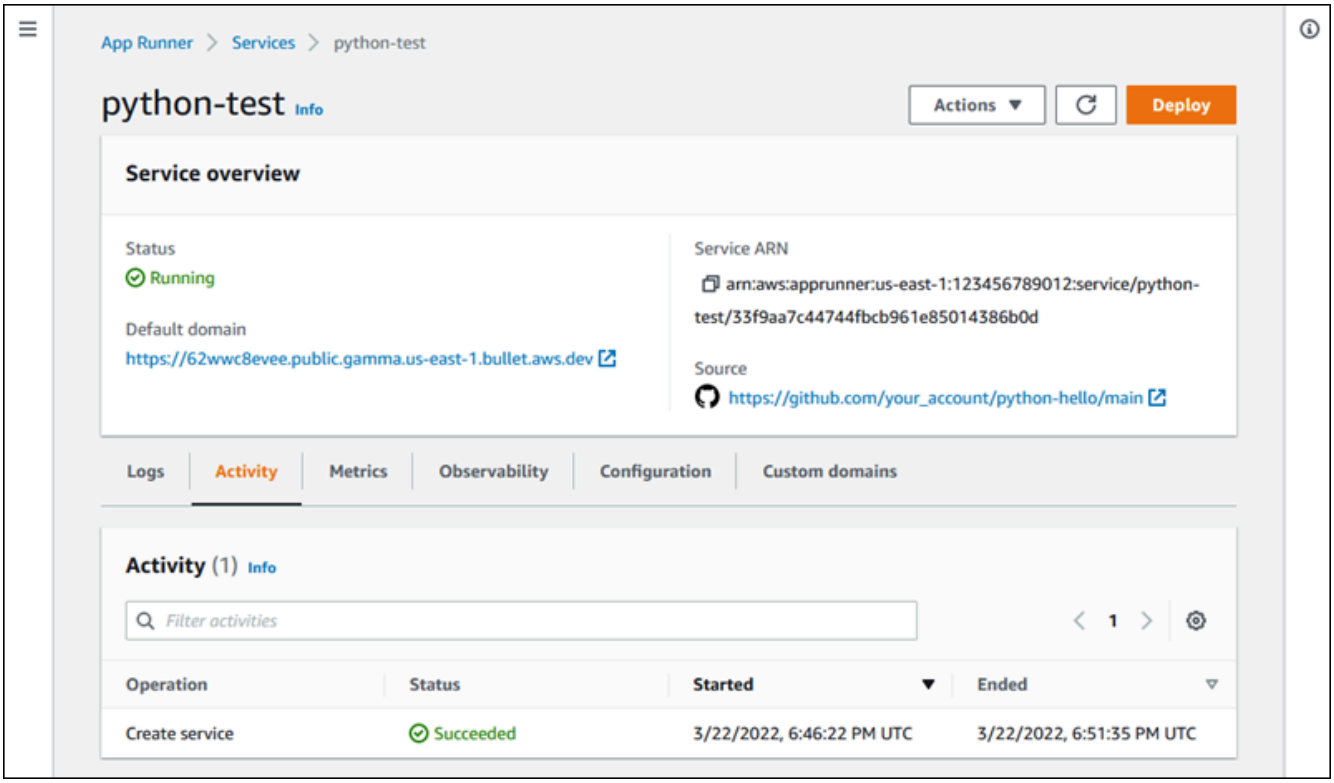
> **Note**
>
> Para una mejor compatibilidad con los servicios de App Runner, le recomendamos que utilice Amazon Route 53 como proveedor de DNS. Si no utiliza Amazon Route 53 para administrar sus registros de DNS públicos, póngase en contacto con su proveedor de DNS para obtener información sobre cómo añadir registros.
> Si utiliza Amazon Route 53 como proveedor de DNS, puede añadir un registro CNAME o un registro de alias para el subdominio. Para el dominio raíz, asegúrese de utilizar el registro de alias.

App Runner console

Para asociar (vincular) un dominio personalizado mediante la consola de App Runner

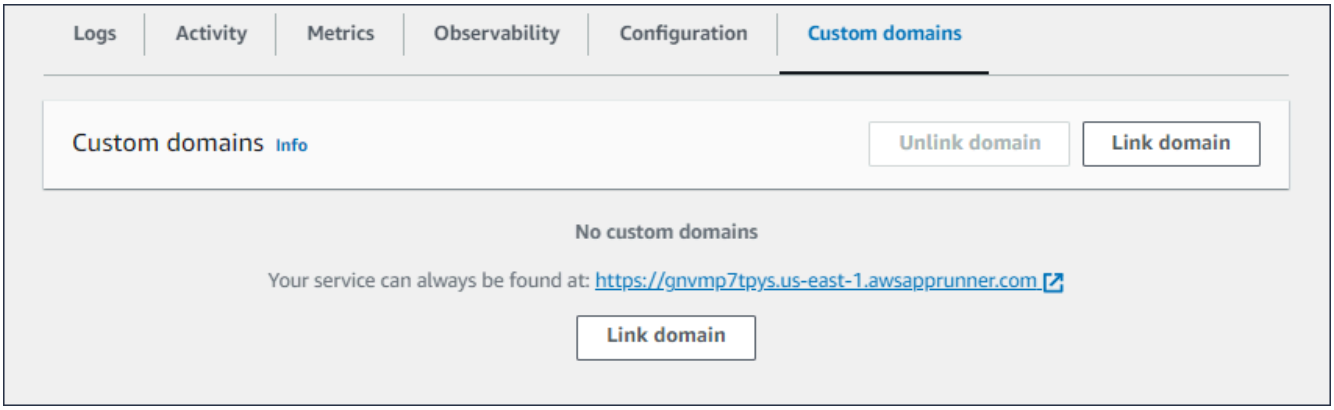
1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

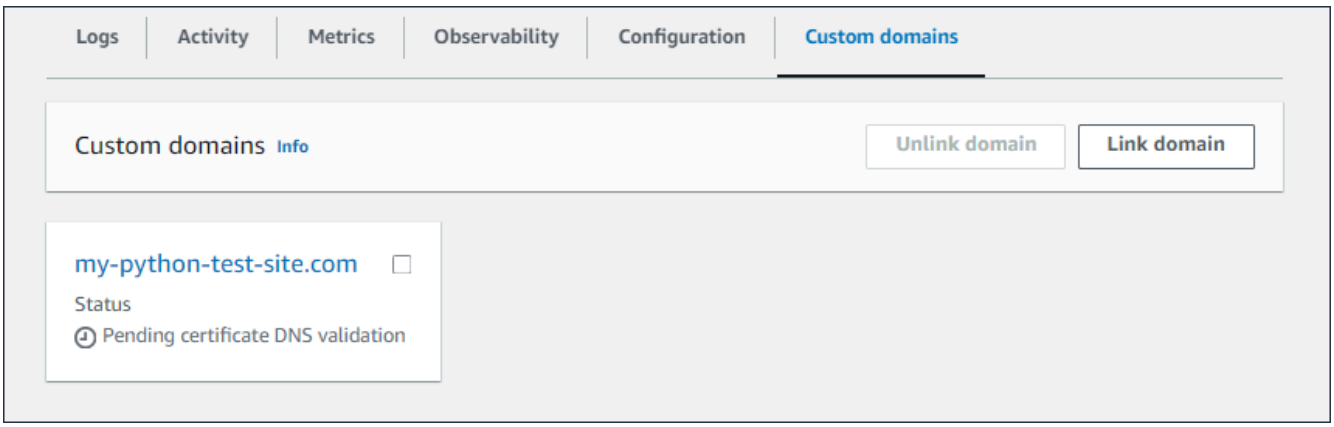
   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, selecciona la pestaña Dominios personalizados.

   La consola muestra los dominios personalizados que están asociados a su servicio o no hay dominios personalizados.

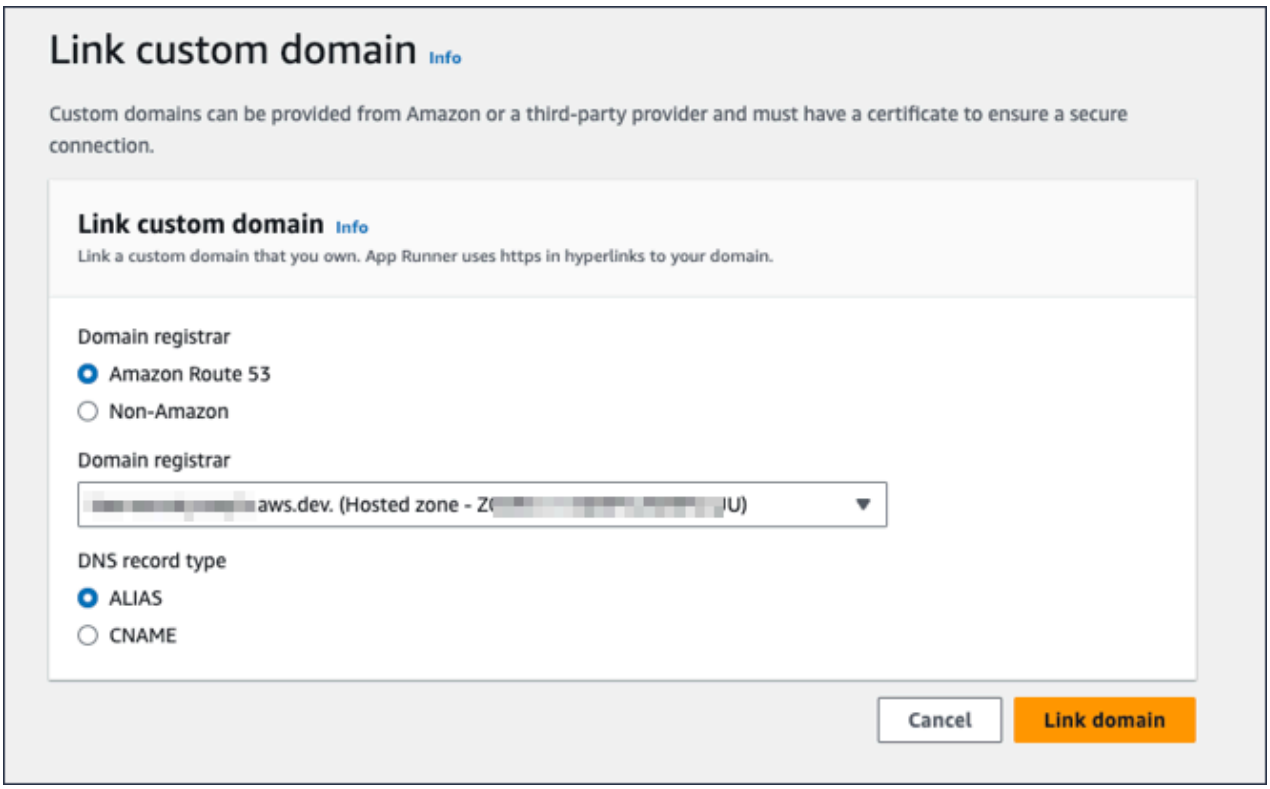




4. En la pestaña Dominios personalizados, selecciona Vincular dominio.
5. Aparece la página Vincular un dominio personalizado.
   - Si su dominio personalizado está registrado en Amazon Route 53, seleccione Amazon Route 53 como registrador de dominios.
     a. Seleccione el nombre de dominio en la lista desplegable. Esta lista muestra el nombre de sus nombres de dominio de Route 53 y el identificador de la zona alojada.

     > **Note**
     >
     > Primero debes crear un dominio de Route 53 mediante el servicio Amazon Route 53 desde la misma AWS cuenta que utilizas para administrar los demás recursos de App Runner.

     b. Seleccione el tipo de registro DNS.
     c. Elige Vincular dominio.

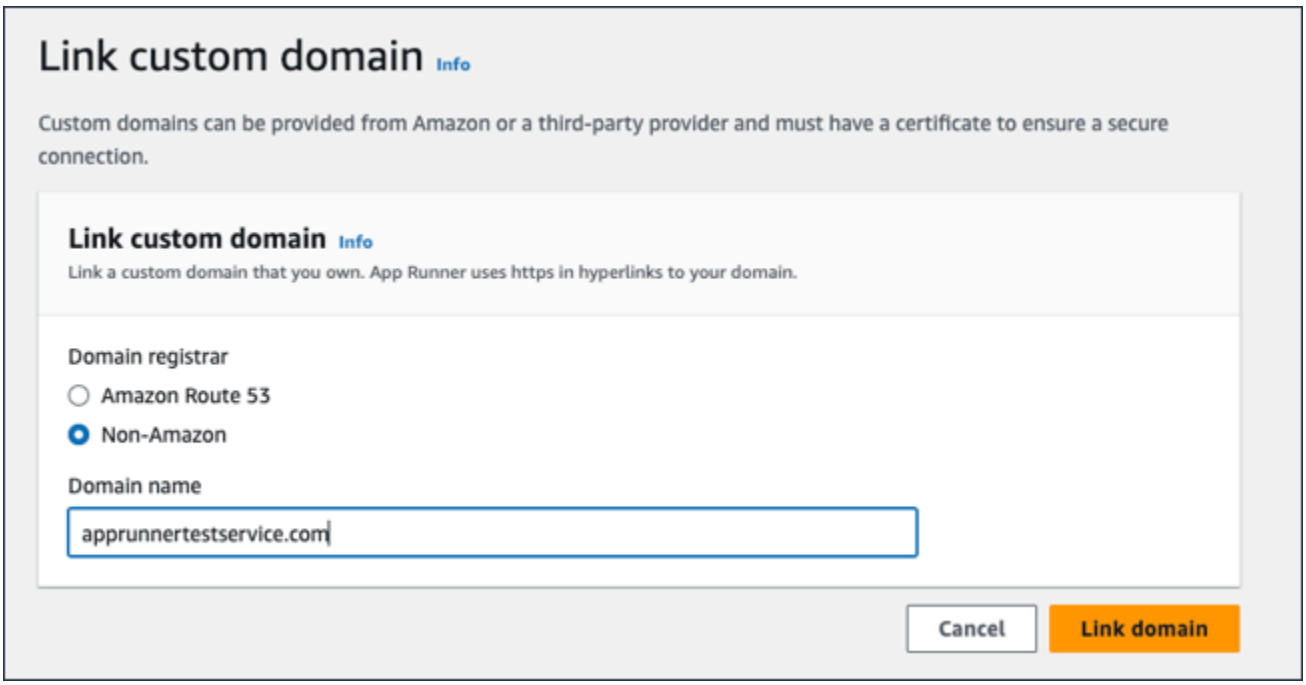


> **Note**
>
> Si App Runner muestra un mensaje de error que indica que el intento de configuración automática ha fallado, puede continuar configurando los registros DNS manualmente. Este problema puede surgir si el mismo nombre de dominio se desvinculó anteriormente de un servicio, sin que los registros del proveedor de DNS que apuntan al servicio se eliminen posteriormente. En este caso, App Runner no podrá sobrescribir automáticamente estos registros. Para finalizar la configuración del DNS, omita el resto de los pasos de este procedimiento y, a continuación, siga las instrucciones que se indican. Configurar un registro de alias de Amazon Route 53

- Si tu dominio personalizado está registrado en otro registrador de dominios, selecciona No Amazon como registrador de dominios.

  a. Introduce el nombre del dominio.

  b. Selecciona Vincular el dominio.



6. Aparece la página Configurar DNS.

   - Si Amazon Route 53 es su proveedor de DNS, este paso es opcional.

     En este punto, App Runner ha configurado automáticamente su dominio de Route 53 con la validación de certificados y los registros de DNS necesarios.

     > **Note**
     >
     > Si este mismo nombre de dominio estaba previamente desvinculado de un servicio, sin que posteriormente se eliminaran los registros del proveedor de DNS que apuntan al servicio, la configuración automática que intentó realizar App Runner podría haber fallado. Para solucionar este problema y completar la asociación de DNS, continúe con los pasos (1) y (2) de la página Configurar el DNS para copiar los registros de destino y de certificado actuales al proveedor de DNS.

   - Copie los registros de validación de certificados y los registros de destino de DNS y agréguelos a su servidor DNS. Luego, App Runner puede validar que eres el propietario del dominio o lo controlas.

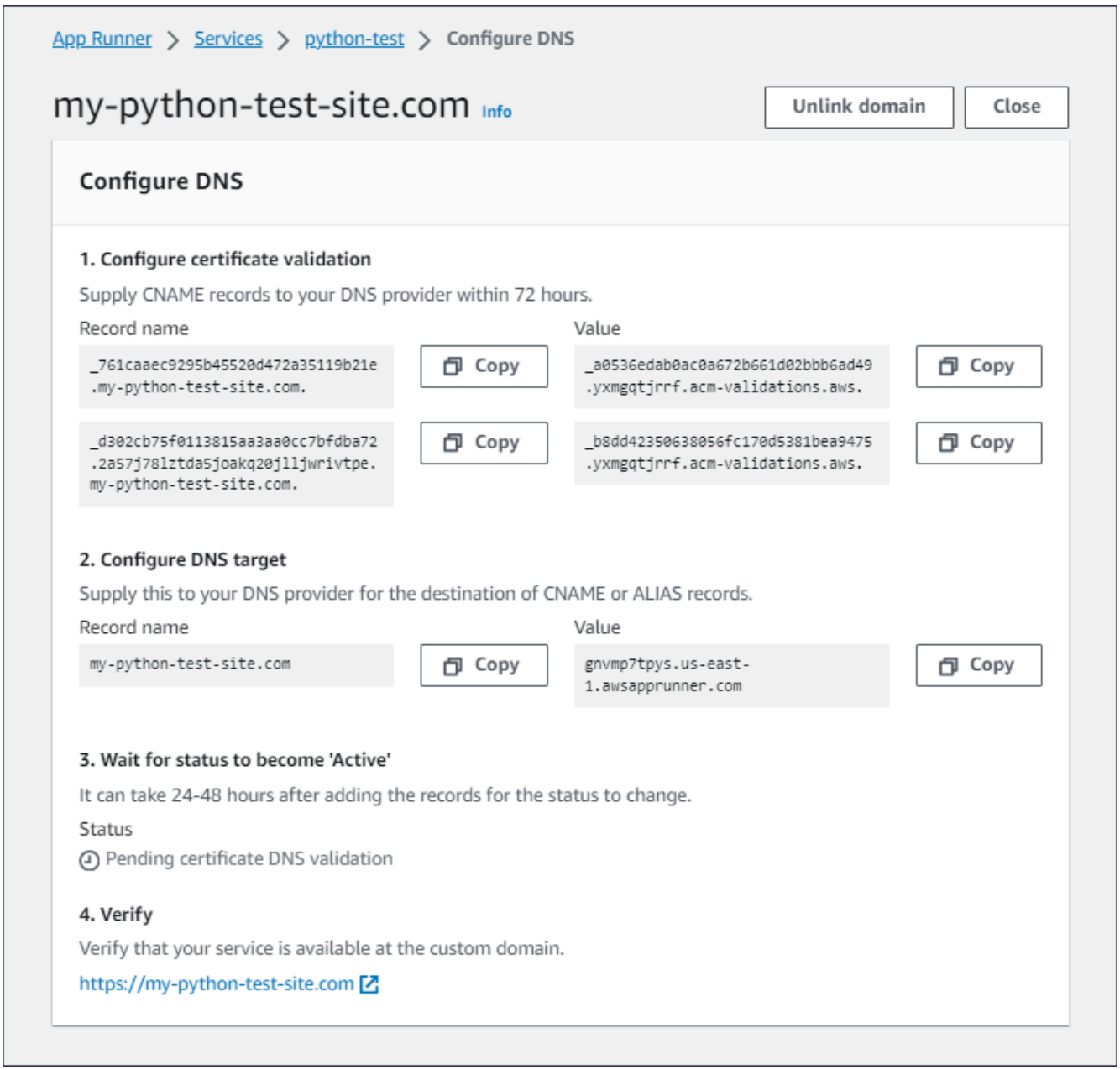
> **Note**
>
> Para renovar automáticamente sus certificados de dominio personalizados, asegúrese de no eliminar los registros de validación de certificados de su servidor DNS.

- Para obtener más información sobre la configuración de la validación de certificados, consulte la Validación de DNS en la Guía del AWS Certificate Manager usuario.
- Para obtener información sobre cómo configurar el destino de DNS con el registro de alias de Amazon Route 53, consultethe section called “Configurar un registro de alias de Amazon Route 53”.

- Si utilizas un proveedor de DNS que no sea Amazon Route 53, sigue estos pasos.
  - Copie los registros de validación de certificados y los registros de destino de DNS y agréguelos a su servidor DNS. Luego, App Runner puede validar que eres el propietario del dominio o lo controlas.

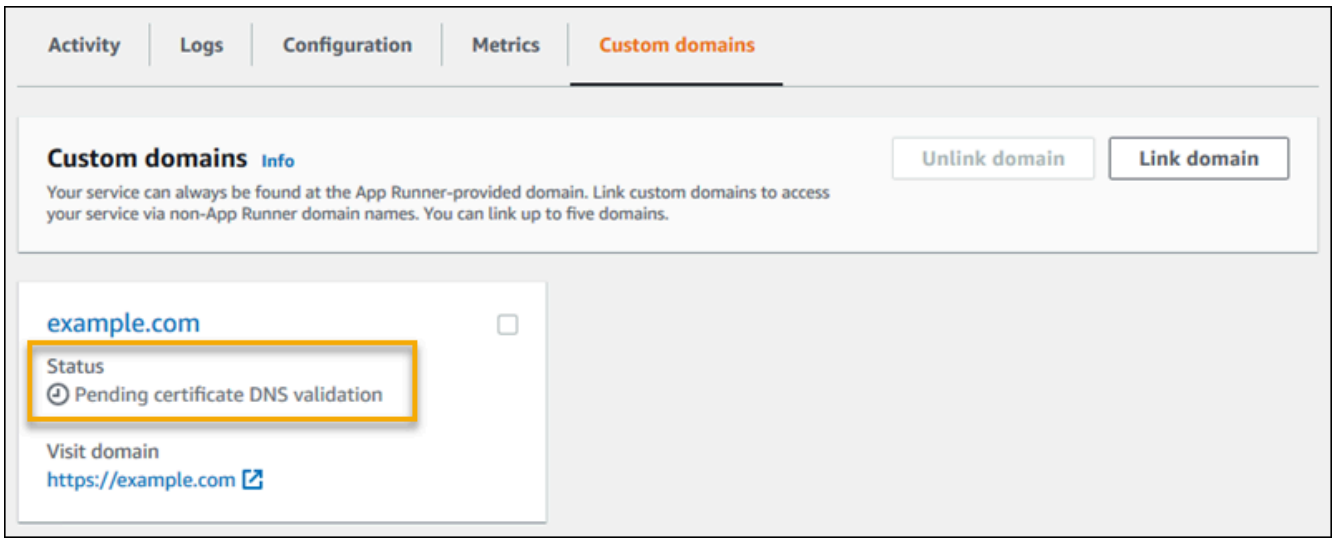
> **Note**
>
> Para renovar automáticamente sus certificados de dominio personalizados, asegúrese de no eliminar los registros de validación de certificados de su servidor DNS.

- Para obtener más información sobre la configuración de la validación de certificados, consulte la Validación de DNS en la Guía del AWS Certificate Manager usuario.
- Para obtener instrucciones sobre cómo configurar un destino de DNS en otros registradores, como Shopify GoDaddy, Hover, etc., consulta su documentación específica sobre cómo añadir un destino de DNS.

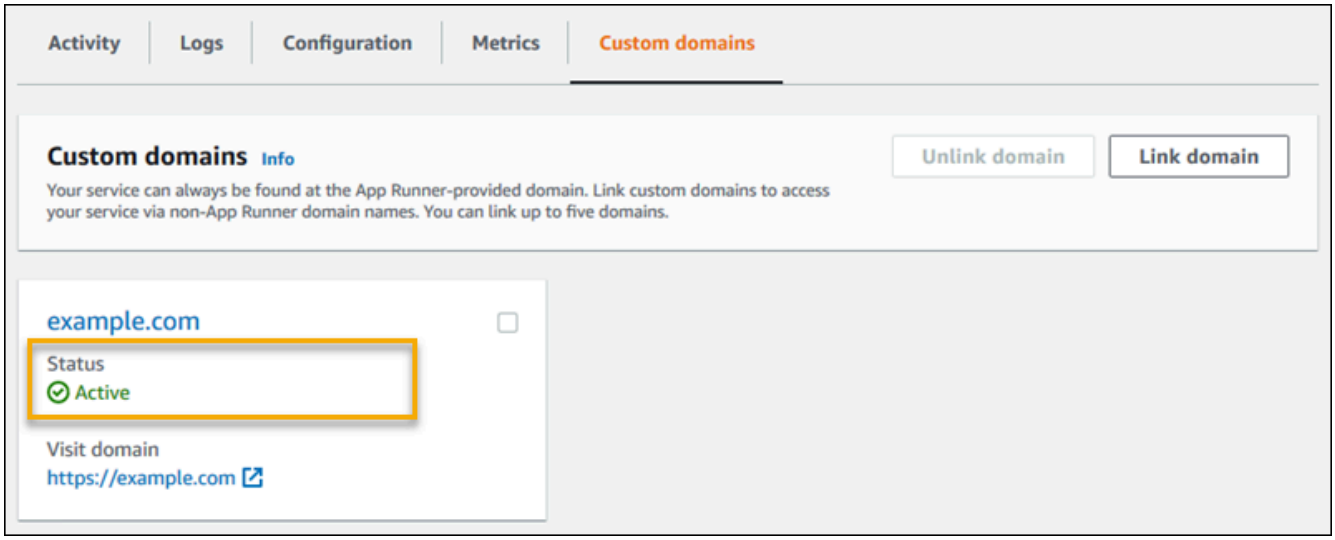


7. Selecciona Cerrar

La consola vuelve a mostrar el panel de control. La pestaña Dominios personalizados tiene un nuevo mosaico que muestra el dominio que acabas de vincular con el estado de validación de DNS del certificado pendiente.



8. Cuando el estado del dominio cambie a Activo, navegue hasta él para comprobar que el dominio funciona para enrutar el tráfico.



> **Note**
>
> Para obtener instrucciones sobre cómo solucionar los errores relacionados con el dominio personalizado, consultethe section called “Nombres de dominio personalizados”.

Para desasociar (desvincular) un dominio personalizado mediante la consola de App Runner

1. En la pestaña Dominios personalizados, selecciona el mosaico del dominio que quieres desasociar y, a continuación, selecciona Desvincular el dominio.

2. En el cuadro de diálogo Desvincular el dominio, verifica la acción seleccionando Desvincular el dominio.

> **Note**
>
> Debes eliminar los registros del dominio que desasociaste de tu servidor DNS.

App Runner API or AWS CLI

Para asociar un dominio personalizado a tu servicio mediante la API de App Runner AWS CLI, llama a la acción de la AssociateCustomDomainAPI. Cuando la llamada se realiza correctamente, se devuelve un CustomDomainobjeto que describe el dominio personalizado que se está asociando a tu servicio. El objeto muestra un `CREATING` estado y contiene una lista de CertificateValidationRecordobjetos. La llamada también devuelve el alias de destino que puede usar para configurar el destino DNS. Se trata de registros que puede añadir a su DNS.

Para desasociar un dominio personalizado de tu servicio mediante la API de App Runner AWS CLI, llama a la acción de la DisassociateCustomDomainAPI. Cuando la llamada se realiza correctamente, se devuelve un CustomDomainobjeto que describe el dominio personalizado que se va a disociar de tu servicio. El objeto muestra un `DELETING` estado.

Temas

- Configure el registro de alias de Amazon Route 53 para su DNS de destino

Configure el registro de alias de Amazon Route 53 para su DNS de destino

> **Note**
>
> No es necesario que siga este procedimiento si Amazon Route 53 es su proveedor de DNS. En este caso, App Runner configura automáticamente su dominio de Route 53 con la validación de certificados y los registros DNS necesarios para vincularlo a su aplicación web de App Runner.
> Si el intento de configuración automática de App Runner falló, siga este procedimiento para completar la configuración de DNS. Si el mismo nombre de dominio se desvinculó anteriormente de un servicio, sin que posteriormente se eliminen los registros del proveedor de DNS que apuntan al servicio, App Runner no podrá sobrescribir automáticamente estos

registros. En este procedimiento se explica cómo copiarlos manualmente al DNS de Route 53.

Puede usar Amazon Route 53 como proveedor de DNS para enrutar el tráfico a su servicio App Runner. Es un servicio web de sistema de nombres de dominio (DNS) escalable y de alta disponibilidad. El registro Amazon Route 53 contiene la configuración que controla cómo se enruta el tráfico a tu servicio App Runner. Puede crear un registro CNAME o un registro ALIAS. Para ver una comparación entre los registros CNAME y los de alias, consulte Elegir entre registros con alias y sin alias, en la Guía para desarrolladores de Amazon Route 53.

> **Note**
>
> Amazon Route 53 admite actualmente el registro de alias para los servicios que se creen después del 1 de agosto de 2022.

Amazon Route 53 console

Para configurar el registro de alias de Amazon Route 53

1. Inicie sesión en la consola de Route 53 Consola de administración de AWS y ábrala.
2. En el panel de navegación, elija Zonas alojadas.
3. Elija el nombre de la zona alojada que quiere usar para enrutar el tráfico a su servicio App Runner.
4. Elija Crear registro.
5. Especifique los siguientes valores:

   - Política de enrutamiento: elija la política de enrutamiento aplicable. Para obtener más información, consulte Elegir una política de enrutamiento.
   - Nombre de registro: ingresa el nombre de dominio que deseas usar para enrutar el tráfico a tu servicio App Runner. El valor predeterminado es el nombre de la zona alojada. Por ejemplo, si el nombre de la zona alojada es `example.com` y desea utilizarla para `acme.example.com` enrutar el tráfico a su entorno, introduzca`acme`.
   - Value/Route tráfico a: seleccione un alias para la aplicación App Runner y, a continuación, elija la región de la que proviene el punto final. Elija el nombre de dominio de la aplicación a la que desea dirigir el tráfico.

- Tipo de registro: acepte el valor predeterminado, A: dirección IPv4.
- Evalúe el estado del objetivo: acepte el valor predeterminado, sí.

6. Elija Crear registros.

El registro de alias de Route 53 que creó se propaga en todos los servidores de Route 53 en 60 segundos. Cuando los servidores de Route 53 se propaguen con tu registro de alias, puedes enrutar el tráfico a tu servicio de App Runner utilizando el nombre del registro de alias que creaste.

Para obtener información sobre cómo solucionar problemas si los cambios de DNS tardan demasiado en propagarse, consulte ¿Por qué mis cambios de DNS tardan tanto en propagarse en Route 53 y en los resolvers públicos? .

Amazon Route 53 API or AWS CLI

Para configurar el registro de alias de Amazon Route 53 mediante la API de Amazon Route 53 o realizar una AWS CLI llamada a la acción de la ChangeResourceRecordSetsAPI. Para obtener más información sobre el identificador de la zona alojada de destino de Route 53, consulte los puntos de enlace del servicio.

Pausar y reanudar un servicio de App Runner

Si necesitas deshabilitar tu aplicación web temporalmente y detener la ejecución del código, puedes pausar el AWS App Runner servicio. App Runner reduce la capacidad de computación del servicio a cero.

Cuando estés listo para volver a ejecutar la aplicación, puedes reanudar el servicio de App Runner. App Runner aprovisiona nueva capacidad de computación, implementa la aplicación en ella y la ejecuta. El código fuente de la aplicación no se ha vuelto a implementar y no es necesario compilarlo. Por el contrario, App Runner se reanuda con la versión actualmente implementada. La aplicación conserva su dominio de App Runner.

Important

- Al pausar el servicio, la aplicación pierde su estado. Por ejemplo, se pierde cualquier almacenamiento efímero que haya utilizado el código. En el caso del código, pausar y reanudar el servicio equivale a implementarlo en un servicio nuevo.

- Si pausas un servicio debido a una falla en el código (por ejemplo, un error descubierto o un problema de seguridad), no podrás implementar una nueva versión antes de reanudar el servicio.

  Por lo tanto, te recomendamos que mantengas el servicio en funcionamiento y, en su lugar, vuelvas a la última versión estable de la aplicación.
- Al reanudar el servicio, App Runner despliega la última versión de la aplicación que se utilizó antes de pausar el servicio. Si agregaste nuevas versiones de origen desde que pausaste el servicio, App Runner no las implementará automáticamente aunque hayas seleccionado la implementación automática. Por ejemplo, supongamos que tienes nuevas versiones de imágenes en el repositorio de imágenes o nuevas confirmaciones en el repositorio de código. Estas versiones no se implementan automáticamente.

  Para implementar una versión más reciente, realiza una implementación manual o agrega otra versión al repositorio de origen después de reanudar el servicio de App Runner.

Cómo pausar y eliminar comparados

Pausa el servicio App Runner para deshabilitarlo temporalmente. Solo se cancelan los recursos de cómputo y los datos almacenados (por ejemplo, la imagen del contenedor con la versión de la aplicación) permanecen intactos. Reanudar el servicio es rápido: la aplicación está lista para implementarse en nuevos recursos de cómputo. Tu dominio de App Runner sigue siendo el mismo.

Elimina tu servicio App Runner para eliminarlo permanentemente. Se eliminarán los datos almacenados. Si necesitas volver a crear el servicio, App Runner tiene que volver a buscar el código fuente y también compilarlo si se trata de un repositorio de código. La aplicación web obtiene un nuevo dominio de App Runner.

Cuando tu servicio está en pausa

Cuando pausas tu servicio y se encuentra en el estado En pausa, responde de forma diferente a las solicitudes de acción, incluidas las llamadas a la API o las operaciones de la consola. Cuando un servicio está en pausa, puedes seguir realizando acciones de App Runner que no modifiquen la definición o la configuración del servicio de forma que afecte a su tiempo de ejecución. En otras palabras, si una acción cambia el comportamiento, la escala u otras características de un servicio en ejecución, no podrás realizar esa acción en un servicio pausado.

Las siguientes listas proporcionan información sobre las acciones de la API que puedes y no puedes realizar en un servicio pausado. Del mismo modo, se permiten o deniegan las operaciones de consola equivalentes.

Acciones: tú poder realizar en un servicio pausado

- *`List*`*y *`Describe*`* acciones: acciones que solo leen información.
- *`DeleteService`*— Siempre puedes eliminar un servicio.
- *`TagResource`*, *`UntagResource`* — Las etiquetas están asociadas a un servicio, pero no forman parte de su definición y no afectan a su comportamiento en tiempo de ejecución.

Acciones: tú no poder realizar en un servicio pausado

- *`StartDeployment`*acciones (o un despliegue manual mediante la consola)
- *`UpdateService`*(o un cambio de configuración mediante la consola, excepto en lo que respecta a los cambios de etiquetado)
- *`CreateCustomDomainAssociations`*, *`DeleteCustomDomainAssociations`*
- *`CreateConnection`*, *`DeleteConnection`*

Pausa y reanuda el servicio

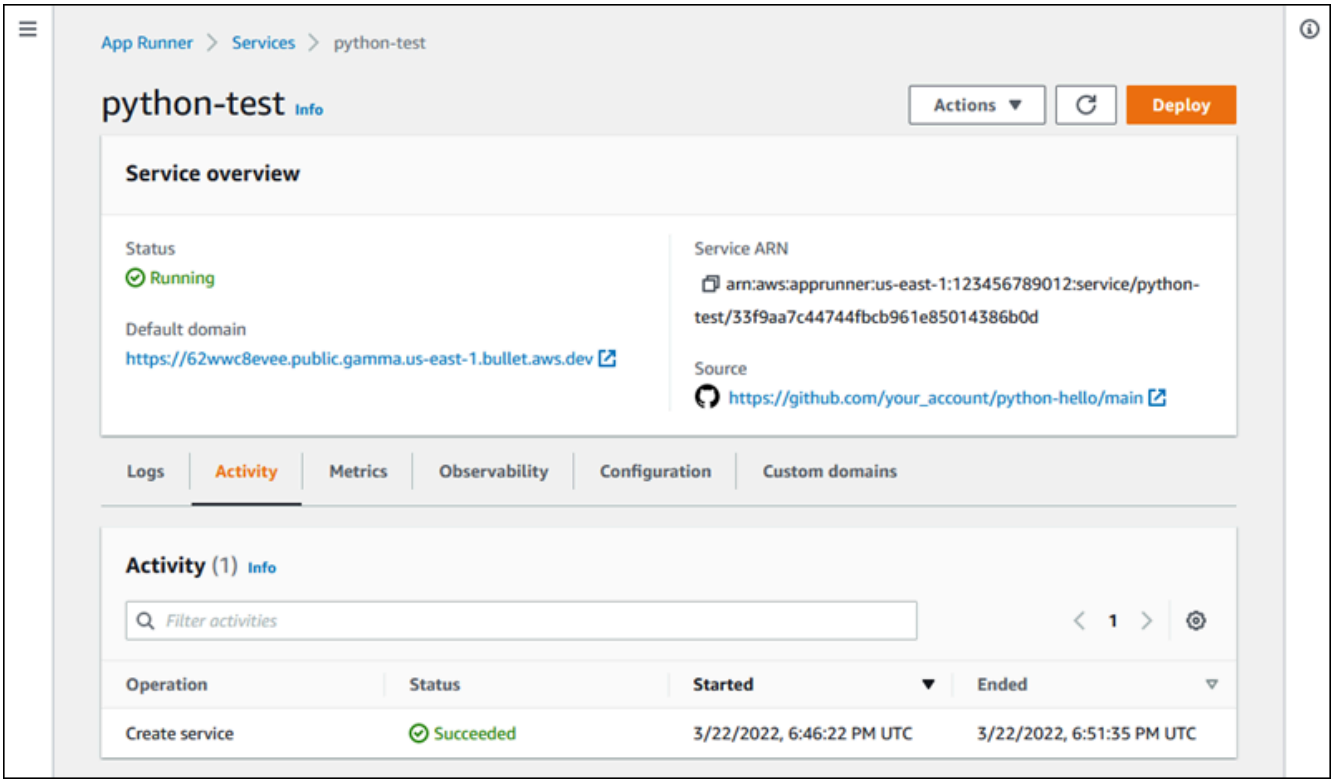
Pausa y reanuda el servicio de App Runner mediante uno de los siguientes métodos:

App Runner console

Para pausar el servicio mediante la consola de App Runner

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.



3. Selecciona Acciones y, a continuación, selecciona Pausa.

   En la página del panel de control del servicio, el estado del servicio cambia a Funcionamiento en curso y, a continuación, cambia a En pausa. El servicio ahora está en pausa.

Para reanudar el servicio mediante la consola de App Runner

1. Selecciona Acciones y, a continuación, selecciona Reanudar.

   En la página del panel de control del servicio, el estado del servicio cambia a Funcionamiento en curso.

2. Espere a que se reanude el servicio. En la página del panel de control del servicio, el estado del servicio vuelve a ser En ejecución.
3. Para comprobar que la reanudación del servicio se ha realizado correctamente, en la página del panel de control del servicio, selecciona el valor del dominio de App Runner. Es la URL del sitio web de tu servicio. Comprueba que la aplicación web se ejecuta correctamente.

App Runner API or AWS CLI

Para pausar tu servicio mediante la API de App Runner o AWS CLI llama a la acción de la PauseServiceAPI. Si la llamada devuelve una respuesta correcta y muestra un objeto de servicio`"Status": "OPERATION_IN_PROGRESS"`, App Runner comienza a pausar el servicio.

Para reanudar el servicio mediante la API de App Runner o bien AWS CLI, llama a la acción de la ResumeServiceAPI. Si la llamada devuelve una respuesta correcta y muestra un objeto de servicio`"Status": "OPERATION_IN_PROGRESS"`, App Runner comienza a reanudar el servicio.

Eliminar un servicio de App Runner

Cuando desee finalizar la aplicación web que se está ejecutando en su AWS App Runner servicio, puede eliminarlo. Al eliminar un servicio, se detiene el servicio web en ejecución, se eliminan los recursos subyacentes y se eliminan los datos asociados.

Es posible que desee eliminar un servicio de App Runner por uno o varios de los siguientes motivos:

- Ya no necesitas la aplicación web: por ejemplo, está retirada o es una versión de desarrollo que ya no utilizas.
- Has alcanzado la cuota de servicio de App Runner: quieres crear un nuevo servicio en la misma Región de AWS y has alcanzado la cuota asociada a tu cuenta. Para obtener más información, consulte the section called “Cuotas de recursos de App Runner”.
- Consideraciones de seguridad o privacidad: quieres que App Runner elimine los datos que almacena para tu servicio.

Comparación entre pausar y eliminar

Pausa el servicio App Runner para deshabilitarlo temporalmente. Solo se cancelan los recursos de cómputo y los datos almacenados (por ejemplo, la imagen del contenedor con la versión de la aplicación) permanecen intactos. Reanudar el servicio es rápido: la aplicación está lista para implementarse en nuevos recursos de cómputo. Tu dominio de App Runner sigue siendo el mismo.

Elimina tu servicio App Runner para eliminarlo permanentemente. Se eliminarán los datos almacenados. Si necesitas volver a crear el servicio, App Runner tiene que volver a buscar el código fuente y también compilarlo si se trata de un repositorio de código. La aplicación web obtiene un nuevo dominio de App Runner.

¿Qué elimina App Runner?

Cuando eliminas tu servicio, App Runner elimina algunos elementos asociados y no elimina otros. En las siguientes listas se proporcionan los detalles.

Elementos que App Runner elimina:

- Imagen contenedora: una copia de la imagen que implementaste o de la imagen que App Runner creó a partir de tu código fuente. Se almacena en Amazon Elastic Container Registry (Amazon ECR) mediante archivos Cuentas de AWS internos propiedad de App Runner.
- Configuración del servicio: los ajustes de configuración que están asociados a su servicio de App Runner. Se almacenan en Amazon DynamoDB mediante archivos Cuentas de AWS internos que son propiedad de App Runner.

Elementos que App Runner no elimina:

- Conexión: es posible que tengas una conexión asociada a tu servicio. Una conexión de App Runner es un recurso independiente que puede compartirse entre varios servicios de App Runner. Si ya no necesitas la conexión, puedes eliminarla de forma explícita. Para obtener más información, consulte the section called “Conexiones”.
- Certificados de dominio personalizados: si vinculas dominios personalizados a un servicio de App Runner, App Runner crea internamente certificados que rastrean la validez del dominio. Se almacenan en AWS Certificate Manager (ACM). App Runner no elimina el certificado hasta siete días después de desvincular un dominio de tu servicio o de eliminarlo. Para obtener más información, consulte the section called “Nombres de dominio personalizados”.

Elimina tu servicio

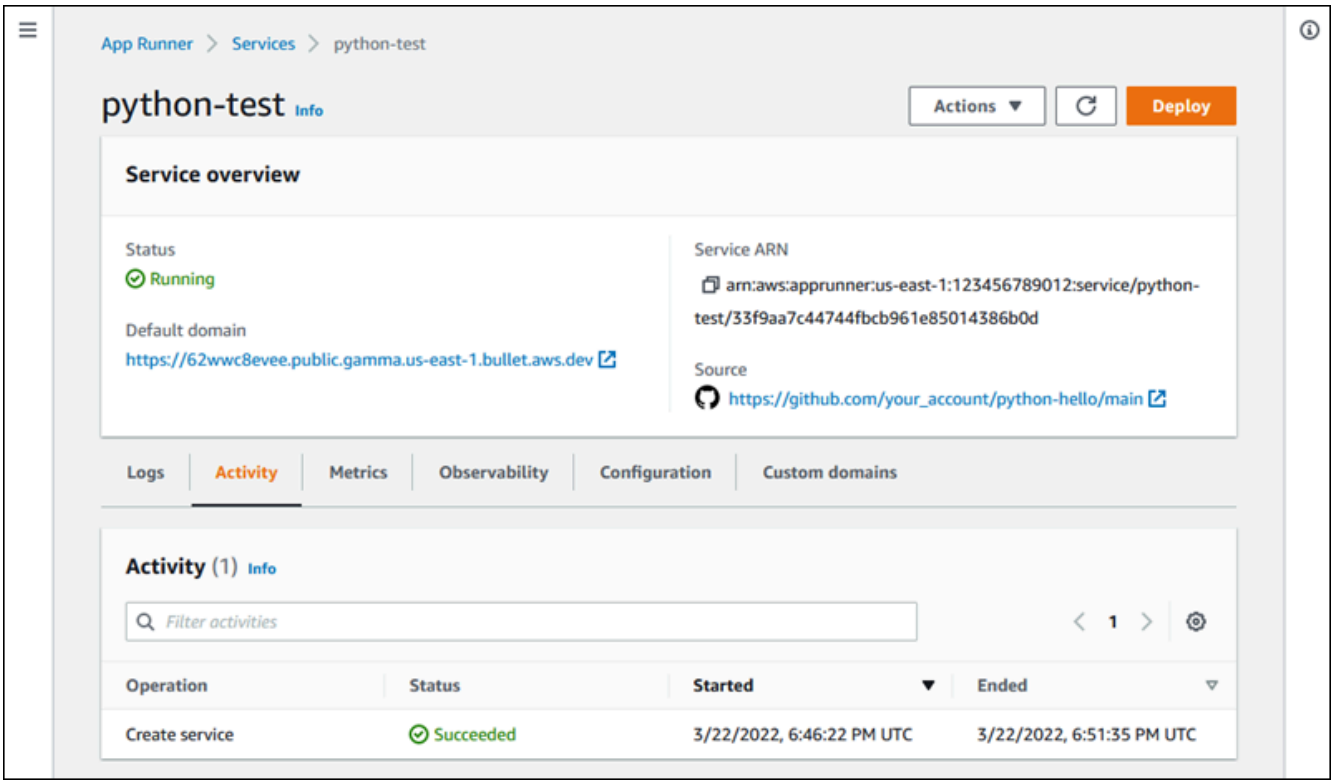
Elimine el servicio App Runner mediante uno de los siguientes métodos:

App Runner console

Para eliminar tu servicio mediante la consola de App Runner

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

La consola muestra el panel de servicios con una descripción general del servicio.



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

   La consola lo lleva a la página de servicios. El servicio eliminado muestra el estado Operación en curso y, a continuación, el servicio desaparece de la lista. Su servicio ahora está eliminado.

App Runner API or AWS CLI

Para eliminar tu servicio mediante la API de App Runner o AWS CLI, llama a la acción de la DeleteServiceAPI. Si la llamada devuelve una respuesta correcta y muestra un objeto de servicio`"Status": "OPERATION_IN_PROGRESS"`, App Runner empezará a eliminar el servicio.

Hacer referencia a variables de entorno

Con App Runner, puedes hacer referencia a los secretos y las configuraciones como variables de entorno de tu servicio al crear un servicio o actualizarlo.

Puede hacer referencia a datos de configuración no confidenciales, como los tiempos de espera y los recuentos de reintentos en texto sin formato, como pares clave-valor. Los datos de configuración a los que hace referencia en texto sin formato no están cifrados y otras personas los pueden ver en los registros de aplicaciones y de configuración del servicio de App Runner.

> **Note**
>
> Por motivos de seguridad, no hagas referencia a ningún dato confidencial en texto sin formato en tu servicio de App Runner.

Hacer referencia a datos confidenciales como variables de entorno

App Runner permite hacer referencia de forma segura a datos confidenciales como variables de entorno en su servicio. Considere la posibilidad de almacenar los datos confidenciales a los que desee hacer referencia en nuestro AWS Secrets Manageralmacén de AWS Systems Manager parámetros. Luego, puedes hacer referencia a ellos de forma segura en tu servicio como variables de entorno desde la consola de App Runner o llamando a la API. Esto separa eficazmente la administración de secretos y parámetros del código de la aplicación y la configuración del servicio, lo que mejora la seguridad general de las aplicaciones que se ejecutan en App Runner.

> **Note**
>
> App Runner no le cobra por hacer referencia a Secrets Manager y SSM Parameter Store como variables de entorno. Sin embargo, usted paga un precio estándar por usar Secrets Manager y SSM Parameter Store.
> Para obtener más información sobre los precios, consulte los siguientes temas:
>
> - AWS Precios de Secrets Manager
> - AWS Precios de SSM Parameter Store

El siguiente es el proceso para hacer referencia a datos confidenciales como variables de entorno:

1. Guarde los datos confidenciales, como las claves de API, las credenciales de la base de datos, los parámetros de conexión a la base de datos o las versiones de las aplicaciones, como secretos o parámetros en el almacén de parámetros AWS Secrets Manager o en el almacén de AWS Systems Manager parámetros.
2. Actualiza la política de IAM de tu rol de instancia para que App Runner pueda acceder a los secretos y parámetros almacenados en Secrets Manager y SSM Parameter Store. Para obtener más información, consulte Permisos de ..
3. Haga referencia de forma segura a los secretos y parámetros como variables de entorno asignando un nombre y proporcionando su nombre de recurso de Amazon (ARN). Puede añadir variables de entorno al crear un servicio o al actualizar la configuración de un servicio. Puede usar una de las siguientes opciones para agregar variables de entorno:
   - Consola de App Runner
   - API de App Runner
   - Archivo de configuración de la `apprunner.yaml`

   > **Note**
   >
   > No puedes asignar `PORT` un nombre a una variable de entorno al crear o actualizar tu servicio de App Runner. Es una variable de entorno reservada para el servicio App Runner.

   Para obtener más información sobre cómo hacer referencia a secretos y parámetros, consulte Administrar variables de entorno.

> **Note**
>
> Como App Runner solo almacena la referencia al secreto y al parámetro ARNs, los datos confidenciales no son visibles para otras personas en la configuración del servicio y en los registros de aplicaciones de App Runner.

Consideraciones

- Asegúrate de actualizar el rol de la instancia con los permisos adecuados para acceder a los secretos y parámetros del almacén de parámetros AWS Secrets Manager o AWS Systems Manager dentro de él. Para obtener más información, consulte Permisos de ..
- Asegúrate de que el almacén de AWS Systems Manager parámetros esté en el Cuenta de AWS mismo lugar que el servicio que deseas lanzar o actualizar. Actualmente, no puedes hacer referencia a los parámetros del almacén de parámetros de SSM en todas las cuentas.
- Cuando los secretos y los valores de los parámetros se rotan o cambian, no se actualizan automáticamente en tu servicio de App Runner. Vuelva a implementar el servicio App Runner, ya que App Runner solo extrae los secretos y los parámetros durante la implementación.
- También tienes la opción de llamar AWS Secrets Manager directamente al almacén de AWS Systems Manager parámetros a través del SDK de tu servicio de App Runner.
- Para evitar errores, asegúrate de lo siguiente cuando hagas referencia a ellas como variables de entorno:
  - Especifica el ARN correcto del secreto.
  - Debe especificar el nombre correcto o el ARN del parámetro.

Permisos

Para habilitar la referencia a los secretos y parámetros almacenados en el almacén de parámetros AWS Secrets Manager o en el almacén de parámetros de SSM, añade los permisos adecuados a la política de IAM de tu función de instancia para acceder a Secrets Manager y al almacén de parámetros de SSM.

> **Note**
>
> App Runner no puede acceder a los recursos de tu cuenta sin tu permiso. El permiso se otorga mediante la actualización de la política de IAM.

Puedes usar las siguientes plantillas de políticas para actualizar tu rol de instancia en la consola de IAM. Puede modificar estas plantillas de políticas para adaptarlas a sus requisitos específicos. Para obtener más información sobre la actualización de un rol de instancia, consulte Modificación de un rol en la Guía del usuario de IAM.

> **Note**
>
> También puedes copiar las siguientes plantillas de la consola de App Runner al crear las variables de entorno.

Copia la siguiente plantilla en tu rol de instancia para añadir permisos desde los que hacer referencia a los secretos AWS Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue",
            "kms:Decrypt*"
        ],
        "Resource": [
            "arn:aws:secretsmanager:us-east-1:111122223333:secret:my-secret",
            "arn:aws:kms:us-east-1:111122223333:key/my-key"
         ]
     }
   ]
}
```

Copia la siguiente plantilla en tu rol de instancia para añadir permisos para hacer referencia a los parámetros del almacén de AWS Systems Managerparámetros.

JSON

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

```
                "ssm:GetParameters"
            ],
            "Resource": [
            "arn:aws:ssm:us-east-1:111122223333:parameter/my-parameter"
            ]
        }
    ]
}
```

Administrar las variables de entorno

Administre las variables de entorno de su servicio de App Runner mediante uno de los siguientes métodos:

- the section called “Consola de App Runner”
- the section called “API de App Runner o AWS CLI”

Consola de App Runner

Al crear un servicio o actualizar un servicio en la consola de App Runner, puede agregar variables de entorno.

Agregar una variable de entorno

Para añadir una variable de entorno

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En función de si va a crear o actualizar un servicio, lleve a cabo uno de los siguientes pasos:
   - Si va a crear un servicio nuevo, elija Crear un servicio de App Runner y vaya a Configurar el servicio.
   - Si va a actualizar un servicio existente, seleccione el servicio que desee actualizar y vaya a la pestaña Configuración del servicio.
3. Ve a Variables de entorno (opcional) en Configuración del servicio.
4. Elija una de las siguientes opciones en función de sus necesidades:

- Seleccione Texto sin formato en la fuente de la variable de entorno e introduzca sus pares clave-valor en Nombre de la variable de entorno y Valor de la variable de entorno, respectivamente.

  > **Note**
  >
  > Elija Texto sin formato si desea hacer referencia a datos no confidenciales. Estos datos no están cifrados y otras personas los pueden ver en la configuración del servicio App Runner y en los registros de la aplicación.

- Elija Secrets Manager en la fuente de la variable de entorno para hacer referencia al secreto que está almacenado AWS Secrets Manager como variable de entorno en su servicio. Proporcione el nombre de la variable de entorno y el nombre del recurso de Amazon (ARN) del secreto al que hace referencia en Nombre de la variable de entorno y Valor de la variable de entorno, respectivamente.
- Elija SSM Parameter Store en la fuente de variables de entorno para hacer referencia al parámetro almacenado en SSM Parameter Store como variable de entorno de su servicio. Proporcione el nombre de la variable de entorno y el ARN del parámetro al que hace referencia en Nombre de la variable de entorno y Valor de la variable de entorno, respectivamente.

  > **Note**
  >
  > - No puedes asignar `PORT` un nombre a una variable de entorno al crear o actualizar tu servicio de App Runner. Es una variable de entorno reservada para el servicio App Runner.
  > - Si el parámetro del almacén de parámetros de SSM coincide Región de AWS con el servicio que desea lanzar, puede especificar el nombre completo del recurso de Amazon (ARN) o el nombre del parámetro. Si el parámetro se encuentra en una región diferente, debe especificar el ARN completo.
  > - Asegúrese de que el parámetro al que hace referencia esté en la misma cuenta que el servicio que va a lanzar o actualizar. Actualmente, no puedes hacer referencia al parámetro SSM Parameter Store en todas las cuentas.

5. Seleccione Añadir variable de entorno para hacer referencia a otra variable de entorno.

6. Amplíe las plantillas de políticas de IAM para ver y copiar las plantillas de políticas de IAM proporcionadas para el almacén de parámetros de SSM AWS Secrets Manager y el SSM. Solo tienes que hacerlo si aún no has actualizado la política de IAM de tu rol de instancia con los permisos necesarios. Para obtener más información, consulte Permisos de ..

Eliminar la variable de entorno

Antes de eliminar una variable de entorno, asegúrese de que el código de la aplicación esté actualizado para reflejar lo mismo. Si el código de la aplicación no está actualizado, es posible que el servicio App Runner falle.

Para eliminar variables de entorno

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Ve a la pestaña Configuración del servicio que deseas actualizar.
3. Ve a Variables de entorno (opcional) en Configuración del servicio.
4. Seleccione Eliminar junto a la variable de entorno que desee eliminar. Recibirá un mensaje para confirmar la eliminación.
5. Elija Eliminar.

API de App Runner o AWS CLI

Puede hacer referencia a los datos confidenciales almacenados en Secrets Manager y SSM Parameter Store agregándolos como variables de entorno en su servicio.

> **Note**
>
> Actualiza la política de IAM de tu rol de instancia para que App Runner pueda acceder a los secretos y parámetros almacenados en Secrets Manager y SSM Parameter Store. Para obtener más información, consulte Permisos de ..

Para hacer referencia a los secretos y las configuraciones como variables de entorno

1. Cree un secreto o una configuración en Secrets Manager o SSM Parameter Store.

En los siguientes ejemplos se muestra cómo crear un secreto y un parámetro mediante el almacén de parámetros SSM.

Example Crear un secreto: solicitud

El siguiente ejemplo muestra cómo crear un secreto que represente la credencial de la base de datos.

```
aws secretsmanager create-secret \
–name DevRdsCredentials \
–description "Rds credentials for development account." \
–secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
```

Example Crear un secreto: respuesta

```
arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
```

Example Crear una configuración: solicitud

El siguiente ejemplo muestra cómo crear un parámetro que represente la cadena de conexión RDS.

```
aws systemsmanager put-parameter \
–name DevRdsConnectionString \
–value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
–type "String" \
–description "Rds connection string for development account."
```

Example Creación de una configuración: respuesta

```
arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
```

2. Haga referencia a los secretos y las configuraciones que se almacenan en Secrets Manager y SSM Parameter Store agregándolos como variables de entorno. Puedes agregar variables de entorno al crear o actualizar tu servicio de App Runner.

   En los siguientes ejemplos, se muestra cómo hacer referencia a los secretos y las configuraciones como variables de entorno en un servicio de App Runner basado en código y en uno basado en imágenes.

Example Archivo de entrada. json para el servicio App Runner basado en imágenes

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "<image-identifier>",
      "ImageConfiguration": {
        "Port": "<port>",
        "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
          "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
        }
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Servicio App Runner basado en imágenes: solicitud

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example Servicio App Runner basado en imágenes: respuesta

```
{
...
      "ImageRepository": {
         "ImageIdentifier":"<image-identifier>",
         "ImageConfiguration":{
            "Port": "<port>",
            "RuntimeEnvironmentSecrets":{
```

```
               "Credential1":
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
               "Credential2": "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
            },
            "ImageRepositoryType":"ECR"
      }
   },
   "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

Example Archivo de entrada. json para el servicio App Runner basado en código

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-
github-connection/XXXXXXXXXX"
    },
    "AutoDeploymentsEnabled": false,
    "CodeRepository": {
      "RepositoryUrl": "<repository-url>",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      },
      "CodeConfiguration": {
        "ConfigurationSource": "API",
        "CodeConfigurationValues": {
          "Runtime": "<runtime>",
          "BuildCommand": "<build-command>",
          "StartCommand": "<start-command>",
          "Port": "<port>",
          "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
```

```
            "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
          }
        }
      }
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Servicio App Runner basado en código: solicitud

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example Servicio App Runner basado en código: respuesta

```
{
...
   "SourceConfiguration":{
      "CodeRepository":{
         "RepositoryUrl":"<repository-url>",
         "SourceCodeVersion":{
            "Type":"Branch",
            "Value":"main"
         },
         "CodeConfiguration":{
            "ConfigurationSource":"API",
            "CodeConfigurationValues":{
               "Runtime":"<runtime>",
               "BuildCommand":"<build-command>",
               "StartCommand":"<start-command>",
               "Port":"<port>",
               "RuntimeEnvironmentSecrets":{
                  "Credential1" :
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXX",
                  "Credential2" : "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
```

```
                }
            }
          }
      },
      "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

3. El `apprunner.yaml` modelo se actualiza para reflejar los secretos adicionales.

   A continuación se muestra un ejemplo del `apprunner.yaml` modelo actualizado.

   Example **`apprunner.yaml`**

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - python -m pip install flask
run:
  command: python app.py
  network:
    port: 8080
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from:
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX"
    - name: my-parameter
      value-from: "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter-
name>"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Redes con App Runner

En este capítulo, se describen las configuraciones de red de sus AWS App Runner servicios.

En este capítulo, aprenderá lo siguiente:

- Cómo configurar el tráfico entrante para puntos finales públicos y privados. Para obtener más información, consulte Configurar las configuraciones de red para el tráfico entrante.
- Cómo configurar el tráfico saliente para acceder a otras aplicaciones que se ejecutan en una Amazon VPC. Para obtener más información, consulte Habilitar el acceso a la VPC para el tráfico saliente.

Temas

Terminología

Para saber cómo personalizar el tráfico de la red para adaptarlo a sus necesidades, comprendamos los siguientes términos que se utilizan en este capítulo.

Términos generales

Para saber qué se necesita para asociarse a una Amazon Virtual Private Cloud (VPC), comprendamos los siguientes términos:

- VPC: una VPC de Amazon es una red virtual aislada de forma lógica que le proporciona un control total sobre su entorno de red virtual, incluida la ubicación de los recursos, la conectividad y la seguridad. Se trata de una red virtual que se parece mucho a una red tradicional que utilizaría en su propio centro de datos.
- Punto final de la interfaz de VPC: el punto final de la interfaz de VPC, un AWS PrivateLink recurso, conecta una VPC a un servicio de punto final. Cree un punto final de interfaz de VPC para enviar tráfico a los servicios de punto final que utilizan un Network Load Balancer para distribuir el tráfico. El tráfico destinado al servicio de punto de conexión se resuelve mediante DNS.

- Regiones: cada región es un área geográfica independiente en la que puedes alojar un servicio de App Runner.
- Zonas de disponibilidad: una zona de disponibilidad es una ubicación aislada dentro de una AWS región. Se trata de uno o más centros de datos discretos con alimentación, redes y conectividad redundantes. Las zonas de disponibilidad permiten que las aplicaciones de producción sean altamente disponibles, tolerantes a errores y tengan escalabilidad.
- Subredes: una subred es un rango de direcciones IP de la VPC. Una subred debe residir en una sola zona de disponibilidad. Puede lanzar un AWS recurso en una subred específica. Utilice una subred pública para los recursos que deben conectarse a Internet y una subred privada para los recursos que no dispondrán de conexión a Internet.
- Grupos de seguridad: un grupo de seguridad controla el tráfico que puede llegar y salir de los recursos a los que está asociado. Los grupos de seguridad proporcionan un nivel de seguridad adicional para proteger los AWS recursos de cada subred, lo que le permite controlar mejor el tráfico de la red. Al crear una VPC, incluye un grupo de seguridad predeterminado. Puede crear grupos de seguridad adicionales para cada VPC. Puede asociar un grupo de seguridad solo a los recursos de la VPC para la que se creó.
- Pila doble: una pila doble es un tipo de dirección que admite el tráfico de red tanto desde los puntos de conexión como IPv4 desde los puntos finales. IPv6

Término específico de la configuración del tráfico saliente

Conector VPC

Un conector de VPC es un recurso de App Runner que permite al servicio App Runner acceder a las aplicaciones que se ejecutan en una Amazon VPC privada.

Términos específicos para configurar el tráfico entrante

Para saber cómo puedes hacer que tus servicios sean accesibles de forma privada solo desde una Amazon VPC, entendamos los siguientes términos:

- Conexión de entrada de VPC: la conexión de entrada de VPC es un recurso de App Runner que proporciona un punto final de App Runner para el tráfico entrante. App Runner asigna el recurso de conexión de entrada de VPC entre bastidores cuando eliges un punto final privado en la consola de App Runner para el tráfico entrante. El recurso de conexión de entrada de VPC conecta el servicio App Runner con el punto final de la interfaz de VPC de Amazon VPC.

> **Note**
>
> Si utilizas la API de App Runner, el recurso de conexión de entrada de VPC no se crea automáticamente.

- Punto final privado: el punto final privado es una opción de consola de App Runner que se selecciona para configurar el tráfico de red entrante para que solo se pueda acceder a él desde una Amazon VPC.

Configuración de redes para el tráfico entrante

Puede configurar su servicio para recibir tráfico entrante desde un punto final público o privado.

La configuración predeterminada es un punto final público. Abre el servicio a cualquier tráfico entrante de la Internet pública. También le brinda la flexibilidad de elegir entre IPv4 dos tipos de direcciones (IPv4 y IPv6) para su servicio.

Un punto de conexión privado solo permite que el tráfico de una VPC de Amazon acceda a tu servicio App Runner. Esto se logra configurando un punto final de interfaz de VPC, un AWS PrivateLink recurso, para su servicio de App Runner. De este modo, se crea una conexión privada entre Amazon VPC y su servicio App Runner. También le ofrece la flexibilidad de elegir entre IPv4 dos tipos de direcciones (IPv4 y IPv6) para su servicio.

Los siguientes son los temas que se tratan como parte de la configuración de la red para el tráfico entrante:

- Cómo configurar el tráfico entrante para que el servicio esté disponible de forma privada solo desde una Amazon VPC. Para obtener más información, consulte Habilitar un punto final privado para el tráfico entrante.
- Cómo configurar su servicio para recibir tráfico de Internet desde el tipo de dirección de doble pila. Para obtener más información, consulte Habilitar la doble pila para el tráfico entrante público.

Encabezados

Con App Runner puedes acceder a la fuente original IPv4 y a IPv6 las direcciones del tráfico que entra en tu aplicación. Las direcciones IP de origen originales se conservan al asignarles el

encabezado de la `X-Forwarded-For` solicitud. Esto permite que sus aplicaciones obtengan las direcciones IP de origen originales cuando sea necesario.

> **Note**
>
> Si su servicio está configurado para usar un punto final privado, el encabezado de la `X-Forwarded-For` solicitud no se puede usar para acceder a las direcciones IP de origen originales. Si se usa, recupera valores falsos.

Habilitación de un punto final privado para el tráfico entrante

De forma predeterminada, al crear un AWS App Runner servicio, se puede acceder al servicio a través de Internet. Sin embargo, también puede hacer que su servicio App Runner sea privado y solo sea accesible desde una Amazon Virtual Private Cloud (Amazon VPC).

Con tu servicio App Runner privado, tienes un control total sobre el tráfico entrante, lo que añade un nivel de seguridad adicional. Esto resulta útil en una variedad de casos de uso, como la ejecución de aplicaciones web internas APIs, corporativas o aplicaciones que aún están en desarrollo y que requieren un mayor nivel de privacidad y seguridad, o que necesitan cumplir requisitos de conformidad específicos.

> **Note**
>
> Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante desde la fuente, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar WAF web. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP. Para obtener más información sobre la seguridad de la infraestructura y los grupos de seguridad, incluidas las prácticas recomendadas, consulte los siguientes temas de la Guía del usuario de Amazon VPC: Controle el tráfico de red y Controle el tráfico a sus recursos de AWS mediante grupos de seguridad.

Cuando su servicio App Runner es privado, puede acceder a él desde una Amazon VPC. No se requiere una puerta de enlace a Internet, un dispositivo NAT o una conexión VPN.

> **Note**
>
> App Runner admite IPv4 una pila doble (ambos IPv4 IPv6), tanto para el tráfico entrante como para el saliente.

Consideraciones

- Antes de configurar un punto final de interfaz de VPC para App Runner, consulte las consideraciones de la AWS PrivateLink guía.
- Las políticas de puntos de conexión de VPC no son compatibles con App Runner. De forma predeterminada, se permite el acceso total a App Runner a través del punto final de la interfaz de VPC. Como alternativa, puede asociar un grupo de seguridad a las interfaces de red de puntos finales para controlar el tráfico a App Runner a través del punto final de la interfaz de VPC.
- Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar la web de WAF. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP.
- Después de habilitar un punto de conexión privado, solo se puede acceder al servicio desde la VPC y no se puede acceder a él desde Internet.
- Para una mayor disponibilidad, se recomienda seleccionar al menos dos subredes en la zona de disponibilidad diferentes para el punto final de la interfaz de VPC. No recomendamos usar solo una subred.
- Si elige la opción de doble pila para el tipo de dirección IP, asegúrese de que sus subredes puedan admitir el tráfico de doble pila.
- Puede usar el mismo punto final de la interfaz de VPC para acceder a varios servicios de App Runner en una VPC.

Para obtener información sobre los términos utilizados en esta sección, consulte Terminología.

Permisos

La siguiente es la lista de permisos necesarios para habilitar el punto final privado:

- ec2: CreateTags
- ec2: CreateVpcEndpoint
- ec2: ModifyVpcEndpoint
- ec2: DeleteVpcEndpoints
- ec2: DescribeSubnets
- ec2: DescribeVpcEndpoints
- ec2: DescribeVpcs

Punto final de la interfaz VPC

Un punto final de interfaz de VPC es un AWS PrivateLinkrecurso que conecta una Amazon VPC a un servicio de punto final. Puede especificar en qué Amazon VPC desea que se pueda acceder a su servicio App Runner pasando por un punto final de la interfaz de VPC. Para crear un punto final de interfaz de VPC, especifique lo siguiente:

- La Amazon VPC para habilitar la conectividad.
- Agregue grupos de seguridad. De forma predeterminada, se asigna un grupo de seguridad al punto final de la interfaz de VPC. Puede optar por asociar un grupo de seguridad personalizado para controlar aún más el tráfico de red entrante.
- Agregue subredes. Para garantizar una mayor disponibilidad, se recomienda seleccionar al menos dos subredes para cada zona de disponibilidad desde la que accederás al servicio App Runner. Se crea un punto final de la interfaz de red en cada subred que se habilita para el punto final de la interfaz de VPC. Se trata de interfaces de red administradas por el solicitante que sirven como punto de entrada para el tráfico destinado a App Runner. Una interfaz de red administrada por el solicitante es una interfaz de red que un AWS servicio crea en la VPC en su nombre.
- Si usa la API, agregue el punto final de la interfaz de VPC de App Runner. `Servicename` Por ejemplo:

```
com.amazonaws.region.apprunner.requests
```

Puede crear un punto final de interfaz de VPC mediante uno de los siguientes servicios: AWS

- Consola de App Runner. Para obtener más información, consulte Administrar un punto final privado.

- Consola o API de Amazon VPC y AWS Command Line Interface ()AWS CLI. Para obtener más información, consulte Acceso a los Servicios de AWS a través de AWS PrivateLink en la Guía de AWS PrivateLink .

> **Note**
>
> Se le cobrará por cada punto final de la interfaz de VPC que utilice en AWS PrivateLink función del precio. Por lo tanto, para mejorar la rentabilidad, puede usar el mismo punto final de la interfaz de VPC para acceder a varios servicios de App Runner dentro de una VPC. Sin embargo, para un mejor aislamiento, considere la posibilidad de asociar un punto final de interfaz de VPC diferente para cada uno de sus servicios de App Runner.

Conexión de ingreso de VPC

Una conexión de entrada de VPC es un recurso de App Runner que especifica un punto final de App Runner para el tráfico entrante. App Runner asigna el recurso de conexión de entrada de VPC entre bastidores cuando eliges un punto final privado en la consola de App Runner para el tráfico entrante. Elija esta opción para permitir que solo el tráfico de una VPC de Amazon acceda a su servicio App Runner. El recurso de conexión de entrada de VPC conecta el servicio App Runner con el punto final de la interfaz de VPC de Amazon VPC. Puede crear un recurso de conexión de entrada de VPC solo si utiliza las operaciones de la API para configurar los ajustes de red para el tráfico entrante. Para obtener más información sobre cómo crear un recurso de conexión de entrada de VPC, consulte la referencia de CreateVpcIngressConnectionla AWS App Runner API.

> **Note**
>
> Un recurso de conexión de entrada de VPC de App Runner puede conectarse a un punto final de la interfaz de VPC de Amazon VPC. Además, solo puede crear un recurso de conexión de entrada de VPC para cada servicio de App Runner.

Punto final privado

El punto de conexión privado es una opción de consola de App Runner que puedes elegir si solo quieres recibir tráfico entrante de una VPC de Amazon. Al elegir la opción de punto de enlace privado en la consola de App Runner, tiene la opción de conectar el servicio a una VPC mediante

la configuración de su punto de enlace de interfaz de VPC. Entre bastidores, App Runner asigna un recurso de conexión de entrada de VPC al punto final de la interfaz de VPC que usted configure.

Resumen

Haga que su servicio sea privado permitiendo que solo el tráfico de una VPC de Amazon acceda a su servicio de App Runner. Para ello, debe crear un punto final de interfaz de VPC para la Amazon VPC seleccionada mediante App Runner o Amazon VPC. En la consola de App Runner, se crea un punto de enlace de la interfaz de VPC al habilitar el punto de enlace privado para el tráfico entrante. A continuación, App Runner crea automáticamente un recurso de conexión de entrada de VPC y se conecta al punto final de la interfaz de VPC y a su servicio de App Runner. Esto crea una conexión de servicio privada que garantiza que solo el tráfico de la VPC seleccionada pueda acceder al servicio de App Runner.

Administración de puntos finales privados

Administre el punto final privado para el tráfico entrante mediante uno de los siguientes métodos:

- the section called “Consola App Runner”
- the section called “API de App Runner o AWS CLI”

> **Note**
>
> Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar WAF web. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP.
> Para obtener más información sobre la seguridad de la infraestructura y los grupos de seguridad, incluidas las prácticas recomendadas, consulte los siguientes temas de la Guía del usuario de Amazon VPC: Controle el tráfico de red y Controle el tráfico a sus recursos de AWS mediante grupos de seguridad.

Consola App Runner

Al crear un servicio mediante la consola de App Runner o al actualizar su configuración más adelante, puede optar por configurar el tráfico entrante.

Para configurar el tráfico entrante, elige una de las siguientes opciones.

- Punto final público: para que todos los servicios de Internet puedan acceder a tu servicio. De forma predeterminada, se selecciona el punto final público.
- Punto final privado: para hacer que su servicio App Runner sea accesible únicamente desde una Amazon VPC.

Habilite el punto final privado

Habilite un punto de enlace privado asociándolo al punto de enlace de la interfaz de VPC de la Amazon VPC a la que desea acceder. Puede crear un nuevo punto final de interfaz de VPC o elegir uno existente.

Para crear un punto final de interfaz de VPC

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Vaya a la sección Redes en Configurar el servicio.
3. Elija Punto final privado para el tráfico de red entrante. Se abren las opciones para conectarse a un VCP mediante el punto final de la interfaz de VPC.
4. Elija Crear un nuevo punto final. Se abre el cuadro de diálogo Crear un nuevo punto final de interfaz de VPC.
5. Introduzca un nombre para el punto final de la interfaz de VPC.
6. Elija el punto final de la interfaz de VPC necesario en la lista desplegable disponible.
7. Elija el grupo de seguridad de la lista desplegable. La adición de grupos de seguridad proporciona una capa de seguridad adicional al punto final de la interfaz de VPC. Se recomienda elegir dos o más grupos de seguridad. Si no eliges un grupo de seguridad, App Runner asigna un grupo de seguridad predeterminado al punto final de la interfaz de VPC. Asegúrese de que las reglas del grupo de seguridad no bloqueen los recursos que desean comunicarse con su servicio de App Runner. Las reglas del grupo de seguridad deben permitir que los recursos interactúen con tu servicio de App Runner.

> **Note**
>
> Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar WAF web. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP.
> Para obtener más información sobre la seguridad de la infraestructura y los grupos de seguridad, incluidas las prácticas recomendadas, consulte los siguientes temas de la Guía del usuario de Amazon VPC: Controle el tráfico de red y Controle el tráfico a sus recursos de AWS mediante grupos de seguridad.

8. Elija las subredes necesarias de la lista desplegable. Se recomienda seleccionar al menos dos subredes para cada zona de disponibilidad desde la que accederá al servicio App Runner.

> **Note**
>
> Si está configurando el punto final para doble pila, asegúrese de que su infraestructura y el punto final de VPC admitan el tráfico de doble pila.

9. (Opcional) Seleccione Añadir nueva etiqueta e introduzca la clave de la etiqueta y el valor de la etiqueta.
10. Seleccione Crear. Se abre la página de configuración del servicio, que muestra el mensaje de que se ha creado correctamente el punto final de la interfaz de VPC en la barra superior.

Para elegir un punto final de interfaz de VPC existente

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Vaya a la sección Redes en Configurar el servicio.
3. Elija Punto final privado para el tráfico de red entrante. Se abren las opciones para conectarse a una VPC mediante el punto final de la interfaz de VPC. Se muestra una lista de los puntos finales de la interfaz de VPC disponibles.
4. Elija el punto de enlace de la interfaz de VPC requerido que aparece en los puntos de enlace de la interfaz de VPC.

5. Seleccione Siguiente para crear el servicio. App Runner habilita el punto final privado.

> **Note**
>
> Una vez creado el servicio, puede optar por editar los grupos de seguridad y las subredes asociados al punto final de la interfaz de VPC, si es necesario.

Para comprobar los detalles del punto de conexión privado, vaya a su servicio y amplíe la sección Redes en la pestaña Configuración. Muestra los detalles de la VPC y del punto final de la interfaz de VPC asociados al punto final privado.

Actualizar el punto final de la interfaz de VPC

Una vez creado el servicio App Runner, puede editar el punto final de la interfaz de VPC asociado al punto final privado.

> **Note**
>
> No puede actualizar el nombre del punto final ni los campos de la VPC.

Para actualizar el punto final de la interfaz de VPC

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Ve a tu servicio y selecciona Configuraciones de red en el panel izquierdo.
3. Elija Tráfico entrante para ver los puntos finales de la interfaz de VPC asociados a los servicios respectivos.
4. Elija el punto final de la interfaz de VPC que desee editar.
5. Elija Edit (Edición de). Se abre el cuadro de diálogo para editar el punto final de la interfaz de VPC.
6. Elija los grupos de seguridad y las subredes necesarios y haga clic en Actualizar. La página que muestra los detalles del punto final de la interfaz de VPC se abre con el mensaje de actualización correcta del punto final de la interfaz de VPC en la barra superior.

> **Note**
>
> Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar la web de WAF. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP.
> Para obtener más información sobre la seguridad de la infraestructura y los grupos de seguridad, incluidas las prácticas recomendadas, consulte los siguientes temas de la Guía del usuario de Amazon VPC: Controle el tráfico de red y Controle el tráfico a sus recursos de AWS mediante grupos de seguridad.

Eliminar punto final de la interfaz de VPC

Si no quieres que tu servicio de App Runner sea de acceso privado, puedes configurar el tráfico entrante como público. Al cambiar a público, se elimina el punto de conexión privado, pero no se elimina el punto de conexión de la interfaz de VPC

Para eliminar el punto final de la interfaz de VPC

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Ve a tu servicio y selecciona Configuraciones de red en el panel izquierdo.
3. Elija Tráfico entrante para ver los puntos finales de la interfaz de VPC asociados a los servicios respectivos.

> **Note**
>
> Antes de eliminar un punto final de la interfaz de VPC, quítelo de todos los servicios a los que esté conectado actualizando su servicio.

4. Elija Eliminar.

Si hay servicios conectados al punto final de la interfaz de VPC, recibirá el mensaje No se puede eliminar el punto final de la interfaz de VPC. Si no hay ningún servicio conectado al punto final de la interfaz de VPC, recibirá un mensaje para confirmar la eliminación.

5. Elija Eliminar. Se abre la página de configuraciones de red para el tráfico entrante con el mensaje de que se ha eliminado correctamente el punto final de la interfaz de VPC en la barra superior.

API de App Runner o AWS CLI

Puede implementar una aplicación en App Runner a la que solo se pueda acceder desde una Amazon VPC.

Para obtener información sobre los permisos necesarios para que su servicio sea privado, consultethe section called “Permisos”.

Para crear una conexión de servicio privado a Amazon VPC

1. Cree un punto final de la interfaz de VPC, un AWS PrivateLink recurso, para conectarse a App Runner. Para ello, especifique las subredes y los grupos de seguridad que desee asociar a la aplicación. A continuación, se muestra un ejemplo de cómo crear un punto final de interfaz de VPC.

> **Note**
>
> Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar la web de WAF. ACLs Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados al WAF. Como resultado, las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP.
> Para obtener más información sobre la seguridad de la infraestructura y los grupos de seguridad, incluidas las prácticas recomendadas, consulte los siguientes temas de la Guía del usuario de Amazon VPC: Controle el tráfico de red y Controle el tráfico a sus recursos de AWS mediante grupos de seguridad.

Example

```
aws ec2 create-vpc-endpoint
 --vpc-endpoint-type: Interface
 --service-name: com.amazonaws.us-east-1.apprunner.requests
 --subnets: subnet1, subnet2
 --security-groups: sg1
```

2. Haga referencia al punto final de la interfaz de VPC mediante las acciones de la API CreateServiceo de UpdateServiceApp Runner a través de la CLI. Configure su servicio para que no sea de acceso público. `IsPubliclyAccessible`Configúrelo en el `IngressConfiguration` miembro del `NetworkConfiguration` parámetro. Si lo desea, puede establecer el `IpAddressType` campo en `IPV4` o`DUAL_STACK`. Si no se establece, este valor se establece de forma predeterminada en. IPV4 El siguiente ejemplo hace referencia al punto final de la interfaz de VPC.

   Example

   ```
   aws apprunner create-service
    --network-configuration:
      {
      "IngressConfiguration":
         {
         "IsPubliclyAccessible": False
         },
         "IpAddressType": "IPV4"
      }
    --service-name: com.amazonaws.us-east-1.apprunner.requests
    --source-configuration: <source_configuration>
   ```

3. Ejecute la acción de la `create-vpc-ingress-connection` API para crear el recurso de conexión de entrada de VPC para App Runner y asócielo al punto final de la interfaz de VPC que creó en el paso anterior. Devuelve un nombre de dominio que se utiliza para acceder al servicio en la VPC especificada. El siguiente es un ejemplo de creación de un recurso de conexión de entrada de VPC.

   Example Solicitud

   ```
   aws apprunner create-vpc-ingress-connection
   ```

```
 --service-arn: <apprunner_service_arn>
 --ingress-vpc-configuration: {"VpcId":<vpc_id>, "VpceId": <vpce_id>}
 --vpc-ingress-connection-name: <vic_connection_name>
```

Example Respuesta

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
   "Status": "PENDING_CREATION",
   "AccountId": <connection_owner_id>,
   "DomainName": <domain_name_associated_with_vpce>,
   "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
   "CreatedAt": <date_created>
}
```

Actualizar la conexión de entrada de VPC

Puede actualizar el recurso de conexión de entrada de VPC. La conexión de entrada de la VPC debe estar en uno de los siguientes estados para poder actualizarse:

- DISPONIBLE
- CREACIÓN FALLIDA
- ACTUALIZACIÓN_FALLIDA

El siguiente es un ejemplo de actualización de un recurso de conexión de entrada de VPC.

Example Solicitud

```
aws apprunner update-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Respuesta

```
{
    "VpcIngressConnectionArn":  <vpc_ingress_connection_arn>,
    "VpcIngressConnectionName":  <vic_connection_name>,
    "ServiceArn":  <apprunner_service_arn>,
```

```
    "Status": "FAILED_UPDATE",
    "AccountId":  <connection_owner_id>,
    "DomainName":  <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt":  <date_created>
}
```

Eliminar la conexión de entrada de VPC

Puede eliminar el recurso de conexión de entrada de la VPC si ya no necesita la conexión privada a la Amazon VPC.

Para poder eliminarla, la conexión de entrada de la VPC debe estar en uno de los siguientes estados:

- DISPONIBLE
- ERROR EN LA CREACIÓN
- ACTUALIZACIÓN FALLIDA
- ELIMINACIÓN FALLIDA

A continuación, se muestra un ejemplo de cómo eliminar una conexión de entrada de VPC

Example Solicitud

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Respuesta

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
   "Status": "PENDING_DELETION",
   "AccountId": <connection_owner_id>,
   "DomainName": <domain_name_associated_with_vpce>,
   "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
   "CreatedAt": <date_created>,
   "DeletedAt": <date_deleted>
```

```
}
```

Usa las siguientes acciones de la API de App Runner para administrar el tráfico entrante privado de tu servicio.

- CreateVpcIngressConnection— Cree un nuevo recurso de conexión de entrada de VPC. App Runner requiere este recurso cuando desea asociar su servicio de App Runner a un punto de conexión de Amazon VPC.
- ListVpcIngressConnections— Devuelve una lista de los puntos finales de AWS App Runner VPC Ingress Connection que están asociados a su cuenta. AWS
- DescribeVpcIngressConnection— Devuelve una descripción completa del recurso de AWS App Runner conexión de entrada de VPC.
- UpdateVpcIngressConnection— Actualice el recurso de conexión de entrada de AWS App Runner VPC.
- DeleteVpcIngressConnection— Eliminar un recurso de conexión de entrada de VPC de App Runner que esté asociado al servicio de App Runner.

Para obtener más información sobre el uso de la API de App Runner, consulta la guía de referencia de la API de App Runner.

IPv6 Habilitación del tráfico entrante

Si desea que su servicio reciba el tráfico de red entrante de IPv6 las direcciones, o de ambas IPv4 IPv6 direcciones, elija el tipo de dirección de doble pila para el punto final. Al crear una nueva aplicación, encontrarás esta configuración en la sección Configurar el servicio > Redes. En los siguientes procedimientos se explica cómo habilitar IPv4 o apilar dos elementos (IPv6 y IPv4) mediante la consola de App Runner o la API de App Runner.

Administrar la doble pila para el tráfico entrante

Administre el tipo de direcciones de doble pila para el tráfico entrante mediante uno de los siguientes métodos:

- the section called “Consola de App Runner”
- the section called “API de App Runner o AWS CLI”

> **Note**
>
> Los siguientes procedimientos explican cómo administrar el tipo de dirección de red para el tráfico entrante público. Para obtener información sobre la administración de tipos de IPv4 direcciones o de doble pila para puntos finales privados, consulte. the section called "Administre puntos finales privados"

Consola de App Runner

Puede elegir un tipo de dirección de doble pila para el tráfico de Internet entrante cuando cree un servicio mediante la consola de App Runner o cuando actualice su configuración más adelante.

Para habilitar el tipo de dirección de doble pila

1. Al crear o actualizar un servicio, expanda la sección Redes en Configurar el servicio.
2. Seleccione Punto final público para el tráfico de red entrante. Si selecciona Punto final público, se abre la opción de tipo de dirección IP del punto final.

   Consulte the section called "Administre puntos finales privados" para ver un procedimiento para administrar los tipos de IPv4 direcciones o de doble pila para puntos finales privados.
3. Expanda el tipo de dirección IP del punto final para ver los siguientes tipos de direcciones IP.

   - IPv4
   - Pila doble (IPv4 y IPv6)

   > **Note**
   >
   > Si no expandes el tipo de dirección IP del punto final para hacer una selección, App Runner IPv4 lo asignará como configuración predeterminada.
4. Elija Dual-stack (IPv4 y). IPv6
5. Seleccione Siguiente y, a continuación, Crear e implementar si va a crear un servicio. De lo contrario, elija Guardar cambios si está actualizando un servicio.

   Cuando se implementa el servicio, la aplicación comienza a recibir tráfico de red desde ambos IPv4 IPv6 puntos finales.

Para cambiar el tipo de dirección

1. Siga los pasos para actualizar un servicio y vaya a Redes.
2. Navegue hasta el tipo de dirección IP del punto final en Tráfico de red entrante y seleccione el tipo de dirección requerido.
3. Seleccione Save changes (Guardar cambios). El servicio se actualiza con la selección que hayas seleccionado.

API de App Runner o AWS CLI

Cuando llames a las acciones de la API CreateServiceo a la API de UpdateServiceApp Runner, usa el `IpAddressType` miembro del `NetworkConfiguration` parámetro para especificar el tipo de dirección. Los valores admitidos que puedes especificar son `IPv4` y`DUAL_STACK`. Especifique `DUAL_STACK` si desea que su servicio reciba tráfico de Internet desde IPv4 y desde los IPv6 puntos de conexión. Si no especifica ningún valor para`IpAddressType`, IPv4 se aplicará de forma predeterminada.

> **Note**
>
> Para ver ejemplos de terminales privados, consultethe section called “API de App Runner o AWS CLI”.

A continuación se muestra un ejemplo para crear un servicio con la pila doble como dirección IP. En este ejemplo, se llama a un `input.json` archivo.

Example Solicitud para crear un servicio con soporte de doble pila

```
aws apprunner create-service \
 --cli-input-json file://input.json
```

Example Contenido de **`input.json`**

```
{
  "ServiceName": "example-service",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
      "ImageConfiguration": {
```

```
        "Port": "8000"
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK"
    }
  }
}
```

Example Respuesta

```
{
  "Service": {
    "ServiceName": "example-service",
    "ServiceId": "<service-id>",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/example-
service/<service-id>",
    "ServiceUrl": "1234567890.us-east-2.awsapprunner.com",
    "CreatedAt": "2023-10-16T12:30:51.724000-04:00",
    "UpdatedAt": "2023-10-16T12:30:51.724000-04:00",
    "Status": "OPERATION_IN_PROGRESS",
    "SourceConfiguration": {
      "ImageRepository": {
        "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
        "ImageConfiguration": {
          "Port": "8000"
        },
        "ImageRepositoryType": "ECR_PUBLIC"
      },
      "AutoDeploymentsEnabled": false
    },
    "InstanceConfiguration": {
      "Cpu": "1024",
      "Memory": "2048"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 5,
      "Timeout": 2,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
```

```
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK",
      "EgressConfiguration": {
        "EgressType": "DEFAULT"
      },
      "IngressConfiguration": {
        "IsPubliclyAccessible": true
      }
    }
  },
  "OperationId": "24bd100b1e111ae1a1f0e1115c4f11de"
}
```

Para obtener más información sobre el parámetro de la API, consulte NetworkConfiguration.

Habilitar el acceso a la VPC para el tráfico saliente

De forma predeterminada, su AWS App Runner aplicación puede enviar mensajes a puntos finales públicos. Esto incluye sus propias soluciones y cualquier otro sitio web o servicio web público. Servicios de AWS Su aplicación puede incluso enviar mensajes a puntos finales públicos de aplicaciones que se ejecutan en una VPC desde Amazon Virtual Private Cloud (Amazon VPC). Si no configura una VPC al lanzar el entorno, App Runner usa la VPC predeterminada, que es pública.

Puede optar por lanzar su entorno en una VPC personalizada para personalizar la configuración de red y seguridad del tráfico saliente. Puede habilitar su AWS App Runner servicio para acceder a las aplicaciones que se ejecutan en una VPC privada desde Amazon Virtual Private Cloud (Amazon VPC). Una vez hecho esto, la aplicación podrá conectarse y enviar mensajes a otras aplicaciones alojadas en una Amazon Virtual Private Cloud (Amazon VPC). Algunos ejemplos son una base de datos de Amazon RDS ElastiCache, Amazon y otros servicios privados que se alojan en una VPC privada.

Conector VPC

Para asociar el servicio a una VPC, cree un punto final de VPC desde la consola de App Runner, denominado VPC Connector. Para crear un conector de VPC, especifique la VPC, una o más subredes y, de forma opcional, uno o más grupos de seguridad. Después de configurar un conector de VPC, puede usarlo con uno o más servicios de App Runner.

Latencia única

Si configuras el servicio App Runner con un conector de VPC personalizado para el tráfico saliente, es posible que experimente una latencia de inicio única de dos a cinco minutos. El proceso de inicio espera hasta que el conector de VPC esté listo para conectarse a otros recursos antes de establecer el estado del servicio en En ejecución. Puede configurar un servicio con un conector de VPC personalizado al crearlo por primera vez, o puede hacerlo posteriormente mediante una actualización del servicio.

Tenga en cuenta que si reutiliza la misma configuración de conector de VPC para otro servicio, no habrá latencia. La configuración del conector de VPC se basa en la combinación de subred y grupo de seguridad. Para una configuración de conector de VPC determinada, la latencia solo se produce una vez, durante la creación inicial del hiperplano del conector de VPC ENIs (interfaces de red elásticas).

Más información sobre los conectores de VPC personalizados y Hyperplane AWS

Los conectores de VPC de App Runner se basan en AWS Hyperplane, el sistema de red interno de Amazon que está detrás de varios AWS recursos, como Network Load Balancer, NAT Gateway y AWS. PrivateLink La tecnología AWS Hyperplane ofrece capacidades de alto rendimiento y baja latencia, además de un mayor grado de intercambio. Se crea un ENI de Hyperplane en las subredes al crear un conector de VPC y asociarlo a su servicio. La configuración de un conector de VPC se basa en una combinación de grupo de seguridad y subred, y puedes hacer referencia al mismo conector de VPC en varios servicios de App Runner. Como resultado, el Hyperplane subyacente ENIs se comparte entre los servicios de App Runner. Este uso compartido es posible, incluso si se amplía la cantidad de tareas necesarias para gestionar la carga de solicitudes, y se traduce en una utilización más eficiente del espacio IP de la VPC. Para obtener más información, consulte Análisis detallado sobre las redes de VPC de AWS App Runner en el blog sobre contenedores de AWS.

Subred

Cada subred se encuentra en una zona de disponibilidad específica. Para una alta disponibilidad, le recomendamos que seleccione subredes en al menos tres zonas de disponibilidad. Si la región tiene menos de tres zonas de disponibilidad, le recomendamos que seleccione sus subredes en todas las zonas de disponibilidad compatibles.

Al seleccionar una subred para su VPC, asegúrese de elegir una subred privada, no una subred pública. Esto se debe a que, al crear un conector de VPC, el servicio App Runner crea una ENI de Hyperplane en cada una de las subredes. A cada ENI de Hyperplane se le asigna solo una dirección IP privada y se etiqueta con una etiqueta de la clave. AWSAppRunnerManaged Si eliges una subred pública, se producirán errores al ejecutar el servicio App Runner. Sin embargo, si su servicio necesita acceder a algunos servicios que están en Internet o a otros públicos Servicios de AWS, consultethe section called “Consideraciones a la hora de seleccionar una subred”.

Consideraciones a la hora de seleccionar una subred

- Cuando conectas tu servicio a una VPC, el tráfico saliente no tiene acceso a la Internet pública. Todo el tráfico saliente de la aplicación se dirige a través de la VPC a la que está conectado el servicio. Todas las reglas de red de la VPC se aplican al tráfico saliente de la aplicación. Esto significa que sus servicios no pueden acceder a la Internet pública y. AWS APIs Para acceder, realiza una de las siguientes acciones:
  - Conecte las subredes a Internet a través de una puerta de enlace NAT.
  - Configure los puntos finales de VPC para los Servicios de AWS que desee acceder. Su servicio permanece dentro de la Amazon VPC mediante el uso de. AWS PrivateLink
- Algunas zonas de disponibilidad Regiones de AWS no admiten las subredes que se pueden usar con los servicios de App Runner. Si eliges subredes en estas zonas de disponibilidad, tu servicio no se podrá crear ni actualizar. En estas situaciones, App Runner proporciona un mensaje de error detallado que indica las subredes y zonas de disponibilidad no compatibles. Cuando esto ocurra, solucione el problema eliminando las subredes no compatibles de su solicitud e inténtelo de nuevo.
- Todas las subredes que seleccione deben tener el mismo tipo de dirección IP, ya sea de doble pila o de doble pila. IPv4

Security group (Grupo de seguridad)

Si lo desea, puede especificar los grupos de seguridad a los que App Runner puede acceder AWS en las subredes especificadas. Si no especificas grupos de seguridad, App Runner usa el grupo de

seguridad predeterminado de la VPC. El grupo de seguridad predeterminado permite todo el tráfico de salida.

La adición de un grupo de seguridad proporciona una capa de seguridad adicional a los conectores de VPC, lo que le proporciona un mayor control sobre el tráfico de la red. El conector de VPC solo se usa para la comunicación saliente desde su aplicación. Las reglas de salida se utilizan para permitir la comunicación con los puntos finales de destino deseados. También debe asegurarse de que todos los grupos de seguridad que estén asociados al recurso de destino cuenten con las reglas de entrada adecuadas. De lo contrario, estos recursos no pueden aceptar el tráfico que proviene de los grupos de seguridad del conector de VPC.

> **Note**
>
> Cuando asocias tu servicio a una VPC, el siguiente tráfico no se ve afectado:
>
> - Tráfico entrante: los mensajes entrantes que recibe la aplicación no se ven afectados por una VPC asociada. Los mensajes se enrutan a través del nombre de dominio público asociado a tu servicio y no interactúan con la VPC.
> - Tráfico de App Runner: App Runner gestiona varias acciones en tu nombre, como extraer el código fuente y las imágenes, enviar registros y recuperar información confidencial. El tráfico que generan estas acciones no se enruta a través de la VPC.

Para obtener más información sobre cómo AWS App Runner se integra con Amazon VPC, consulte Redes de VPC de AWS App Runner.

Administrar el acceso a la VPC

> **Note**
>
> Si crea un conector de VPC de tráfico saliente para un servicio, el proceso de inicio del servicio que sigue experimentará una latencia única. Puedes establecer esta configuración para un servicio nuevo al crearlo o, posteriormente, con una actualización del servicio. Para obtener más información, consulte Latencia única el capítulo Redes con App Runner de esta guía.

Administre el acceso a la VPC para sus servicios de App Runner mediante uno de los siguientes métodos:

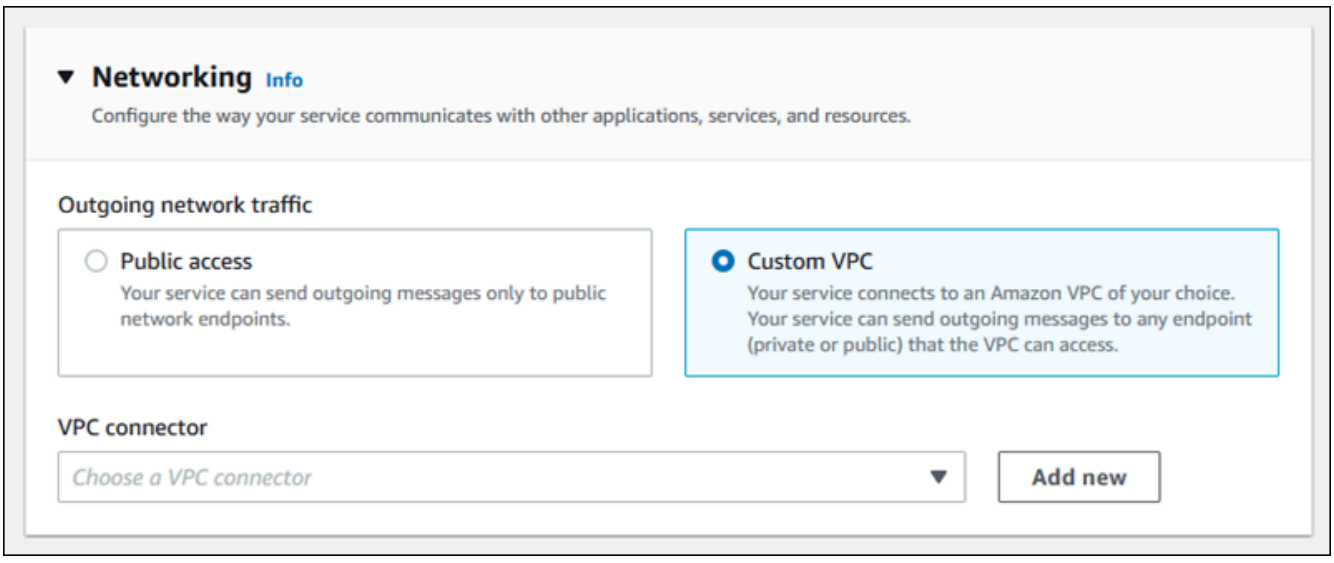
App Runner console

Al crear un servicio mediante la consola de App Runner o al actualizar su configuración más adelante, puede optar por configurar el tráfico saliente. Busca la sección de configuración de redes en la página de la consola. Para el tráfico de red saliente, elija una de las siguientes opciones:

- Acceso público: para asociar su servicio a puntos finales públicos u otros Servicios de AWS.
- VPC personalizada: para asociar su servicio a una VPC de Amazon VPC. Su aplicación puede conectarse y enviar mensajes a otras aplicaciones alojadas en una Amazon VPC.

Para habilitar la VPC personalizada

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Vaya a la sección Redes en Configurar el servicio.

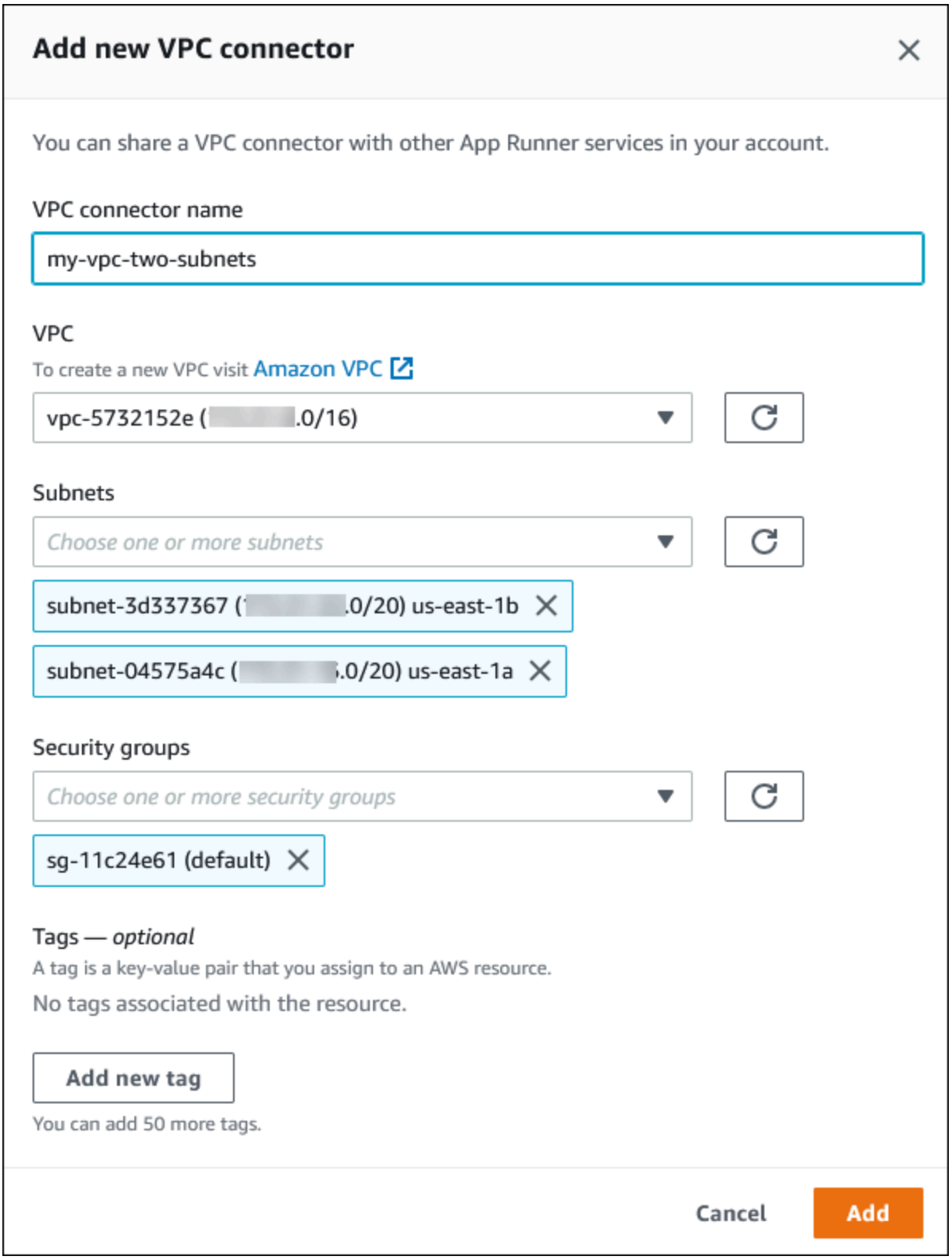


3. Elija VPC personalizada para el tráfico de red saliente.
4. En el panel de navegación, elija el conector VPC.

   Si creó los conectores de VPC, la consola mostrará una lista de los conectores de VPC de su cuenta. Puede elegir un conector de VPC existente y elegir Siguiente para revisar la configuración. A continuación, vaya al último paso. Como alternativa, puede añadir un nuevo conector de VPC mediante los siguientes pasos.

5. Elija Añadir nuevo para crear un nuevo conector de VPC para su servicio.

   A continuación, se abre el cuadro de diálogo Añadir nuevo conector de VPC.



6. Introduzca un nombre para el conector de VPC y seleccione la VPC necesaria de la lista disponible.
7. En el caso de las subredes, seleccione una subred para cada zona de disponibilidad desde la que vaya a acceder al servicio App Runner. Para obtener una mejor disponibilidad, elija tres subredes. O bien, si hay menos de tres subredes, elija todas las subredes disponibles.

   > **Note**
   >
   > - Asegúrese de asignar subredes privadas al conector de VPC. Si asigna subredes públicas al conector de VPC, su servicio no se crea o se revierte automáticamente durante una actualización.
   > - Si el tráfico saliente es de doble pila, asegúrese de que todas las subredes que seleccione estén configuradas para doble pila en la consola de VPC.

8. (Opcional) En el grupo de seguridad, selecciona los grupos de seguridad que deseas asociar a las interfaces de red de los puntos finales.
9. (Opcional) Para agregar una etiqueta, elija Agregar etiqueta nueva e ingrese la clave y el valor de la etiqueta.
10. Elija Añadir.

    Los detalles del conector de VPC que ha creado aparecen en Conector de VPC.
11. Seleccione Siguiente para revisar la configuración y, a continuación, elija Crear e implementar.

    App Runner crea un recurso de conector de VPC para usted y, a continuación, lo asocia a su servicio. Si el servicio se ha creado correctamente, la consola muestra el panel de control del servicio, con una descripción general del nuevo servicio.

App Runner API or AWS CLI

Cuando llames a las acciones de la API CreateServiceo de UpdateServiceApp Runner, usa el `EgressConfiguration` miembro del `NetworkConfiguration` parámetro para especificar un recurso de conector de VPC para tu servicio.

Use las siguientes acciones de la API de App Runner para administrar los recursos del conector de VPC.

- CreateVpcConnector— Crea un nuevo conector de VPC.
- ListVpcConnectors— Devuelve una lista de los conectores de VPC que están asociados a su. Cuenta de AWS La lista incluye descripciones completas.
- DescribeVpcConnector— Devuelve una descripción completa de un conector de VPC.
- DeleteVpcConnector— Elimina un conector de VPC. Si alcanza la cuota de conectores de VPC para usted Cuenta de AWS, es posible que tenga que eliminar los conectores de VPC innecesarios.

Para implementar una aplicación en App Runner que tenga acceso saliente a una VPC, primero debe crear un conector de VPC. Para ello, especifique una o más subredes y grupos de seguridad para asociarlos a la aplicación. A continuación, puede hacer referencia al conector de VPC en la CLI Create o UpdateServicea través de ella, como se muestra en el siguiente ejemplo:

```
            cat > vpc-connector.json <<EOF
{
"VpcConnectorName": "my-vpc-connector",
"Subnets": [
"subnet-a",
"subnet-b",
"subnet-c"
],
"SecurityGroups": [
"sg-1",
"sg-2"
]
}
EOF

aws apprunner create-vpc-connector \
--cli-input-json file:///vpc-connector.json

cat > service.json <<EOF

{
"ServiceName": "my-vpc-connected-service",
"SourceConfiguration": {
"ImageRepository": {
"ImageIdentifier": "<ecr-image-identifier> ",
"ImageConfiguration": {
```

```
"Port": "8000"
},
"ImageRepositoryType": "ECR"
}
},
"NetworkConfiguration": {
"EgressConfiguration": {
"EgressType": "VPC",
"VpcConnectorArn": "arn:aws:apprunner:..../my-vpc-connector"
}
}
}
EOF

aws apprunner create-service \
--cli-input-json file:///service.js
```

Observabilidad de tu servicio App Runner

AWS App Runner se integra con varios AWS servicios para proporcionarte un amplio conjunto de herramientas de observabilidad para tu servicio App Runner. Los temas de este capítulo describen estas capacidades.

Temas

Seguimiento de la actividad del servicio App Runner

AWS App Runner usa una lista de operaciones para realizar un seguimiento de la actividad en tu servicio App Runner. Una operación representa una llamada asíncrona a una acción de la API, como crear un servicio, actualizar una configuración e implementar un servicio. En las siguientes secciones, se muestra cómo realizar un seguimiento de la actividad en la consola de App Runner y cómo utilizar la API.

Realice un seguimiento de la actividad del servicio App Runner

Realice un seguimiento de la actividad del servicio de App Runner mediante uno de los siguientes métodos:

App Runner console

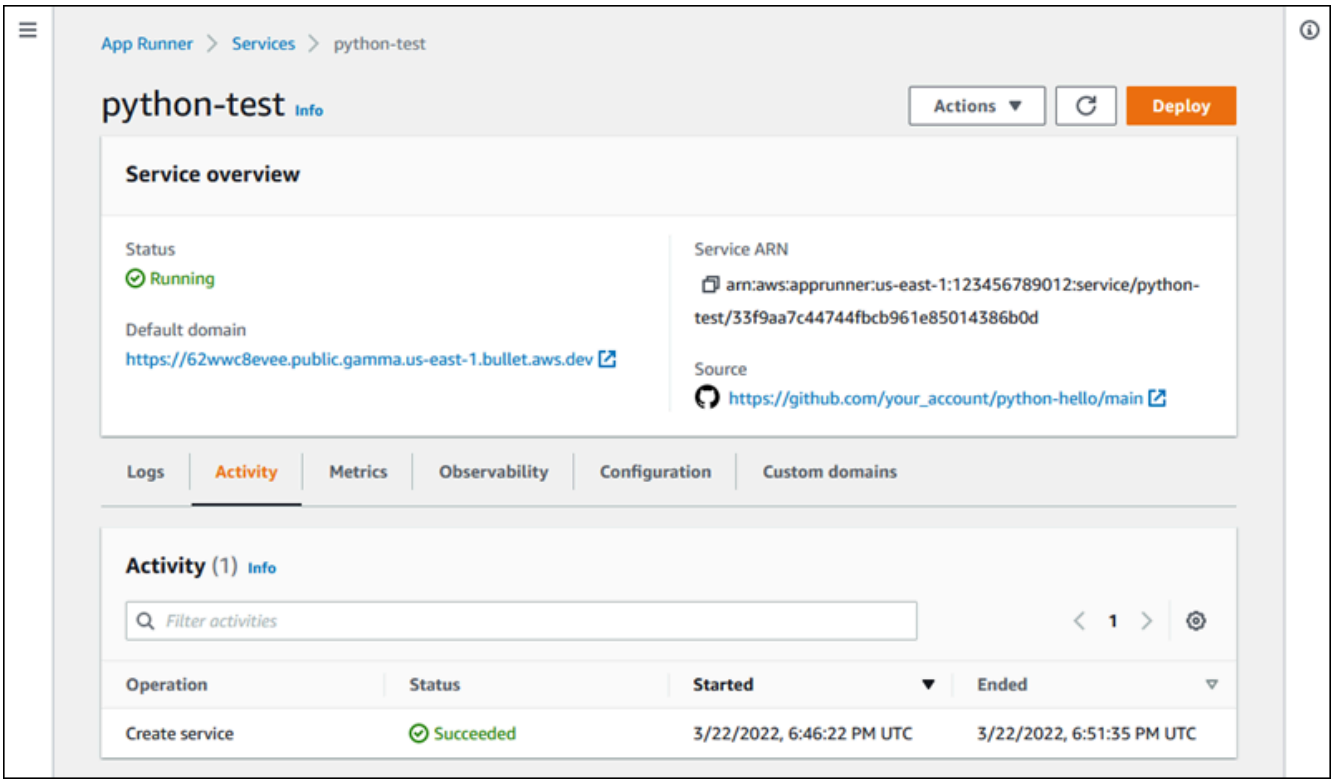
La consola de App Runner muestra la actividad del servicio de App Runner y ofrece más formas de explorar las operaciones.

Para ver la actividad de su servicio

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.

2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, selecciona la pestaña Actividad, si aún no está seleccionada.

   La consola muestra una lista de operaciones.

4. Para buscar operaciones específicas, introduzca un término de búsqueda hacia abajo en la lista. Puede buscar cualquier valor que aparezca en la tabla.

5. Elija cualquier operación de la lista para ver o descargar el registro relacionado.

App Runner API or AWS CLI

La ListOperationsacción, con el nombre de recurso de Amazon (ARN) de un servicio de App Runner, devuelve una lista de las operaciones que se produjeron en este servicio. Cada elemento de la lista contiene un identificador de operación y algunos detalles de seguimiento.

Ver los registros de App Runner transmitidos a CloudWatch Logs

Puede usar Amazon CloudWatch Logs para monitorear, almacenar y acceder a los archivos de registro que generan sus recursos en varios AWS servicios. Para obtener más información, consulta la Guía del usuario de Amazon CloudWatch Logs.

AWS App Runner recopila el resultado de las implementaciones de sus aplicaciones y de su servicio activo y lo transmite a CloudWatch Logs. En las siguientes secciones, se enumeran los flujos de registro de App Runner y se muestra cómo verlos en la consola de App Runner.

Grupos de registros y transmisiones de App Runner

CloudWatch Logs mantiene los datos de registro en flujos de registro que luego organiza en grupos de registros. Un flujo de registro es una secuencia de eventos de registro de una fuente específica. Un grupo de registro es un grupo de flujos de registro que comparten la misma configuración de retención, monitorización y control de acceso.

App Runner define dos grupos de CloudWatch registros, cada uno con varios flujos de registro, para cada servicio de App Runner de su empresa Cuenta de AWS.

Registros de servicio

El grupo de registros de servicios contiene los datos de registro generados por App Runner mientras administra el servicio de App Runner y actúa en consecuencia.

Nombre del grupo de registro	Ejemplo
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/service`

Dentro del grupo de registros de servicios, App Runner crea un flujo de registro de eventos para capturar la actividad durante el ciclo de vida del servicio de App Runner. Por ejemplo, esto podría estar iniciando la aplicación o pausándola.

Además, App Runner crea un flujo de registro para cada operación asíncrona de larga duración relacionada con tu servicio. El nombre del flujo de registro refleja el tipo de operación y el identificador específico de la operación.

Un despliegue es un tipo de operación. Los registros de implementación contienen el resultado del registro de los pasos de creación e implementación que App Runner lleva a cabo al crear un servicio o implementar una nueva versión de la aplicación. Los nombres de los flujos de registro de implementación comienzan y terminan con el ID de la operación que realiza la implementación. `deployment/` Esta operación es una CreateServicellamada para la implementación inicial de la aplicación o una StartDeploymentllamada para cada implementación posterior.

Dentro de un registro de despliegue, cada mensaje de registro comienza con un prefijo:

- `[AppRunner]`— El resultado que App Runner genera durante la implementación.
- `[Build]`— Salida de tus propios scripts de compilación.

Nombre del flujo de registro	Ejemplo
`events`	N/A (nombre fijo)
`operation-type` `/`*`operation-id`*	`deployment/c2c8eeedea164f45 9cf78f12a8953390`

Registros de aplicaciones

El grupo de registros de aplicaciones contiene el resultado del código de la aplicación en ejecución.

Nombre del grupo de registro	Ejemplo
`/aws/apprunner/` *`service-n ame`* `/`*`service-id`* `/application`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application`

Dentro del grupo de registros de aplicaciones, App Runner crea un flujo de registro para cada instancia (unidad de escalado) en la que se ejecuta la aplicación.

Nombre del flujo de registro	Ejemplo
`instance/` *`instance-id`*	`instance/1a80bc9134a84699b7 b3432ebeebb591`

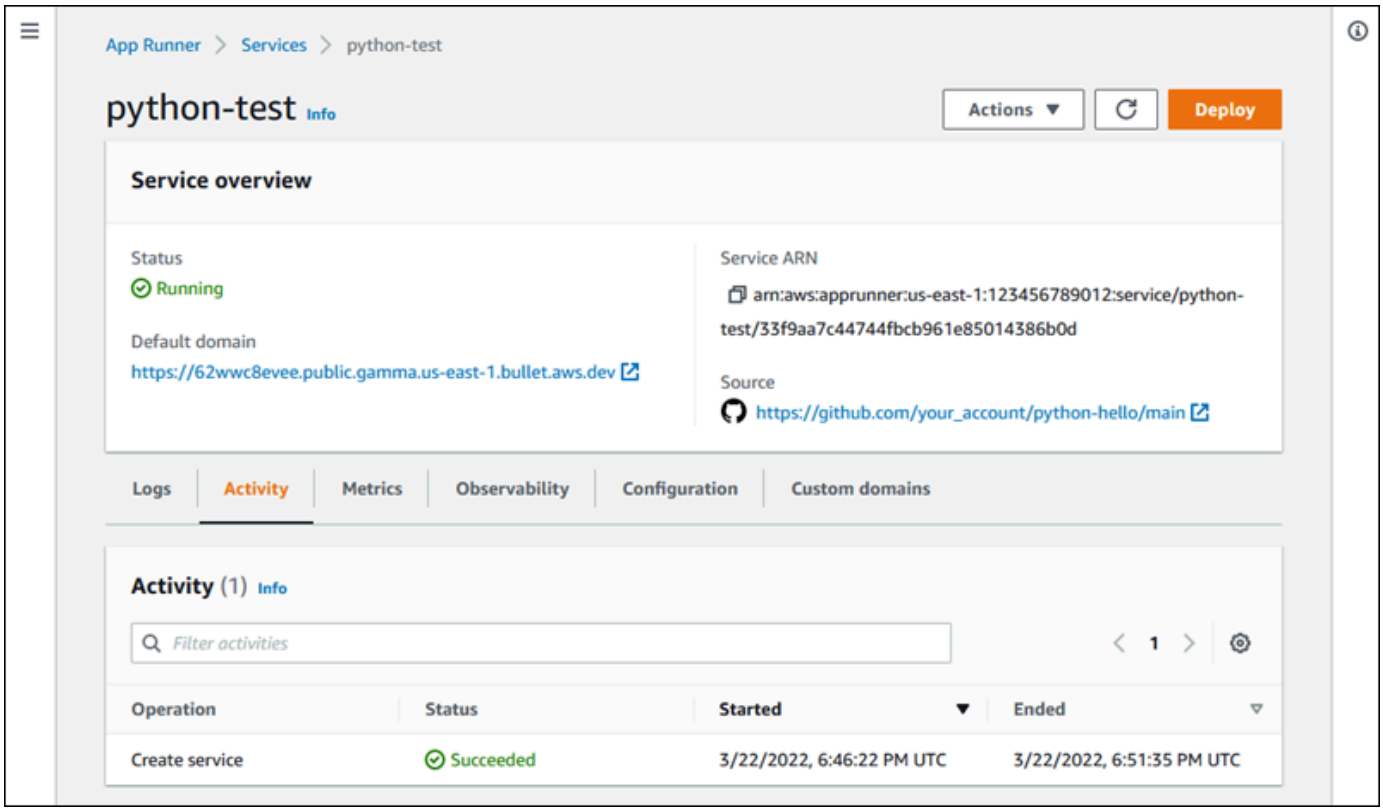
Ver los registros de App Runner en la consola

La consola de App Runner muestra un resumen de todos los registros de tu servicio y te permite verlos, explorarlos y descargarlos.

Para ver los registros de su servicio

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.

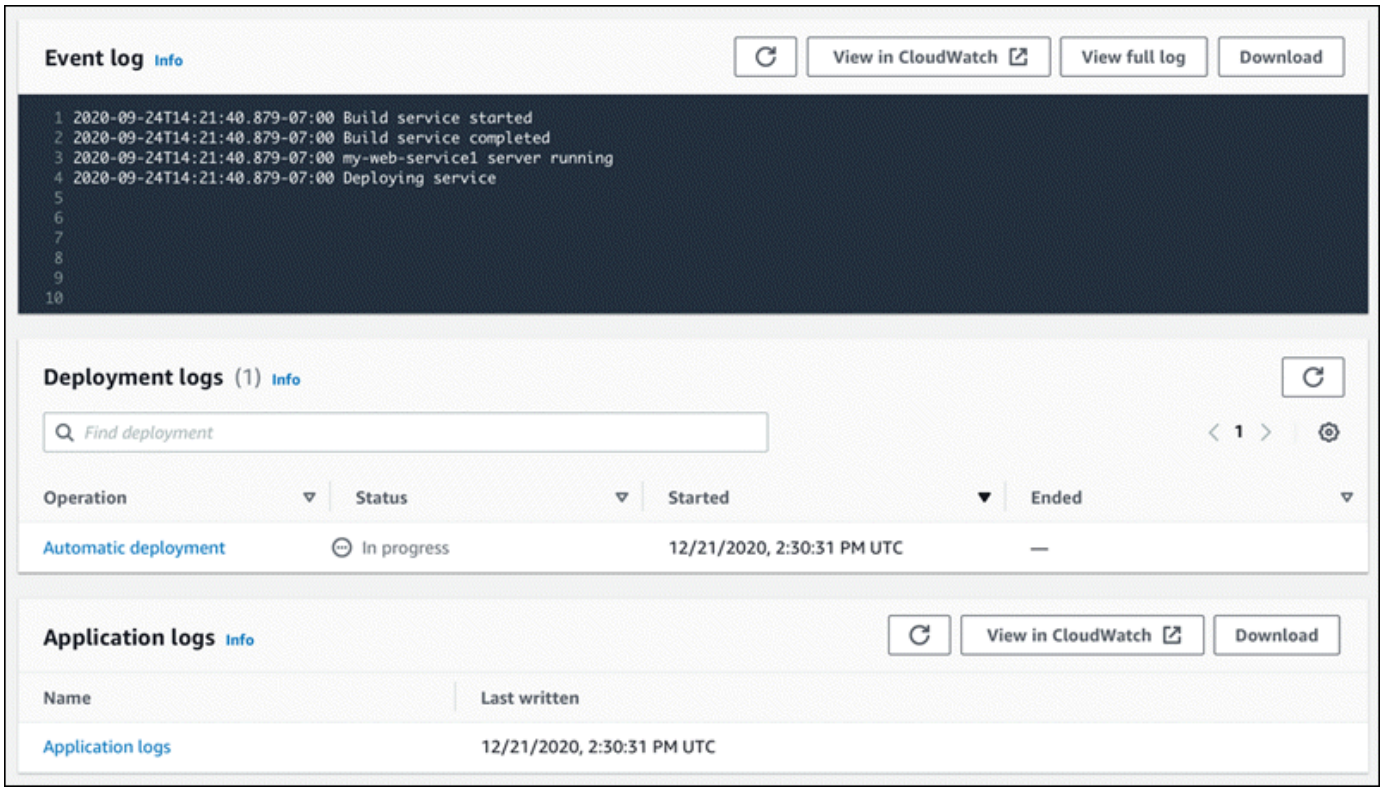


3. En la página del panel de servicios, selecciona la pestaña Registros.

   La consola muestra algunos tipos de registros en varias secciones:

   - Registro de eventos: actividad del ciclo de vida del servicio App Runner. La consola muestra los eventos más recientes.
   - Registros de implementación: orienta las implementaciones del repositorio a tu servicio de App Runner. La consola muestra un flujo de registros independiente para cada implementación.

- Registros de aplicaciones: el resultado de la aplicación web que se implementó en el servicio App Runner. La consola combina los resultados de todas las instancias en ejecución en un único flujo de registro.



4. Para encontrar despliegues específicos, consulta la lista de registros de despliegues introduciendo un término de búsqueda. Puede buscar cualquier valor que aparezca en la tabla.
5. Para ver el contenido de un registro, elija Ver registro completo (registro de eventos) o el nombre del flujo de registro (registros de implementación y aplicación).
6. Seleccione Descargar para descargar un registro. Para un flujo de registro de despliegue, seleccione primero un flujo de registro.
7. Selecciona Ver en CloudWatch para abrir la CloudWatch consola y utilizar todas sus funciones para explorar los registros de servicio de App Runner. Para un flujo de registro de implementación, primero seleccione un flujo de registro.

> **Note**
>
> La CloudWatch consola resulta especialmente útil si desea ver los registros de aplicaciones de instancias específicas en lugar del registro de aplicaciones combinado.

Ver las métricas de servicio de App Runner notificadas a CloudWatch

Amazon CloudWatch supervisa los recursos de Amazon Web Services (AWS) y las aplicaciones en las que se ejecuta AWS en tiempo real. Puede utilizarlas CloudWatch para recopilar y realizar un seguimiento de las métricas, que son variables que puede medir para sus recursos y aplicaciones. También puede utilizarla para crear alarmas que controlen las métricas. Cuando se alcanza un determinado umbral, CloudWatch envía notificaciones o realiza cambios automáticamente en los recursos monitoreados. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

AWS App Runner recopila una variedad de métricas que te proporcionan una mayor visibilidad del uso, el rendimiento y la disponibilidad de tus servicios de App Runner. Algunas métricas rastrean las instancias individuales que ejecutan tu servicio web, mientras que otras se refieren al nivel de servicio general. En las siguientes secciones, se enumeran las métricas de App Runner y se muestra cómo verlas en la consola de App Runner.

Métricas de App Runner

App Runner recopila las siguientes métricas relacionadas con tu servicio y las publica CloudWatch en el espacio de `AWS/AppRunner` nombres.

> **Note**
>
> Antes del 23 de agosto de 2023, las métricas de utilización de la CPU y de la memoria se basaban en las unidades de vCPU y los megabytes de memoria utilizados, en lugar del porcentaje de utilización, tal como se calcula en la actualidad. Si tu aplicación se ejecutó en App Runner antes de esta fecha y decides volver a ver las métricas de esa fecha en App Runner o en la CloudWatch consola, verás una pantalla con las métricas de ambas unidades y, como resultado, también verás algunas irregularidades.

> **Important**
>
> Deberás actualizar CloudWatch las alarmas que se basen en los valores de las métricas de uso de la CPU y de la memoria antes del 23 de agosto de 2023. Actualice las alarmas para

que se activen en función del porcentaje de utilización en lugar de la vCPU o los megabytes. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

Las métricas a nivel de instancia se recopilan para cada instancia (unidad de escalado) de forma individual.

¿Qué se mide?	Métrica	Descripción
CPU utilization	`CPUUtilization`	El porcentaje de uso medio de la CPU durante períodos de un minuto respecto del uso total de la CPU reservado por la configuración del servicio.
Memory utilization	`MemoryUtilization`	El porcentaje de uso medio de memoria durante períodos de un minuto sobre la memoria total reservada por la configuración del servicio.

Las métricas del nivel de servicio se recopilan para todo el servicio.

¿Qué se mide?	Métrica	Descripción
CPU utilization	`CPUUtilization`	El porcentaje de uso total de la CPU en todas las instancias durante períodos de un minuto respecto del uso total de la CPU reservado por la configuración del servicio.
Memory utilization	`MemoryUtilization`	El porcentaje de uso de memoria agregado en todas las instancias durante períodos de un minuto sobre la memoria total reservada por la configuración del servicio.
Concurrency	`Concurrency`	El número aproximado de solicitudes simultáneas que gestiona el servicio.
HTTP request count	`Requests`	El número de solicitudes HTTP que ha recibido el servicio.

¿Qué se mide?	Métrica	Descripción

HTTP status counts	`2xxStatus Responses`
	
	`4xxStatus Responses`
	
	`5xxStatus Responses`
		El número de solicitudes HTTP que devolvier on cada estado de respuesta, agrupadas por categoría (2XX, 4XX, 5XX).

HTTP request latency	`RequestLatency`	El tiempo, en milisegundos, que tardó su servicio web en procesar las solicitudes HTTP.
Instance counts	`ActiveInstances`	El número de instancias que procesan las solicitud es HTTP para tu servicio. **Note** Si la `ActiveInstances` métrica muestra cero, significa que no hay solicitud es para el servicio. No indica que el número de instancias del servicio sea cero.

Ver las métricas de App Runner en la consola

La consola de App Runner muestra gráficamente las métricas que App Runner recopila para tu servicio y ofrece más formas de explorarlas.

Note

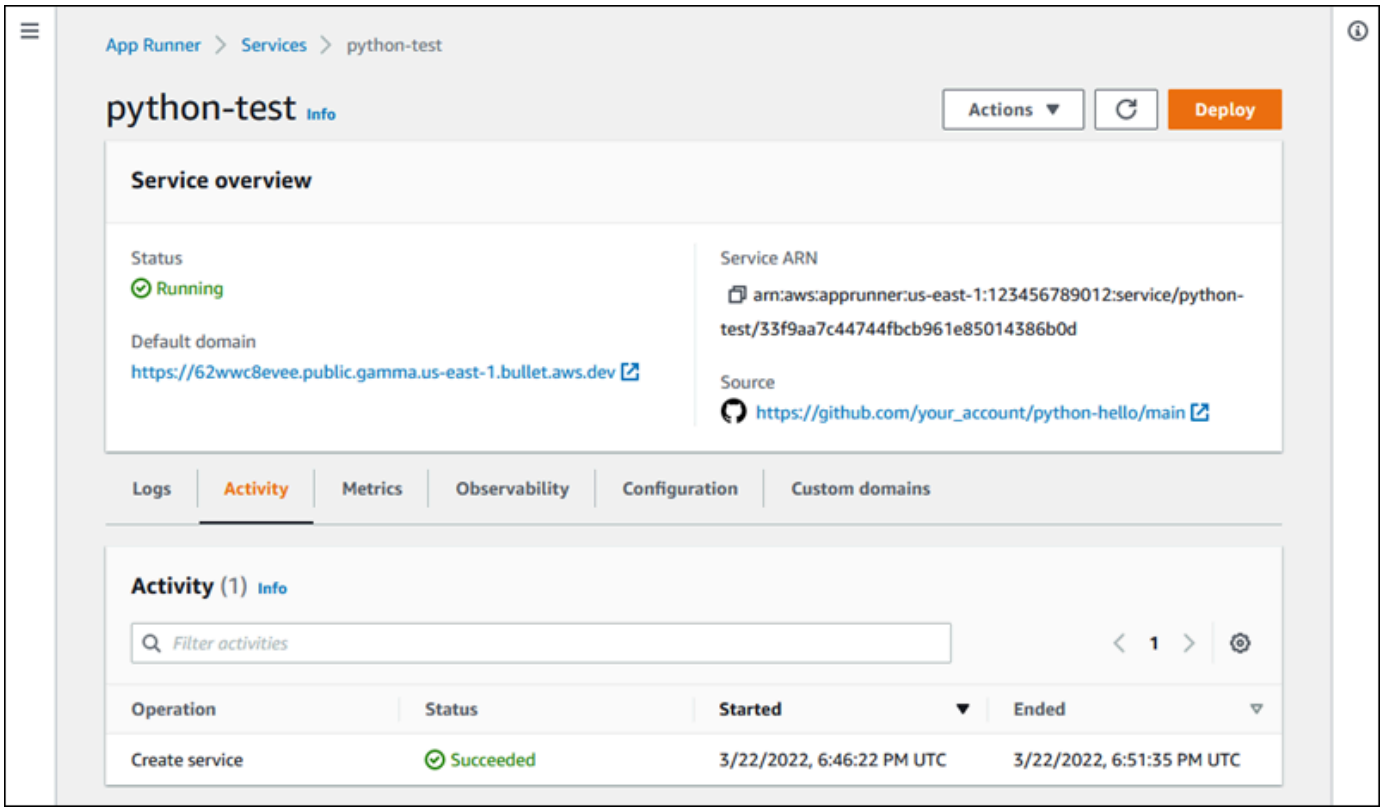
En este momento, la consola solo muestra las métricas de servicio. Para ver las métricas de la instancia, usa la CloudWatch consola.

Para ver los registros de tu servicio

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.

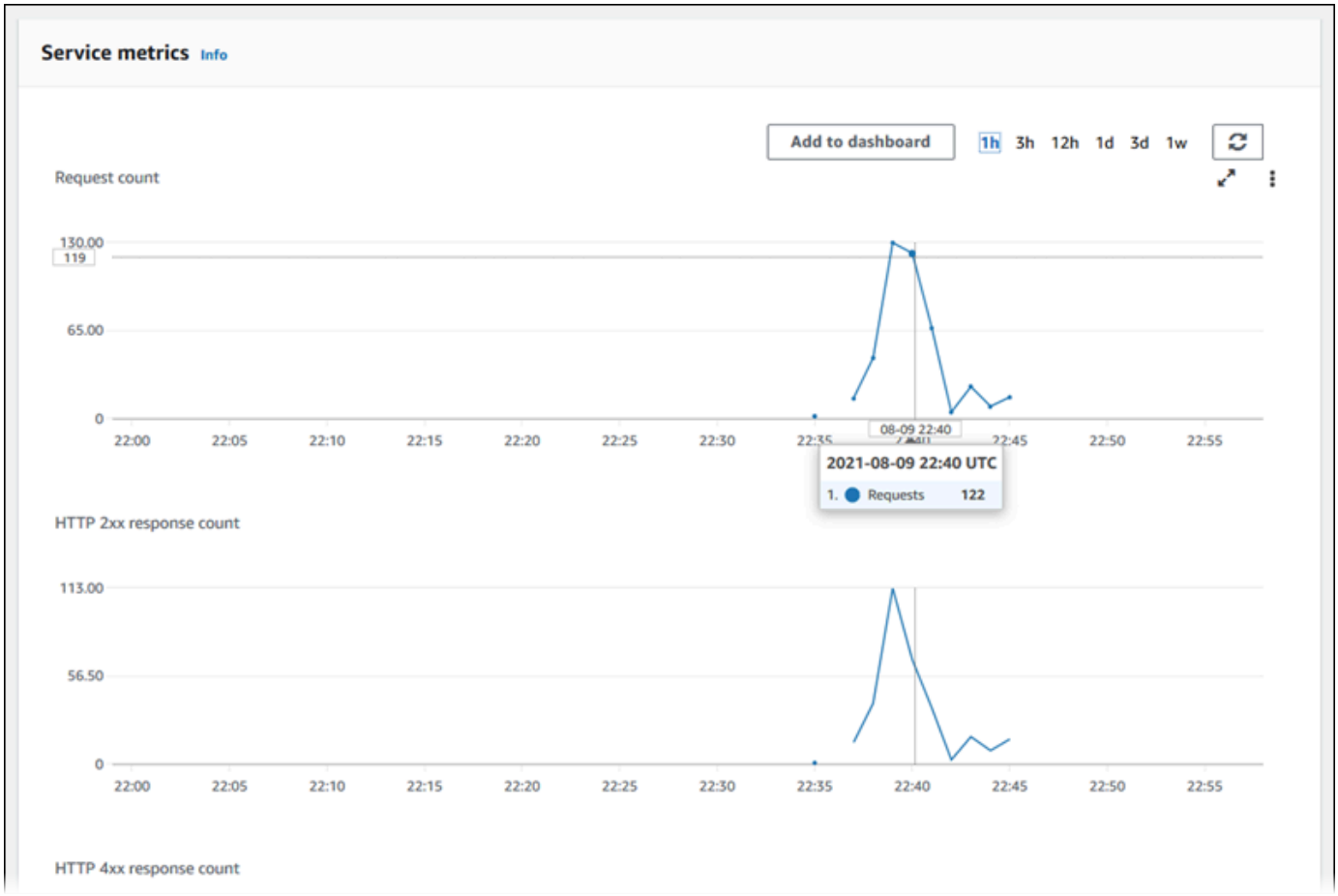
2. En el panel de navegación, selecciona Servicios y, a continuación, selecciona tu servicio de App Runner.

   La consola muestra el panel de servicios con una descripción general del servicio.



3. En la página del panel de servicios, selecciona la pestaña Métricas.

   La consola muestra un conjunto de gráficos de métricas.



4. Elija una duración (por ejemplo, 12 horas) para ajustar los gráficos de métricas al período reciente de esa duración.
5. Seleccione Añadir al panel de control en la parte superior de una de las secciones del gráfico o utilice el menú de cualquier gráfico para añadir las métricas pertinentes a un panel de control de la CloudWatch consola y poder investigarlo más a fondo.

Gestión de eventos de App Runner en EventBridge

Con Amazon EventBridge, puedes configurar reglas basadas en eventos que supervisen un flujo de datos en tiempo real de tu AWS App Runner servicio para detectar determinados patrones. Cuando coincide el patrón de una regla, EventBridge inicia una acción en un objetivo AWS Lambda, como Amazon ECS y Amazon SNS. AWS Batch Por ejemplo, puede establecer una regla para el envío de notificaciones por correo electrónico mediante la señalización de un tema de Amazon SNS cada vez que falle una implementación en su servicio. O bien, puedes configurar una función Lambda para que notifique a un canal de Slack cada vez que se produzca un error en la actualización de un

servicio. Para obtener más información EventBridge, consulta la Guía EventBridge del usuario de Amazon.

App Runner envía los siguientes tipos de eventos a EventBridge

- Cambio de estado del servicio: cambio en el estado de un servicio de App Runner. Por ejemplo, el estado de un servicio cambió a`DELETE_FAILED`.
- Cambio en el estado de la operación del servicio: cambio en el estado de una operación prolongada y asíncrona en un servicio de App Runner. Por ejemplo, se ha empezado a crear un servicio, se ha completado correctamente una actualización de servicio o se ha completado la implementación de un servicio con errores.

Crear una EventBridge regla para actuar en función de los eventos de App Runner

Un EventBridge evento es un objeto que define algunos EventBridge campos estándar, como el AWS servicio de origen y el tipo de detalle (evento), y un conjunto de campos específico del evento con los detalles del evento. Para crear una EventBridge regla, se utiliza la EventBridge consola para definir un patrón de eventos (qué eventos se deben rastrear) y especificar una acción objetivo (qué se debe hacer en un partido). Un patrón de eventos es similar a los eventos con los que coincide. Se especifica un subconjunto de campos para que coincidan y, para cada campo, se especifica una lista de valores posibles. En este tema se proporcionan ejemplos de eventos y patrones de eventos de App Runner.

Para obtener más información sobre la creación de EventBridge reglas, consulta Cómo crear una regla para un AWS servicio en la Guía del EventBridge usuario de Amazon.

> **Note**
>
> Algunos servicios admiten patrones predefinidos en EventBridge. Esto simplifica la creación de un patrón de eventos. Usted selecciona los valores de los campos en un formulario y EventBridge genera el patrón automáticamente. En este momento, App Runner no admite patrones predefinidos. Tienes que introducir el patrón como un objeto JSON. Puede utilizar los ejemplos de este tema como punto de partida.

Ejemplos de eventos de App Runner

Estos son algunos ejemplos de eventos a los que App Runner envía EventBridge.

- Un evento de cambio de estado del servicio. En concreto, un servicio que ha cambiado de `RUNNING` estado `OPERATION_IN_PROGRESS` a estado.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T11:54:23Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "previousServiceStatus": "OPERATION_IN_PROGRESS",
    "currentServiceStatus": "RUNNING",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service status is set to RUNNING.",
    "severity": "INFO"
  }
}
```

- Un evento de cambio de estado de operación. En concreto, una `UpdateService` operación que se completó satisfactoriamente.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Operation Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T18:43:48Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
```

```
  ],
  "detail": {
    "operationStatus": "UpdateServiceCompletedSuccessfully",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service update completed successfully. New application and
 configuration is deployed.",
    "severity": "INFO"
  }
}
```

Ejemplos de patrones de eventos de App Runner

Los siguientes ejemplos muestran los patrones de eventos que puedes usar en EventBridge las reglas para hacer coincidir uno o más eventos de App Runner. Un patrón de eventos es similar a un evento. Incluya solo los campos que desee que coincidan y proporcione una lista en lugar de un escalar para cada uno de ellos.

- Haga coincidir todos los eventos de cambio de estado del servicio para los servicios de una cuenta específica, cuando el servicio ya no esté en `RUNNING` estado.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- Haga coincidir todos los eventos de cambio de estado de la operación para los servicios de una cuenta específica en los que la operación falló.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
```

```
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```

Referencia de eventos de App Runner

Cambio de estado del servicio

Un evento de cambio de estado del servicio se ha `detail-type` establecido en`AppRunner Service Status Change`. Tiene los siguientes campos y valores de detalle:

```
"serviceId": "your service ID",
"serviceName": "your service name",
"message": "Service status is set to CurrentStatus.",
"previousServiceStatus": "any valid service status",
"currentServiceStatus": "any valid service status",
"severity": "varies"
```

Cambio de estado de la operación

Se ha `detail-type` establecido un evento de cambio de estado de operación en`AppRunner Service Operation Status Change`. Tiene los siguientes campos y valores de detalle:

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

En la siguiente tabla se enumeran todos los códigos de estado posibles y los mensajes relacionados.

Status	Mensaje
`CreateServiceStarted`	Se ha iniciado la creación del servicio.

Status	Mensaje
`CreateServiceCompletedSuccessfully`	La creación del servicio se completó correctamente.
`CreateServiceFailed`	No se pudo crear el servicio. Para obtener más información, consulte los registros de servicio.
`DeleteServiceStarted`	Se ha iniciado la eliminación del servicio.
`DeleteServiceCompletedSuccessfully`	La eliminación del servicio se completó correctamente.
`DeleteServiceFailed`	No se pudo eliminar el servicio.
`UpdateServiceStarted`	
`UpdateServiceCompletedSuccessfully`	La actualización del servicio se completó correctamente. Se han implementado la nueva aplicación y configuración.
	La actualización del servicio se completó correctamente. Se ha implementado la nueva configuración.
`UpdateServiceFailed`	Error en la actualización del servicio. Para obtener más información, consulte los registros de servicio.
`DeploymentStarted`	Se ha iniciado la implementación.
`DeploymentCompletedSuccessfully`	El despliegue se completó correctamente.
`DeploymentFailed`	Error de implementación Para obtener más información, consulte los registros de servicio.
`PauseServiceStarted`	Se inició la pausa del servicio.
`PauseServiceCompletedSuccessfully`	La pausa del servicio se completó correctamente.
`PauseServiceFailed`	Falló la pausa del servicio.

Status	Mensaje
`ResumeServiceStarted`	Se inició la reanudación del servicio.
`ResumeServiceCompletedSuccessfully`	El currículum del servicio se completó correctamente.
`ResumeServiceFailed`	Falló la reanudación del servicio.

Registrar llamadas a la API de App Runner con AWS CloudTrail

App Runner está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en App Runner. CloudTrailcaptura todas las llamadas a la API de App Runner como eventos. Las llamadas capturadas incluyen llamadas desde la consola de App Runner y llamadas en código a las operaciones de la API de App Runner. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de App Runner. Si no configuras una ruta, podrás ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puedes determinar la solicitud que se realizó a App Runner, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulta la Guía AWS CloudTrail del usuario.

Información sobre App Runner en CloudTrail

CloudTrail está habilitada en tu cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en App Runner, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de tus eventos Cuenta de AWS, incluidos los eventos de App Runner, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de

eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail Integraciones y servicios compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de App Runner se registran CloudTrail y se documentan en la referencia de la AWS App Runner API. Por ejemplo, las llamadas a las `CreateService` `StartDeployment` acciones y las llamadas generan entradas en los archivos de CloudTrail registro. `DeleteConnection`

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario de IAM.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail .

Descripción de las entradas de los archivos de registro de App Runner

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción y los parámetros de la solicitud. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a las API públicas, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la `CreateService` acción.

> **Note**
>
> Por motivos de seguridad, algunos valores de propiedades se redactan en los registros y se sustituyen por el textoHIDDEN_DUE_TO_SECURITY_REASONS. Esto evita la exposición involuntaria de información secreta. Sin embargo, aún puede ver que estas propiedades se incluyeron en la solicitud o se devolvieron en la respuesta.

Ejemplo de entrada de CloudTrail registro para la acción de **`CreateService`** App Runner


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/aws-user",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "aws-user"
  },
  "eventTime": "2020-10-02T23:25:33Z",
  "eventSource": "apprunner.amazonaws.com",
  "eventName": "CreateService",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/77.0.3865.75 Safari/537.36",
  "requestParameters": {
    "serviceName": "python-test",
    "sourceConfiguration": {
      "codeRepository": {
        "repositoryUrl": "https://github.com/github-user/python-hello",
        "sourceCodeVersion": {
          "type": "BRANCH",
          "value": "main"
        },
        "codeConfiguration": {
          "configurationSource": "API",
          "codeConfigurationValues": {
            "runtime": "python3",
            "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
```

```
            "port": "8080",
            "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
          }
        }
      },
      "autoDeploymentsEnabled": true,
      "authenticationConfiguration": {
        "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/your-
connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "healthCheckConfiguration": {
      "protocol": "HTTP"
    },
    "instanceConfiguration": {
      "cpu": "256",
      "memory": "1024"
    }
  },
  "responseElements": {
    "service": {
        "serviceName": "python-test",
        "serviceId": "dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceArn": "arn:aws:apprunner:us-east-2:123456789012:service/python-test/
dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceUrl": "generated domain",
        "createdAt": "2020-10-02T23:25:32.650Z",
        "updatedAt": "2020-10-02T23:25:32.650Z",
        "status": "OPERATION_IN_PROGRESS",
        "sourceConfiguration": {
            "codeRepository": {
                "repositoryUrl": "https://github.com/github-user/python-hello",
                "sourceCodeVersion": {
                    "type": "Branch",
                    "value": "main"
                },
                "sourceDirectory": "/",
                "codeConfiguration": {
                    "codeConfigurationValues": {
                        "configurationSource": "API",
                        "runtime": "python3",
                        "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "port": "8080",
```

```
                        "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    }
                }
            },
            "autoDeploymentsEnabled": true,
            "authenticationConfiguration": {
                "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/
your-connection/e7656250f67242d7819feade6800f59e"
            }
        },
        "healthCheckConfiguration": {
            "protocol": "HTTP",
            "path": "/",
            "interval": 5,
            "timeout": 2,
            "healthyThreshold": 3,
            "unhealthyThreshold": 5
        },
        "instanceConfiguration": {
            "cpu": "256",
            "memory": "1024"
        },
        "autoScalingConfigurationSummary": {
            "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
            "autoScalingConfigurationName": "DefaultConfiguration",
            "autoScalingConfigurationRevision": 1
        }
    }
},
  "requestID": "1a60af60-ecf5-4280-aa8f-64538319ba0a",
  "eventID": "e1a3f623-4d24-4390-a70b-bf08a0e24669",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

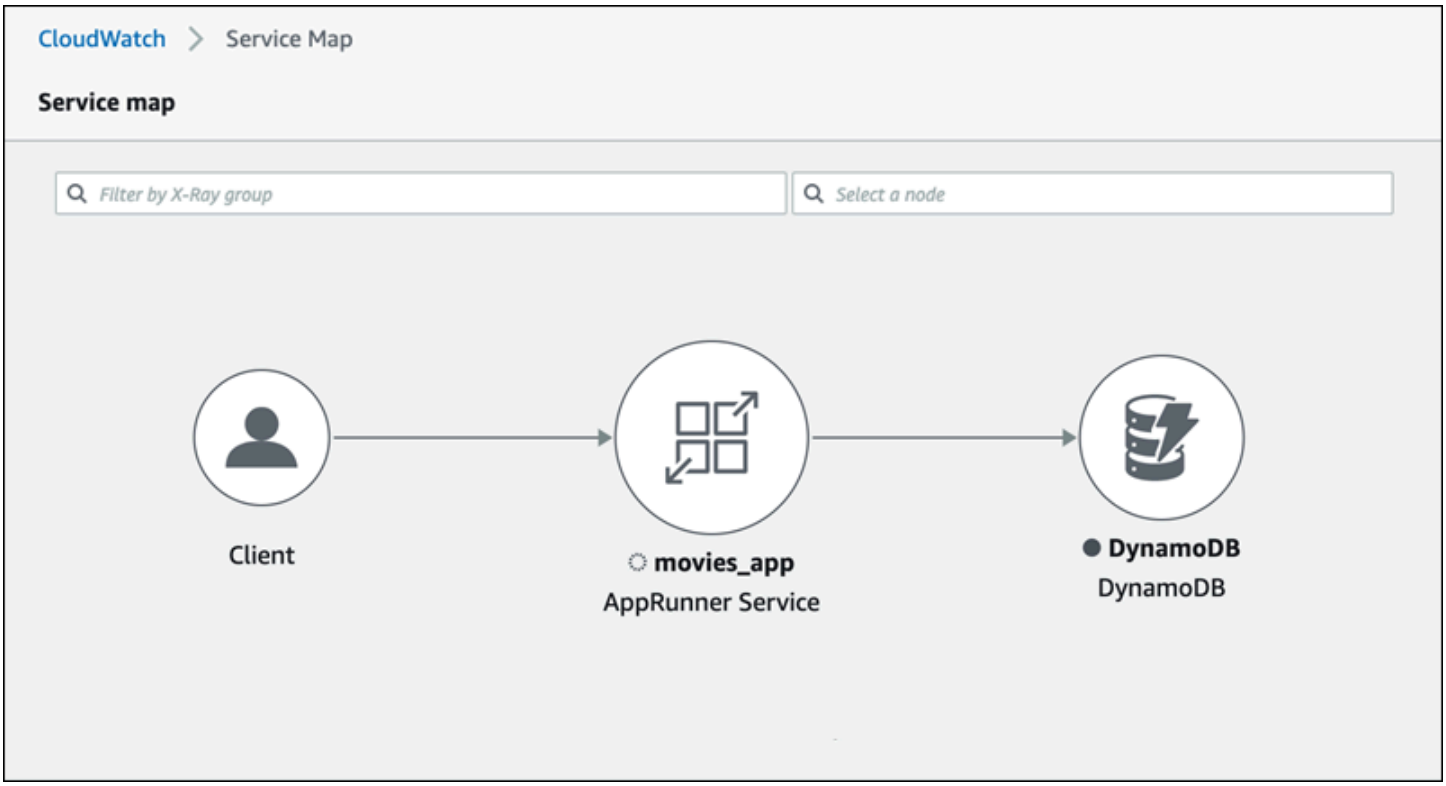
Rastreo de su aplicación App Runner con X-Ray

AWS X-Ray es un servicio que recopila datos sobre las solicitudes que atiende su aplicación y proporciona herramientas que puede utilizar para ver, filtrar y obtener información sobre esos datos a

fin de identificar problemas y oportunidades de optimización. Para cualquier solicitud rastreada hasta su aplicación, puede ver información detallada no solo sobre la solicitud y la respuesta, sino también sobre las llamadas que la aplicación realiza a AWS recursos intermedios, microservicios, bases de datos y sitios web HTTP. APIs

X-Ray utiliza datos de rastreo de los AWS recursos que impulsan sus aplicaciones en la nube para generar un gráfico de servicio detallado. El gráfico de servicios muestra el cliente, el servicio front-end y los servicios back-end a los que llama el servicio front-end para procesar solicitudes y mantener los datos. Utilice el gráfico de servicios para identificar cuellos de botella, picos de latencia y otros problemas que puede resolver a fin de mejorar el desempeño de las aplicaciones.

A fin de obtener más información sobre X-Ray, consulte la Guía para desarrolladores de AWS X-Ray.



Instrumente su aplicación para el rastreo

Instrumente su aplicación de servicio App Runner para el rastreo mediante OpenTelemetryuna especificación de telemetría portátil. En este momento, App Runner es compatible con AWS Distro for OpenTelemetry (ADOT), una OpenTelemetry implementación que recopila y presenta información de telemetría mediante servicios. AWS X-Ray implementa el componente de rastreo.

Según el SDK de ADOT específico que utilice en su aplicación, ADOT admite hasta dos enfoques de instrumentación: automática y manual. Para obtener más información sobre la instrumentación con su SDK, consulte la documentación de ADOT y elija su SDK en el panel de navegación.

Configuración del tiempo de ejecución

Las siguientes son las instrucciones generales de configuración del tiempo de ejecución para instrumentar la aplicación de servicio App Runner para el rastreo.

Para configurar el rastreo para su tiempo de ejecución

1. Siga las instrucciones proporcionadas para su tiempo de ejecución en AWS Distro for OpenTelemetry (ADOT) para instrumentar su aplicación.
2. Instale `OTEL` las dependencias necesarias en la `build` sección del `apprunner.yaml` archivo si utiliza el repositorio de código fuente o en el Dockerfile si utiliza una imagen contenedora.
3. Configure las variables de entorno en el `apprunner.yaml` archivo si utiliza el repositorio de código fuente o en el Dockerfile si utiliza una imagen de contenedor.

   Example Variables de entorno

   > **Note**
   >
   > En el siguiente ejemplo, se enumeran las variables de entorno importantes que se van a añadir al `apprunner.yaml` archivo. Añada estas variables de entorno a su Dockerfile si utiliza una imagen de contenedor. Sin embargo, cada tiempo de ejecución puede tener su propia idiosincrasia y es posible que tengas que añadir más variables de entorno a la siguiente lista. Para obtener más información sobre las instrucciones específicas del tiempo de ejecución y ejemplos sobre cómo configurar la aplicación para su entorno de ejecución, consulte AWS Distro for OpenTelemetry and go to your runtime, en la sección Cómo empezar.

```
env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_EXPORTER_OTLP_ENDPOINT
      value: http://localhost:4317
```

```
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

> **Note**
>
> `OTEL_METRICS_EXPORTER=none`es una variable de entorno importante para App Runner, ya que el recopilador de App Runner Otel no acepta el registro de métricas. Solo acepta el rastreo de métricas.

Ejemplo de configuración del tiempo de ejecución

En el siguiente ejemplo, se muestra la instrumentación automática de la aplicación con el SDK de Python de ADOT. El SDK produce automáticamente intervalos con datos de telemetría que describen los valores utilizados por los marcos de Python de su aplicación sin añadir ni una sola línea de código Python. Solo tienes que añadir o modificar unas pocas líneas en dos archivos fuente.

En primer lugar, añada algunas dependencias, como se muestra en el siguiente ejemplo.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

A continuación, instrumente su aplicación. La forma de hacerlo depende de la fuente de servicio: imagen fuente o código fuente.

Source image

Si la fuente de servicio es una imagen, puede instrumentar directamente el Dockerfile que controla la creación de la imagen del contenedor y la ejecución de la aplicación en la imagen. El siguiente ejemplo muestra un Dockerfile instrumentado para una aplicación de Python. Las adiciones a la instrumentación se destacan en negrita.

Example Dockerfile

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install python3.7 -y && curl -O https://bootstrap.pypa.io/get-pip.py &&
 python3 get-pip.py && yum update -y
```

```
COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
RUN opentelemetry-bootstrap --action=install
ENV OTEL_PYTHON_DISABLED_INSTRUMENTATIONS=urllib3
ENV OTEL_METRICS_EXPORTER=none
ENV OTEL_RESOURCE_ATTRIBUTES='service.name=example_app'
CMD OTEL_PROPAGATORS=xray OTEL_PYTHON_ID_GENERATOR=xray opentelemetry-instrument
 python3 app.py
EXPOSE 8080
```

Source code repository

Cuando la fuente de servicio es un repositorio que contiene la fuente de la aplicación, instrumenta indirectamente la imagen mediante los ajustes del archivo de configuración de App Runner. Estos ajustes controlan el Dockerfile que App Runner genera y usa para crear la imagen de tu aplicación. El siguiente ejemplo muestra un archivo de configuración de App Runner instrumentado para una aplicación de Python. Las adiciones a la instrumentación se destacan en negrita.

Example aprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
      - opentelemetry-bootstrap --action=install
run:
  command: opentelemetry-instrument python app.py
  network:
    port: 8080
  env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_PYTHON_ID_GENERATOR
      value: xray
    - name: OTEL_PYTHON_DISABLED_INSTRUMENTATIONS
      value: urllib3
```

```
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

Agrega permisos de X-Ray a tu rol de instancia de servicio de App Runner

Para usar el rastreo de rayos X con su servicio App Runner, debe proporcionar a las instancias del servicio permisos para interactuar con el servicio de X-Ray. Para ello, asocie un rol de instancia a su servicio y añada una política administrada con permisos de X-Ray. Para obtener más información sobre un rol de instancia de App Runner, consultethe section called “Rol de instancia”. Agrega la política `AWSXRayDaemonWriteAccess` administrada a tu rol de instancia y asígnala a tu servicio durante la creación.

Habilite el rastreo de X-Ray para su servicio App Runner

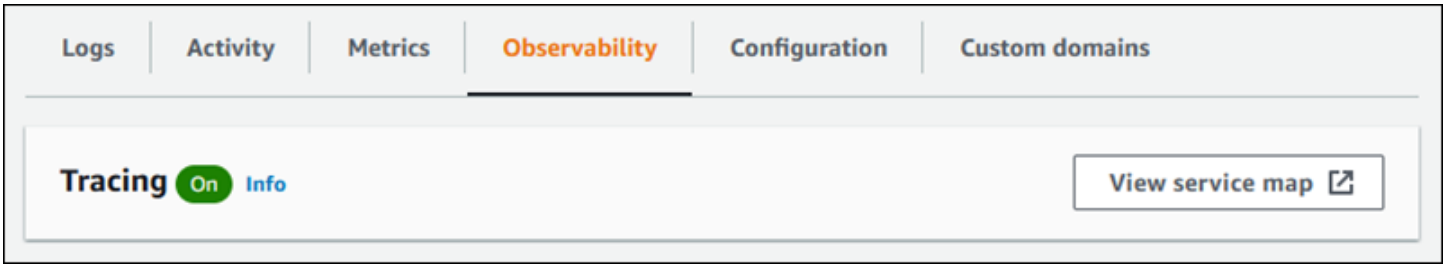
Al crear un servicio, App Runner deshabilita el rastreo de forma predeterminada. Puede habilitar el rastreo de rayos X para su servicio como parte de la configuración de la observabilidad. Para obtener más información, consulte the section called “Gestione la observabilidad”.

Si utilizas la API de App Runner o la AWS CLI, el TraceConfigurationobjeto dentro del objeto de ObservabilityConfigurationrecurso contiene la configuración de rastreo. Para mantener el rastreo desactivado, no especifiques ningún objeto. `TraceConfiguration`

Tanto en el caso de la consola como en el de la API, asegúrate de asociar el rol de instancia descrito en la sección anterior con tu servicio de App Runner.

Vea los datos de rastreo de X-Ray para su servicio App Runner

En la pestaña Observabilidad de la página del panel de servicios de la consola de App Runner, selecciona Ver mapa de servicios para ir a la CloudWatch consola de Amazon.



Usa la CloudWatch consola de Amazon para ver los mapas de servicio y los rastreos de las solicitudes que atiende tu aplicación. Los mapas de servicio muestran información como la latencia

de las solicitudes y las interacciones con otras aplicaciones y AWS servicios. Las anotaciones personalizadas que añades al código te permiten buscar fácilmente los rastros. Para obtener más información, consulta ServiceLens Cómo monitorizar el estado de tus aplicaciones en la Guía del CloudWatch usuario de Amazon.

Asociar una ACL AWS WAF web a su servicio

AWS WAF es un firewall de aplicaciones web que puede usar para proteger su servicio de App Runner. Con las listas de control de acceso AWS WAF web (web ACLs), puedes proteger los puntos finales del servicio de App Runner contra los ataques web más comunes y los bots no deseados.

Una ACL web te proporciona un control detallado de todas las solicitudes web entrantes a tu servicio de App Runner. Puedes definir reglas en una ACL web para permitir, bloquear o monitorear el tráfico web, a fin de garantizar que solo las solicitudes autorizadas y legítimas lleguen a tus aplicaciones web y. APIs Puede personalizar las reglas de ACL web en función de sus necesidades empresariales y de seguridad específicas. Para obtener más información sobre la seguridad de la infraestructura y las prácticas recomendadas para aplicar la red ACLs, consulte Control del tráfico de red en la Guía del usuario de Amazon VPC.

> **Important**
>
> Las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web WAF ACLs no cumplen con las reglas basadas en IP. Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados a WAF. Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar el WAF web. ACLs

Flujo de solicitudes web entrantes

Cuando se asocia una ACL AWS WAF web a un servicio de App Runner, las solicitudes web entrantes se someten al siguiente proceso:

1. App Runner reenvía el contenido de la solicitud de origen a AWS WAF.
2. AWS WAF inspecciona la solicitud y compara su contenido con las reglas que especificó en su ACL web.
3. En función de su inspección, AWS WAF devuelve una `allow` o una `block` respuesta a App Runner.
   - Si se devuelve una `allow` respuesta, App Runner reenvía la solicitud a tu aplicación.

- Si se devuelve una `block` respuesta, App Runner bloquea la solicitud para que no llegue a tu aplicación web. Reenvía la `block` respuesta desde AWS WAF tu aplicación.

  > **Note**
  >
  > De forma predeterminada, App Runner bloquea la solicitud si no se devuelve ninguna respuesta. AWS WAF

Para obtener más información sobre la AWS WAF web ACLs, consulte las listas de control de acceso a la web (web ACLs) en la Guía para AWS WAF desarrolladores.

> **Note**
>
> Usted paga un AWS WAF precio estándar. No incurres en ningún coste adicional por usar la AWS WAF web ACLs para tus servicios de App Runner. Para obtener más información sobre los precios, consulta los AWS WAF precios.

Cómo asociar la web WAF ACLs a tu servicio de App Runner

El siguiente es el proceso de alto nivel para asociar una ACL AWS WAF web a su servicio de App Runner:

1. Cree una ACL web en la AWS WAF consola. Para obtener más información, consulte Creación de una ACL web en la Guía para AWS WAF desarrolladores.
2. Actualice sus permisos AWS Identity and Access Management (de IAM) para AWS WAF. Para obtener más información, consulte Permisos de ..
3. Asocie la ACL web al servicio App Runner mediante uno de los siguientes métodos:
   - Consola de App Runner: asocie una ACL web existente mediante la consola de App Runner al crear o actualizar un servicio de App Runner. Para obtener instrucciones, consulte Administrar la AWS WAF web ACLs.
   - AWS WAF consola: asocie la ACL web mediante la AWS WAF consola a un servicio de App Runner existente. Para obtener más información, consulte Asociar o desasociar una ACL web a un recurso de AWS en la Guía para desarrolladores de AWS WAF .

- AWS CLI: Asocie la ACL web mediante la AWS WAF pública APIs. Para obtener más información sobre el AWS WAF público APIs, consulte la AssociateWebACL en la Guía de referencia de la AWS WAF API.

Consideraciones

- Las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP. Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados a WAF. Si tu aplicación App Runner requiere reglas de control del tráfico IP/CIDR entrante de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar el WAF web. ACLs
- Un servicio de App Runner solo se puede asociar a una ACL web. Sin embargo, puede asociar una ACL web a varios servicios y AWS recursos de App Runner. Algunos ejemplos son los grupos de usuarios de Amazon Cognito y los recursos de Application Load Balancer.
- Al crear una ACL web, pasa un tiempo hasta que la ACL web se propague por completo y esté disponible para App Runner. El tiempo de propagación puede oscilar entre unos segundos y varios minutos. AWS WAF devuelve a `WAFUnavailableEntityException` cuando intenta asociar una ACL web antes de que se haya propagado por completo.

  Si actualizas el navegador o sales de la consola de App Runner antes de que la ACL web se propague por completo, la asociación no se produce. Sin embargo, puedes navegar dentro de la consola de App Runner.
- AWS WAF devuelve un `WAFNonexistentItemException` error cuando llamas a una de las siguientes opciones AWS WAF APIs para un servicio de App Runner que se encuentra en un estado no válido:
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  Los estados no válidos de tu servicio App Runner incluyen:
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`

- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`El estado no es válido solo si se va a eliminar tu servicio de App Runner.

- Tu solicitud podría generar una carga útil superior a los límites de lo que se AWS WAF puede inspeccionar. Para obtener más información sobre cómo AWS WAF gestiona las solicitudes sobredimensionadas de App Runner, consulta la sección Gestión de componentes de solicitudes sobredimensionadas en la Guía para AWS WAF desarrolladores para aprender a gestionar las solicitudes AWS WAF sobredimensionadas de App Runner.
- Si no estableces las reglas adecuadas o tus patrones de tráfico cambian, es posible que una ACL web no sea tan eficaz para proteger tu aplicación.

Permisos

Para trabajar con una ACL web AWS App Runner, añada los siguientes permisos de IAM para AWS WAF:

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

Para obtener más información sobre los permisos de IAM, consulte Políticas y permisos de IAM en la Guía del usuario de IAM.

El siguiente es un ejemplo de la política de IAM actualizada para. AWS WAF Esta política de IAM incluye los permisos necesarios para trabajar con un servicio de App Runner.

Example

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "wafv2:ListResourcesForWebACL",
            "wafv2:GetWebACLForResource",
            "wafv2:AssociateWebACL",
            "wafv2:DisassociateWebACL",
            "apprunner:ListAssociatedServicesForWebAcl",
            "apprunner:DescribeWebAclForService",
            "apprunner:AssociateWebAcl",
            "apprunner:DisassociateWebAcl"
         ],
         "Resource":"*"
      }
   ]
}
```

> **Note**
>
> Si bien debe conceder permisos de IAM, las acciones enumeradas son solo con permisos y no corresponden a una operación de la API.

Administrar la AWS WAF web ACLs

Administre la AWS WAF web ACLs para su servicio de App Runner mediante uno de los siguientes métodos:

- the section called “Consola de App Runner”
- the section called “AWS CLI”

Consola de App Runner

Al crear un servicio o actualizar uno existente en la consola de App Runner, puede asociar o desasociar una ACL AWS WAF web.

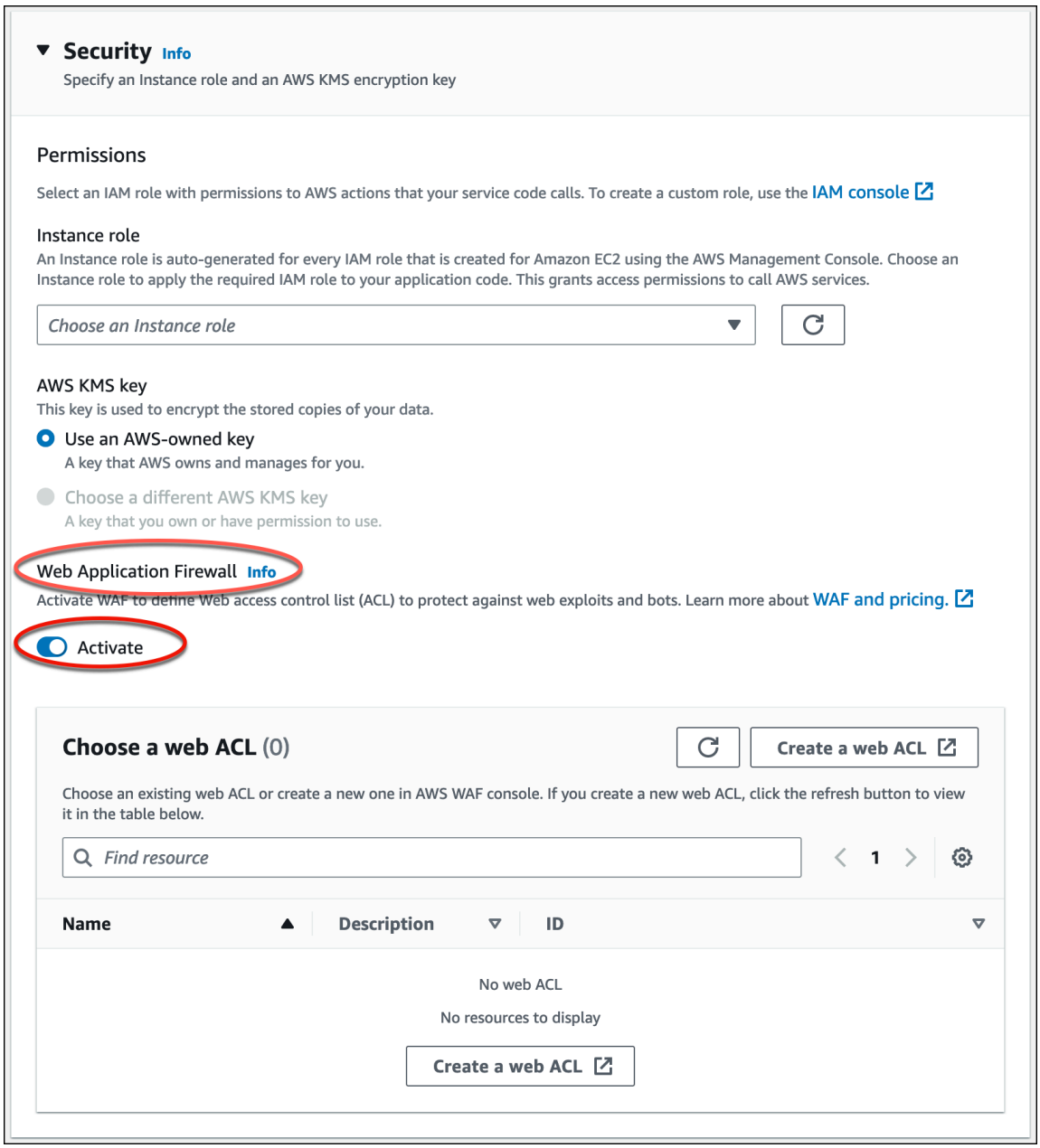
> **Note**
>
> - Un servicio de App Runner solo se puede asociar a una ACL web. Sin embargo, puede asociar una ACL web a más de un servicio de App Runner, además de otros AWS recursos.
> - Antes de asociar una ACL web, asegúrese de actualizar sus permisos de IAM para AWS WAF. Para obtener más información, consulte Permisos de ..

Asociar una ACL web AWS WAF

> **Important**
>
> Las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP. Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados a WAF. Si tu aplicación App Runner requiere reglas de control de tráfico entrante IP/CIDR de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar el WAF web. ACLs

Para asociar una ACL web AWS WAF

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. En función de si va a crear o actualizar un servicio, lleve a cabo uno de los siguientes pasos:
   - Si va a crear un servicio nuevo, elija Crear un servicio de App Runner y vaya a Configurar el servicio.
   - Si va a actualizar un servicio existente, seleccione la pestaña Configuración y, a continuación, elija Editar en Configurar servicio.
3. Ve al firewall de aplicaciones web en Seguridad.
4. Pulse el botón Activar para ver las opciones.



5. Lleve a cabo uno de los siguientes pasos:

   - Para asociar una ACL web existente: elija la ACL web requerida en la tabla Elija una ACL web para asociarla a su servicio de App Runner.
   - Para crear una nueva ACL web: seleccione Crear ACL web para crear una nueva ACL web mediante la AWS WAF consola. Para obtener más información, consulte Creación de una ACL web en la Guía para AWS WAF desarrolladores.

     1. Pulse el botón de actualización para ver la ACL web recién creada en la tabla Elija una ACL web.

2. Seleccione la ACL web requerida.

6. Seleccione Siguiente si va a crear un servicio nuevo o Guardar cambios si va a actualizar un servicio existente. La ACL web seleccionada está asociada a tu servicio App Runner.
7. Para verificar la asociación de la ACL web, elija la pestaña Configuración del servicio y vaya a Configurar el servicio. Diríjase al firewall de aplicaciones web en Seguridad para ver los detalles de la ACL web asociada a su servicio.

> **Note**
>
> Al crear una ACL web, pasa un tiempo hasta que la ACL web se propague por completo y esté disponible para App Runner. El tiempo de propagación puede oscilar entre unos segundos y varios minutos. AWS WAF devuelve a `WAFUnavailableEntityException` cuando intenta asociar una ACL web antes de que se haya propagado por completo.
> Si actualizas el navegador o sales de la consola de App Runner antes de que la ACL web se propague por completo, la asociación no se produce. Sin embargo, puedes navegar dentro de la consola de App Runner.

Desasociar una ACL web AWS WAF

Puedes desasociar los AWS WAF sitios web ACl que ya no necesites actualizando tu servicio App Runner.

Para desasociar una web AWS WAF ACl

1. Abre la consola de App Runner y, en la lista de regiones, selecciona la tuya Región de AWS.
2. Ve a la pestaña Configuración del servicio que deseas actualizar y selecciona Editar en Configurar servicio.
3. Vaya al firewall de aplicaciones web en Seguridad.
4. Desactive el botón de activación. Recibirás un mensaje para confirmar la eliminación.
5. Elija Confirmar. La ACL web está disociada del servicio App Runner.

> **Note**
>
> - Si desea asociar su servicio a otra ACL web, seleccione una ACL web en la tabla Elija una ACL web. App Runner desasocia la ACL web actual e inicia el proceso de asociación con la ACL web seleccionada.
> - Si ningún otro servicio o recurso de App Runner utiliza una ACL web disociada, considere eliminar la ACL web. De lo contrario, seguirá incurriendo en gastos. Para obtener más información sobre los precios, consulte Precios de AWS WAF. Para obtener instrucciones sobre cómo eliminar una ACL web, consulte la DeleteWebACL en la referencia de la AWS WAF API.
> - No puedes eliminar una ACL web que esté asociada a otros servicios activos de App Runner u otros recursos.

AWS CLI

Puede asociar o desasociar una ACL AWS WAF web mediante la AWS WAF pública APIs. El servicio App Runner, con el que desea asociar o desasociar una ACL web, debe estar en un estado válido.

AWS WAF devuelve un `WAFNonexistentItemException` error cuando llamas a una de las siguientes opciones AWS WAF APIs para un servicio de App Runner que se encuentra en un estado no válido:

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

Los estados no válidos de tu servicio App Runner incluyen:

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`El estado no es válido solo si se va a eliminar tu servicio de App Runner.

Para obtener más información sobre el uso AWS WAF público APIs, consulta la Guía AWS WAF de referencia de la API.

> **Note**
>
> Actualice sus permisos de IAM para AWS WAF. Para obtener más información, consulte Permisos de ..

Asociar una ACL AWS WAF web mediante AWS CLI

> **Important**
>
> Las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP. Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados a WAF. Si tu aplicación App Runner requiere reglas de control de tráfico entrante IP/CIDR de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar el WAF web. ACLs

Para asociar una ACL web AWS WAF

1. Cree una ACL AWS WAF web para su servicio con el conjunto preferido de acciones de reglas `Allow` o `Block` las solicitudes web a su servicio. Para obtener más información AWS WAF APIs, consulte la CreateWebACL en la Guía de referencia de la AWS WAF API.

   Example Crear una ACL web: solicitud

   ```
   aws wafv2
   create-web-acl
   --region <region>
   ```

```
--name <web-acl-name>
--scope REGIONAL
--default-action Allow={}
--visibility-config <file-name.json>
# This is the file containing the WAF web ACL rules.
```

2. Asocie la ACL web que creó al servicio App Runner mediante la API `associate-web-acl` AWS WAF pública. Para obtener más información AWS WAF APIs, consulte la AssociateWebACL en la Guía de referencia de la AWS WAF API.

> **Note**
>
> Al crear una ACL web, pasa un tiempo hasta que la ACL web se propague por completo y esté disponible para App Runner. El tiempo de propagación puede oscilar entre unos segundos y varios minutos. AWS WAF devuelve a `WAFUnavailableEntityException` cuando intenta asociar una ACL web antes de que se haya propagado por completo.
> Si actualizas el navegador o sales de la consola de App Runner antes de que la ACL web se propague por completo, la asociación no se produce. Sin embargo, puedes navegar dentro de la consola de App Runner.

Example Asociar una ACL web: solicitud

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. Comprueba que la ACL web esté asociada a tu servicio de App Runner mediante la API `get-web-acl-for-resource` AWS WAF pública. Para obtener más información AWS WAF APIs, consulte el GetWebACLForrecurso en la Guía de referencia de la AWS WAF API.

Example Verifique la ACL web para ver el recurso: solicitud

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

Si no hay ninguna web ACLs asociada a su servicio, recibirá una respuesta en blanco.

Eliminar una ACL AWS WAF web mediante AWS CLI

No puedes eliminar una ACL AWS WAF web si está asociada a un servicio de App Runner.

Para eliminar una ACL AWS WAF web

1. Desasocie la ACL web de su servicio de App Runner mediante la API `disassociate-web-acl` AWS WAF pública. Para obtener más información AWS WAF APIs, consulte la DisassociateWebACL en la Guía de referencia AWS WAF de la API.

   Example Desasociar una ACL web: solicitud

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. Comprueba que la ACL web esté desasociada del servicio de App Runner mediante la `get-web-acl-for-resource` AWS WAF API pública.

   Example Compruebe que la ACL web esté disociada: solicitud

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   La ACL web disociada no aparece en la lista de tu servicio App Runner. Si no hay ninguna web ACLs asociada a su servicio, recibirá una respuesta en blanco.

3. Elimine la ACL web disociada mediante la API `delete-web-acl` AWS WAF pública. Para obtener más información al respecto AWS WAF APIs, consulte la DeleteWebACL en la Guía de referencia de la AWS WAF API.

   Example Eliminar una ACL web: solicitud

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   --id <web_acl_id>
   --lock-token <web_acl_lock_token>
   --region <region>
   ```

4. Verifique que la ACL web se elimine mediante la API `list-web-acl` AWS WAF pública. Para obtener más información al respecto AWS WAF APIs, consulte ListWebACLsla Guía de referencia de la AWS WAF API.

   Example Compruebe que se haya eliminado la ACL web: solicitud

   ```
   aws wafv2 list-web-acls
   --scope REGIONAL
   --region <region>
   ```

   La ACL web eliminada ya no aparece en la lista.

   > **Note**
   >
   > Si una ACL web está asociada a otros servicios activos de App Runner u otros recursos, como los grupos de usuarios de Amazon Cognito, la ACL web no se puede eliminar.

Lista de los servicios de App Runner que están asociados a una ACL web

Una ACL web se puede asociar a varios servicios de App Runner y a otros recursos. Enumere los servicios de App Runner asociados a una ACL web mediante la API `list-resources-for-web-acl` AWS WAF pública. Para obtener más información AWS WAF APIs, consulte la ListResourcesForWebACL en la Guía de referencia de la AWS WAF API.

Example Enumere los servicios de App Runner asociados a una ACL web: solicitud

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example Listar los servicios de App Runner asociados a una ACL web: Respuesta

El siguiente ejemplo ilustra la respuesta cuando no hay servicios de App Runner asociados a una ACL web.

```
{
```

```
  "ResourceArns": []
}
```

Example Enumere los servicios de App Runner asociados a una ACL web: Respuesta

El siguiente ejemplo ilustra la respuesta cuando hay servicios de App Runner asociados a una ACL web.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

Probar y registrar la AWS WAF web ACLs

Cuando estableces una acción de regla como Contar en tu ACL web, AWS WAF agrega la solicitud a un recuento de solicitudes que coinciden con la regla. Para probar una ACL web con tu servicio de App Runner, establece las acciones de la regla en Count y considera el volumen de solicitudes que coinciden con cada regla. Por ejemplo, estableces una regla para la `Block` acción que coincide con un gran número de solicitudes que consideras tráfico de usuarios normal. En ese caso, es posible que tengas que volver a configurar la regla. Para obtener más información, consulta Probar y ajustar tus AWS WAF protecciones en la Guía para AWS WAF desarrolladores.

También puede configurarlo AWS WAF para registrar los encabezados de las solicitudes en un grupo de CloudWatch registros de Amazon Logs, un bucket de Amazon Simple Storage Service (Amazon S3) o un Amazon Data Firehose. Para obtener más información, consulte Registro del tráfico de la ACL web en la Guía para desarrolladores de AWS WAF .

Para acceder a los registros relacionados con la ACL web asociada a su servicio App Runner, consulte los siguientes campos de registro:

- `httpSourceName`: Contiene `APPRUNNER`
- `httpSourceId`: Contiene `customeraccountid-apprunnerserviceid`

Para obtener más información, consulte los ejemplos de registro en la Guía para AWS WAF desarrolladores.

Important

Las reglas de IP de origen para los servicios privados de App Runner que están asociados a la web de WAF ACLs no cumplen con las reglas basadas en IP. Esto se debe a que actualmente no admitimos el reenvío de los datos IP de origen de las solicitudes a los servicios privados de App Runner asociados a WAF. Si tu aplicación App Runner requiere reglas de control de tráfico entrante IP/CIDR de origen, debes usar reglas de grupos de seguridad para los puntos finales privados en lugar de usar el WAF web. ACLs

Configuración de las opciones de servicio de App Runner mediante un archivo de configuración

> **Note**
>
> Los archivos de configuración solo se aplican a los servicios que se basan en el código fuente. No puede usar archivos de configuración con servicios basados en imágenes.

Al crear un AWS App Runner servicio mediante un repositorio de código fuente, AWS App Runner requiere información sobre cómo crear e iniciar el servicio. Puedes proporcionar esta información cada vez que crees un servicio mediante la consola o la API de App Runner. Como alternativa, puede configurar las opciones del servicio mediante un archivo de configuración. Las opciones que especifique en un archivo pasan a formar parte de su repositorio de código fuente y cualquier cambio realizado en estas opciones se registra de forma similar a como se realiza un seguimiento de los cambios en el código fuente. Puedes usar el archivo de configuración de App Runner para especificar más opciones de las que admite la API. No necesitas proporcionar un archivo de configuración si solo necesitas las opciones básicas que admite la API.

El archivo de configuración de App Runner es un archivo YAML cuyo nombre `apprunner.yaml` aparece en el directorio fuente del repositorio de la aplicación. Proporciona opciones de compilación y tiempo de ejecución para tu servicio. Los valores de este archivo indican a App Runner cómo crear e iniciar el servicio y proporcionan un contexto de tiempo de ejecución, como la configuración de la red y las variables de entorno.

El archivo de configuración de App Runner no incluye los ajustes operativos, como la CPU y la memoria.

Para ver ejemplos de archivos de configuración de App Runner, consultethe section called “Ejemplos”. Para obtener una guía de referencia completa, consultethe section called “Referencia”.

Temas

- Ejemplos de archivos de configuración de App Runner
- Referencia del archivo de configuración de App Runner

Ejemplos de archivos de configuración de App Runner

> **Note**
>
> Los archivos de configuración solo se aplican a los servicios que se basan en el código fuente. No puede usar archivos de configuración con servicios basados en imágenes.

Los siguientes ejemplos muestran los archivos AWS App Runner de configuración. Algunos son mínimos y contienen solo los ajustes necesarios. Otras están completas, incluidas todas las secciones del archivo de configuración. Para obtener una descripción general de los archivos de configuración de App Runner, consulteArchivo de configuración de App Runner.

Ejemplos de archivos de configuración

Archivo de configuración mínimo

Con un archivo de configuración mínimo, App Runner asume lo siguiente:

- No se necesitan variables de entorno personalizadas durante la compilación o la ejecución.
- Se utiliza la última versión de tiempo de ejecución.
- Se utilizan el número de puerto y la variable de entorno de puerto predeterminados.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

Archivo de configuración completo

En este ejemplo se muestra el uso de todas las claves de configuración en el formato `apprunner.yaml` original con un tiempo de ejecución gestionado.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Archivo de configuración completo: (usa una versión revisada)

En este ejemplo, se muestra el uso de todas las claves de configuración en un entorno de ejecución gestionado. `apprunner.yaml`

El `pre-run` parámetro solo es compatible con la versión revisada de App Runner. No insertes este parámetro en el archivo de configuración si la aplicación usa versiones en tiempo de ejecución compatibles con la compilación original de App Runner. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

> **Note**
>
> Como este ejemplo es para Python 3.11, utilizamos los `python3` comandos `pip3` y. Para obtener más información, consulte Llamadas para versiones de tiempo de ejecución específicas el tema de la plataforma Python.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
```

```
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Para ver ejemplos de archivos de configuración específicos del tiempo de ejecución gestionado, consulte el subtema específico sobre el tiempo de ejecución que aparece más abajo. Servicio basado en código

Referencia del archivo de configuración de App Runner

> **Note**
>
> Los archivos de configuración solo se aplican a los servicios que se basan en el código fuente. No puede usar archivos de configuración con servicios basados en imágenes.

Este tema es una guía de referencia completa sobre la sintaxis y la semántica de un archivo de AWS App Runner configuración. Para obtener información general sobre los archivos de configuración de App Runner, consulteArchivo de configuración de App Runner.

El archivo de configuración de App Runner es un archivo YAML. `apprunner.yaml`Asígnele un nombre y colóquelo en el directorio fuente del repositorio de la aplicación.

Descripción general de la estructura

El archivo de configuración de App Runner es un archivo YAML. `apprunner.yaml`Asígnele un nombre y colóquelo en el directorio fuente del repositorio de la aplicación.

El archivo de configuración de App Runner contiene las siguientes partes principales:

- Sección superior: contiene claves de nivel superior
- Sección de compilación: configura la etapa de compilación

- Sección de ejecución: configura la etapa de ejecución

Sección superior

Las teclas de la parte superior del archivo proporcionan información general sobre el archivo y el tiempo de ejecución del servicio. Están disponibles las siguientes claves:

- `version`— Necesario. La versión del archivo de configuración de App Runner. Lo ideal es utilizar la última versión.

  Sintaxis

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`— Necesario. El nombre del motor de ejecución que utiliza la aplicación. Para obtener información sobre los tiempos de ejecución disponibles para las distintas plataformas de programación que ofrece App Runner, consulteServicio basado en código.

  > **Note**
  >
  > La convención de nomenclatura de un tiempo de ejecución gestionado es*<language-name><major-version>*.

  Sintaxis

  ```
  runtime: runtime-name
  ```

  Example

  ```
  runtime: python3
  ```

Sección de compilación

La sección de compilación configura la etapa de compilación de la implementación del servicio App Runner. Puede especificar los comandos de compilación y las variables de entorno. Los comandos de compilación son necesarios.

La sección comienza con la `build:` clave y tiene las siguientes subclaves:

- `commands`— Obligatorio. Especifica los comandos que App Runner ejecuta durante las distintas fases de creación. Incluye las siguientes subclaves:
  - `pre-build`— Opcional. Los comandos que App Runner ejecuta antes de la compilación. Por ejemplo, instala npm dependencias o prueba bibliotecas.
  - `build`— Obligatorio. Los comandos que App Runner ejecuta para crear la aplicación. Por ejemplo, utilicepipenv.
  - `post-build`— Opcional. Los comandos que App Runner ejecuta después de la compilación. Por ejemplo, usa Maven para empaquetar los artefactos de compilación en un archivo JAR o WAR, o ejecuta una prueba.

  Sintaxis

```
build:
  commands:
    pre-build:
      - command
      - …
    build:
      - command
      - …
    post-build:
      - command
      - …
```

  Example

```
build:
  commands:
    pre-build:
      - yum install openssl
    build:
      - pip install -r requirements.txt
```

```
    post-build:
      - python manage.py test
```

- `env`— Opcional. Especifica las variables de entorno personalizadas para la etapa de compilación. Definido como mapeos escalares nombre-valor. Puedes hacer referencia a estas variables por su nombre en tus comandos de compilación.

> **Note**
>
> Hay dos `env` entradas distintas en dos ubicaciones diferentes de este archivo de configuración. Un conjunto se encuentra en la sección Construir y el otro en la sección Ejecutar.
>
> - Durante el proceso de compilación*`pre-build`*, se puede hacer referencia al *`env`* conjunto de la sección Construir mediante `pre-run` los comandos `build``post-build`,, y.
>
> Importante: tenga en cuenta que los `pre-run` comandos se encuentran en la sección Ejecutar de este archivo, aunque solo pueden acceder a las variables de entorno que se definen en la sección Crear.
> - El `run` comando puede hacer referencia al `env` conjunto de la sección Ejecutar en el entorno de ejecución.

Sintaxis

```
build:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
    - …
```

Example

```
build:
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
```

```
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Sección de ejecución

La sección de ejecución configura la etapa de ejecución del contenedor de la implementación de la aplicación App Runner. Puede especificar la versión en tiempo de ejecución, los comandos previos a la ejecución (solo en el formato revisado), el comando de inicio, el puerto de red y las variables de entorno.

La sección comienza con la `run:` clave y tiene las siguientes subclaves:

- `runtime-version`— Opcional. Especifica la versión en tiempo de ejecución que deseas bloquear para tu servicio App Runner.

  De forma predeterminada, solo está bloqueada la versión principal. App Runner utiliza las versiones secundarias y de parche más recientes que están disponibles para el tiempo de ejecución en cada implementación o actualización de servicio. Si especificas las versiones principales y secundarias, ambas se bloquean y App Runner solo actualiza las versiones con parches. Si especificas las versiones principales, secundarias y de parche, el servicio se bloquea en una versión de tiempo de ejecución específica y App Runner nunca la actualiza.

  Sintaxis

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  >
  > Los tiempos de ejecución de algunas plataformas tienen componentes de versión diferentes. Consulte los temas específicos de la plataforma para obtener más información.

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run`— Opcional. Solo se revisó el uso de la compilación. Especifica los comandos que App Runner ejecuta después de copiar la aplicación de la imagen de compilación a la imagen de ejecución. Aquí puede introducir comandos para modificar la imagen de ejecución fuera del `/app` directorio. Por ejemplo, si necesita instalar dependencias globales adicionales que residan fuera del `/app` directorio, introduzca los comandos necesarios en esta subsección para hacerlo. Para obtener más información sobre el proceso de creación de App Runner, consulte. Versiones administradas en tiempo de ejecución y compilación de App Runner

  > **Note**
  >
  > - Importante: aunque los `pre-run` comandos aparecen en la sección Ejecutar, solo pueden hacer referencia a las variables de entorno definidas en la sección Crear de este archivo de configuración. No pueden hacer referencia a las variables de entorno definidas en esta sección de ejecución.
  > - El `pre-run` parámetro solo es compatible con la versión revisada de App Runner. No insertes este parámetro en el archivo de configuración si la aplicación usa versiones en tiempo de ejecución compatibles con la compilación original de App Runner. Para obtener más información, consulte Versiones administradas en tiempo de ejecución y compilación de App Runner.

  Sintaxis

  ```
  run:
    pre-run:
        - command
        - …
  ```

- `command`— Necesario. El comando que App Runner usa para ejecutar la aplicación después de completar la compilación de la aplicación.

  Sintaxis

  ```
  run:
    command: command
  ```

- `network`— Opcional. Especifica el puerto que escucha la aplicación. Contiene lo siguiente:

- `port`— Opcional. Si se especifica, este es el número de puerto que escucha la aplicación. El valor predeterminado es `8080`.
- `env`— Opcional. Si se especifica, App Runner pasa el número de puerto al contenedor en esta variable de entorno, además de (no en lugar de) pasar el mismo número de puerto en la variable de entorno predeterminada`PORT`. En otras palabras, si lo especifica`env`, App Runner pasa el número de puerto en dos variables de entorno.

Sintaxis

```
run:
  network:
    port: port-number
    env: env-variable-name
```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- `env`— Opcional. Definición de variables de entorno personalizadas para la etapa de ejecución. Definido como mapeos escalares nombre-valor. Puede hacer referencia a estas variables por su nombre en su entorno de ejecución.

> **Note**
>
> Hay dos `env` entradas distintas en dos ubicaciones diferentes de este archivo de configuración. Un conjunto se encuentra en la sección Construir y el otro en la sección Ejecutar.
>
> - Durante el proceso de compilación*`pre-build`*, se puede hacer referencia al *`env`* conjunto de la sección Construir mediante `pre-run` los comandos `build`,`post-build`, y.
>
> Importante: tenga en cuenta que los `pre-run` comandos se encuentran en la sección Ejecutar de este archivo, aunque solo pueden acceder a las variables de entorno que se definen en la sección Crear.

- El `run` comando puede hacer referencia al `env` conjunto de la sección Ejecutar en el entorno de ejecución.

Sintaxis

```
run:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
  secrets:
    - name: name1
      value-from: arn:aws:secretsmanager:region:aws_account_id:secret:secret-id
    - name: name2
      value-from: arn:aws:ssm:region:aws_account_id:parameter/parameter-name
    - …
```

Example

```
run:
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-
S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

La API de App Runner

La interfaz de programación de AWS App Runner aplicaciones (API) es una RESTful API para realizar solicitudes al servicio App Runner. Puedes usar la API para crear, enumerar, describir, actualizar y eliminar los recursos de App Runner en tu Cuenta de AWS.

Puedes llamar a la API directamente en el código de tu aplicación o puedes usar una de las AWS SDKs.

Para obtener información de referencia completa sobre la API, consulta la Referencia AWS App Runner de la API.

Para obtener más información sobre las herramientas para AWS desarrolladores, consulta Herramientas sobre las que construir AWS.

Temas

- Utilización de AWS CLI para trabajar con App Runner
- Uso AWS CloudShell para trabajar con AWS App Runner

Utilización de AWS CLI para trabajar con App Runner

Para los scripts de línea de comandos, utilice el AWS CLIpara realizar llamadas al servicio App Runner. Para obtener información de AWS CLI referencia completa, consulte el apprunner en la Referencia de AWS CLI comandos.

AWS CloudShell le permite omitir la instalación de AWS CLI en su entorno de desarrollo y utilizarla en su lugar. Consola de administración de AWS Además de evitar la instalación, tampoco es necesario configurar las credenciales ni especificar la región. Su Consola de administración de AWS sesión proporciona este contexto a AWS CLI. Para obtener más información CloudShell y un ejemplo de uso, consultethe section called “Usando AWS CloudShell”.

Uso AWS CloudShell para trabajar con AWS App Runner

AWS CloudShell es un shell preautenticado y basado en un navegador que puedes iniciar directamente desde. Consola de administración de AWS Puede ejecutar AWS CLI comandos en los AWS servicios (incluso AWS App Runner) utilizando el shell que prefiera (Bash o Z shell).

PowerShell Y puede hacerlo sin necesidad de descargar o instalar herramientas de línea de comandos.

Los AWS CloudShell ejecutas desde la Consola de administración de AWS consola y las AWS credenciales que usaste para iniciar sesión en la consola estarán disponibles automáticamente en una nueva sesión de shell. Esta autenticación previa de AWS CloudShell los usuarios permite omitir la configuración de las credenciales al interactuar con AWS servicios como App Runner que utilizan la AWS CLI versión 2 (preinstalada en el entorno informático del shell).

Temas

- Obtener permisos de IAM para AWS CloudShell
- Interactúa con App Runner mediante AWS CloudShell
- Verificar el servicio App Runner mediante AWS CloudShell

Obtener permisos de IAM para AWS CloudShell

Con los recursos de administración de acceso que proporcionan AWS Identity and Access Management, los administradores pueden conceder permisos a los usuarios de IAM para que puedan acceder a las funciones del entorno AWS CloudShell y utilizarlas.

La forma más rápida de que un administrador conceda acceso a los usuarios es mediante una política AWS gestionada. Una política administrada de AWS es una política independiente creada y administrada por AWS. La siguiente política AWS administrada para se CloudShell puede adjuntar a las identidades de IAM:

- `AWSCloudShellFullAccess`: Concede permiso de uso AWS CloudShell con acceso completo a todas las funciones.

Si desea limitar el alcance de las acciones con las que puede realizar un usuario de IAM AWS CloudShell, puede crear una política personalizada que utilice la política `AWSCloudShellFullAccess` gestionada como plantilla. Para obtener más información sobre cómo limitar las acciones que están disponibles para los usuarios CloudShell, consulte Administrar el AWS CloudShell acceso y el uso con políticas de IAM en la Guía del AWS CloudShell usuario.

> **Note**
>
> Tu identidad de IAM también requiere una política que otorgue permiso para realizar llamadas a App Runner. Para obtener más información, consulte the section called “App Runner e IAM”.

Interactúa con App Runner mediante AWS CloudShell

Tras iniciarlo AWS CloudShell desde Consola de administración de AWS, podrás empezar a interactuar inmediatamente con App Runner mediante la interfaz de línea de comandos.

En el siguiente ejemplo, recuperas información sobre uno de tus servicios de App Runner mediante la función AWS CLI in CloudShell.

> **Note**
>
> AWS CLI Al usarlo AWS CloudShell, no necesitas descargar ni instalar ningún recurso adicional. Además, dado que ya está autenticado en el intérprete de comandos, no tiene que configurar las credenciales antes de realizar llamadas.

Example Recuperar la información del servicio de App Runner mediante AWS CloudShell

1. Desde Consola de administración de AWS, puede iniciar CloudShell seleccionando las siguientes opciones disponibles en la barra de navegación:
   - Selecciona el CloudShell icono.
   - Comience a escribir **`cloudshell`** en el cuadro de búsqueda y, a continuación, elija la CloudShellopción cuando aparezca en los resultados de la búsqueda.
2. Para ver todos los servicios actuales de App Runner de su AWS cuenta en la AWS región de la sesión de la consola, introduzca el siguiente comando en la línea de CloudShell comandos:

   ```
   $ aws apprunner list-services
   ```

   El resultado muestra un resumen de la información de sus servicios.

   ```
   {
   ```

```
  "ServiceSummaryList": [
    {
      "ServiceName": "my-app-1",
      "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-20T19:05:25Z",
      "UpdatedAt": "2020-11-23T12:41:37Z",
      "Status": "RUNNING"
    },
    {
      "ServiceName": "my-app-2",
      "ServiceId": "ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-app-2/
ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceUrl": "e2m8rrrx33.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-06T23:15:30Z",
      "UpdatedAt": "2020-11-23T13:21:22Z",
      "Status": "RUNNING"
    }
  ]
}
```

3. Para obtener una descripción detallada de un servicio concreto de App Runner, ingresa el siguiente comando en la línea de CloudShell comandos, usando uno de los ARNs recuperados en el paso anterior:

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

El resultado muestra una descripción detallada del servicio que especificó.

```
{
  "Service": {
    "ServiceName": "my-app-1",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-23T12:41:37Z",
```

```
    "Status": "RUNNING",
    "SourceConfiguration": {
      "CodeRepository": {
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        },
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "[pip install -r requirements.txt]",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "API"
        }
      },
      "AutoDeploymentsEnabled": true,
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/my-
github-connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 10,
      "Timeout": 5,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
```

```
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    }
  }
}
```

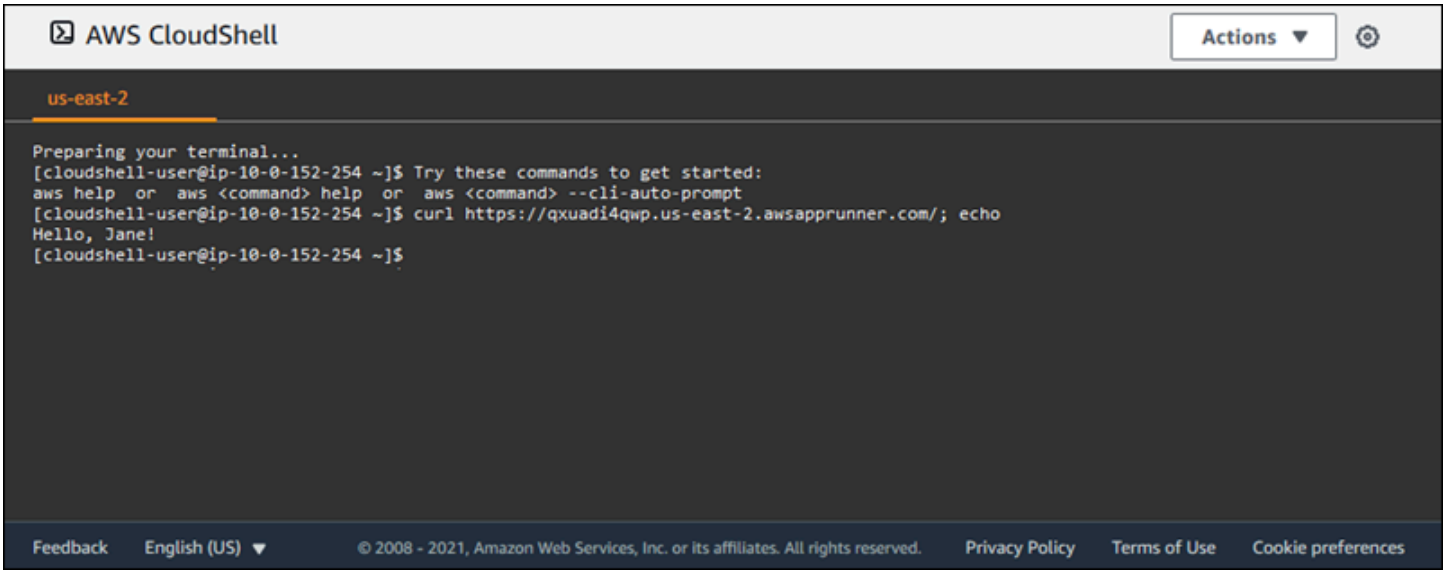
Verificar el servicio App Runner mediante AWS CloudShell

Al crear un servicio de App Runner, App Runner crea un dominio predeterminado para el sitio web del servicio y lo muestra en la consola (o lo devuelve en el resultado de la llamada a la API). Puedes usarlo CloudShell para hacer llamadas a tu sitio web y comprobar que funciona correctamente.

Por ejemplo, después de crear un servicio de App Runner como se describe enIntroducción, ejecute el siguiente comando en CloudShell:

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

El resultado debe mostrar el contenido de la página esperado.



Resolución de problemas

En este capítulo, se proporcionan los pasos para solucionar los errores y problemas más comunes que pueden surgir al utilizar el AWS App Runner servicio. Los mensajes de error pueden aparecer en la consola, la API o la pestaña Registros de la página de tu servicio.

Para obtener más consejos sobre la resolución de problemas y respuestas a preguntas comunes de soporte, visite el Centro de conocimientos de .

Temas

Cuando el servicio no se crea

Si se produce un error al intentar crear un servicio de App Runner, el servicio pasa a un `CREATE_FAILED` estado. Este estado aparece como Error al crear en la consola. Es posible que no se pueda crear un servicio debido a problemas relacionados con uno o varios de los siguientes motivos:

- Su código de la aplicación
- El proceso de creación
- Configuración
- Cuotas de recursos
- Problemas temporales con el subyacente Servicios de AWS que utiliza su servicio

Para solucionar un problema de error al crear un servicio, le recomendamos que haga lo siguiente.

1. Lee los eventos y los registros del servicio para averiguar qué provocó que no se creara el servicio.
2. Realice los cambios necesarios en el código o la configuración.

3. Si has alcanzado tu cuota de servicio, elimina uno o más servicios.
4. Si has alcanzado otra cuota de recursos, es posible que puedas aumentarla si es ajustable.
5. Intenta volver a crear el servicio después de completar todos los pasos anteriores. Para obtener información sobre cómo reconstruir el servicio, consultethe section called “Reconstruir el servicio fallido”.

> **Note**
>
> Una de las cuotas de recursos ajustables que podría estar causando un problema es el recurso vCPU bajo demanda de Fargate.
> El recuento de recursos de vCPU determina la cantidad de instancias que App Runner puede proporcionar a su servicio. Se trata de un valor de cuota ajustable para el recuento de recursos de vCPU bajo demanda de Fargate que reside en el servicio. AWS Fargate Para ver la configuración de la cuota de vCPU de su cuenta o para solicitar un aumento de la cuota, utilice la consola Service Quotas del. Consola de administración de AWS Para obtener más información, consulte las cuotas AWS Fargate de servicio en la Guía para desarrolladores de Amazon Elastic Container Service.

> **Important**
>
> No incurrirá en ningún cargo adicional más allá del intento de creación inicial en caso de un servicio fallido. Aunque el servicio fallido no se pueda utilizar, se tendrá en cuenta para la cuota de servicio. App Runner no elimina automáticamente el servicio fallido, así que asegúrate de eliminarlo cuando termines de analizar el error.

Nombres de dominio personalizados

En esta sección, se explica cómo puedes solucionar y solucionar varios errores que se pueden producir al vincular a un dominio personalizado.

> **Note**
>
> Para aumentar la seguridad de las aplicaciones de App Runner, el dominio*.awsapprunner.com está registrado en la lista pública de sufijos (PSL). Para mayor seguridad, te recomendamos que utilices cookies con un `__Host-` prefijo si alguna vez

necesitas configurar cookies confidenciales en el nombre de dominio predeterminado de tus aplicaciones de App Runner. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

Aparece el error Create Fail para un dominio personalizado

- Compruebe si este error se debe a un problema con los registros de la CAA. Si no hay registros CAA en ninguna parte del árbol de DNS, recibirá un mensaje `fail open` y AWS Certificate Manager emitirá un certificado para verificar el dominio personalizado. Esto permite a App Runner aceptar el dominio personalizado. Si utilizas certificaciones de la CAA en los registros de DNS, asegúrate de que al menos los registros de la CAA de un dominio las incluyan`amazon.com`. De lo contrario, ACM no emitirá un certificado. Como resultado, no se puede crear el dominio personalizado de App Runner.

  En el siguiente ejemplo, se utiliza la herramienta de búsqueda de DNS DiG para mostrar los registros CAA a los que les falta una entrada obligatoria. En el ejemplo `example.com` se utiliza como dominio personalizado. Ejecute los siguientes comandos del ejemplo para comprobar los registros de la CAA.

```
...
;; QUESTION SECTION:
;example.com.           IN  CAA

;; ANSWER SECTION:
example.com.        7200    IN  CAA 0 iodef "mailto:hostmaster@example.com"
example.com.        7200    IN  CAA 0 issue "letsencrypt.org"
...note absence of "amazon.com" in any of the above CAA records...
```

- Corrija los registros de dominio y asegúrese de que al menos un registro de la CAA los incluya`amazon.com`.
- Vuelva a intentar vincular el dominio personalizado con App Runner.

Para obtener instrucciones sobre cómo resolver los errores de la CAA, consulte lo siguiente:

- Problemas con la autorización de la autoridad de certificación (CAA)

- ¿Cómo resuelvo los errores de la CAA al emitir o renovar un certificado ACM?

Se obtiene un error pendiente de validación del certificado DNS para un dominio personalizado

- Comprueba si te has saltado un paso importante en la configuración del dominio personalizado. Además, compruebe si ha configurado incorrectamente un registro de DNS mediante una herramienta de búsqueda de DNS como DiG. En concreto, compruebe si hay los siguientes errores:
  - Cualquier paso omitido.
  - Caracteres no admitidos, como comillas dobles, en los registros DNS.
- Corrija los errores.
- Vuelva a intentar vincular el dominio personalizado con App Runner.

Para obtener instrucciones sobre cómo resolver los errores de validación de la CAA, consulte lo siguiente.

- Validación de DNS
- the section called “Nombres de dominio personalizados”

Comandos básicos de solución de problemas

- Confirme que se puede encontrar un servicio.

```
aws apprunner list-services
```

- Describa un servicio y compruebe su estado.

```
aws apprunner describe-service --service-arn
```

- Comprueba el estado del dominio personalizado.

```
aws apprunner describe-custom-domains --service-arn
```

- Enumere todas las operaciones en curso.

```
aws apprunner list-operations --service-arn
```

Renovación de certificados de dominio personalizados

Cuando agregas un dominio personalizado a tu servicio, App Runner te proporciona un conjunto de registros CNAME que puedes agregar a tu servidor DNS. Estos registros CNAME incluyen registros de certificados. App Runner usa AWS Certificate Manager (ACM) para verificar el dominio. App Runner valida estos registros de DNS para garantizar la propiedad continua de este dominio. Si eliminas los registros CNAME de tu zona DNS, App Runner ya no podrá validarlos y el certificado de dominio personalizado no se renovará automáticamente.

En esta sección se explica cómo resolver los siguientes problemas de renovación de certificados de dominio personalizados:

- the section called “El CNAME se elimina del servidor DNS”.
- the section called “El certificado ha caducado”.

El CNAME se elimina del servidor DNS

- Recupera tus registros CNAME mediante la DescribeCustomDomainsAPI o desde la configuración del dominio personalizado de la consola de App Runner. Para obtener información sobre los almacenados CNAMEs, consulte CertificateValidationRecords.
- Agregue los registros CNAME de validación del certificado a su servidor DNS. Luego, App Runner puede validar que eres el propietario del dominio. Después de agregar los registros CNAME, los registros DNS pueden tardar hasta 30 minutos en propagarse. App Runner y ACM también pueden tardar varias horas en volver a intentar el proceso de renovación del certificado. Para obtener instrucciones sobre cómo añadir registros CNAME, consulte. the section called “Administra dominios personalizados”

El certificado ha caducado

- Desasocie (desvincule) y, a continuación, asocie (vincule) el dominio personalizado a su servicio de App Runner mediante la consola o la API de App Runner. App Runner crea un nuevo registro CNAME de validación de certificados.
- Agregue los nuevos registros CNAME de validación de certificados a su servidor DNS.

  Para obtener instrucciones sobre cómo desasociar (desvincular) y asociar (vincular) el dominio personalizado, consulte. the section called “Administra dominios personalizados”

¿Cómo puedo comprobar que el certificado se ha renovado correctamente

Puede comprobar el estado de los registros de su certificado para comprobar que el certificado se ha renovado correctamente. Puede comprobar el estado de los certificados mediante herramientas como curl.

Para obtener más información sobre la renovación de certificados, consulte los siguientes enlaces:

- ¿Por qué mi certificado ACM está marcado como no apto para la renovación?
- Renovación gestionada de los certificados ACM
- Validación de DNS

Error de enrutamiento de solicitudes HTTP/HTTPS

En esta sección, se explica cómo puedes solucionar y solucionar los errores que podrías encontrar al enrutar el HTTP/HTTPS tráfico a los puntos finales del servicio de App Runner.

Error 404: No se encontró el error al enviar HTTP/HTTPS tráfico a los puntos finales del servicio de App Runner

- Comprueba que apunte a la `Host Header` URL del servicio en la solicitud HTTP, ya que App Runner utiliza la información del encabezado del host para enrutar las solicitudes. La mayoría de los clientes`cURL`, como los navegadores web, dirigen automáticamente el encabezado del host a la URL del servicio. Si tu cliente no establece la URL del servicio como la`Host Header`, recibirás un `404 Not Found` error.

Example El encabezado del host es incorrecto

```
$ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
HTTP/1.1 404 Not Found
transfer-encoding: chunked
```

Example Cabecera de host correcta

```
$ ~ curl -I -H "host: testservice.awsapprunner.com" https://
testservice.awsapprunner.com/
HTTP/1.1 200 OK
content-length: 11772
content-type: text/html; charset=utf-8
```

- Compruebe que su cliente esté configurando correctamente el indicador de nombre de servidor (SNI) para las solicitudes que se enrutan a servicios públicos o privados. Para la terminación de TLS y el enrutamiento de solicitudes, App Runner usa el SNI establecido en la conexión HTTPS.

Cuando el servicio no se conecta a Amazon RDS o al servicio descendente

Es posible que haya un problema de configuración de red con su servicio si no se conecta a una base de datos de Amazon RDS o a otra aplicación o servicio descendente. En este tema se explican algunos pasos para determinar si hay algún problema con la configuración de la red y las opciones para corregirlo. Para obtener más información sobre la configuración del tráfico saliente para App Runner, consulteHabilitar el acceso a la VPC para el tráfico saliente.

> **Note**
>
> Para ver la configuración del conector de VPC, en el panel de navegación izquierdo de la consola de App Runner, selecciona Configuración de red. A continuación, selecciona la pestaña Tráfico saliente. Seleccione un conector de VPC. La página siguiente muestra detalles sobre el conector de VPC. En esta página, puede ver y profundizar en lo siguiente: subredes, grupos de seguridad y servicios de App Runner que utilizan la VPC.

Para determinar la causa de la incapacidad de su aplicación para conectarse a otro servicio descendente

1. Asegúrese de que las subredes utilizadas en los conectores de VPC sean subredes privadas. Si un conector está configurado con una subred pública, el servicio detectará errores, ya que el hiperplano subyacente ENIs (interfaces de red elásticas) de cada subred no tiene un espacio IP público.

   Si sus conectores de VPC utilizan subredes públicas, tiene las siguientes opciones para corregir esta configuración:

   a. Cree una nueva subred privada y úsela en lugar de la subred pública para el conector de VPC. Para obtener más información, consulte Subredes para su VPC en la Guía del usuario de Amazon VPC.

   b. Enrute la subred pública existente a través de puertas de enlace NAT. Para obtener más información, consulte las puertas de enlace NAT en la Guía del usuario de Amazon VPC.

2. Compruebe que las reglas de entrada y salida del grupo de seguridad para el conector de VPC sean correctas. En el panel de navegación izquierdo de la consola de App Runner, seleccione Configuración de red > Tráfico saliente. Seleccione el conector de VPC de la lista. En la página siguiente se enumeran los grupos de seguridad que puede seleccionar para inspeccionar.

3. Compruebe que las reglas de entrada y salida del grupo de seguridad sean correctas para la instancia de RDS u otro servicio descendente al que esté intentando conectarse. Para obtener más información, consulta la guía de servicios del servicio descendente al que intenta conectarse tu aplicación App Runner.

4. Para confirmar que no hay ningún otro tipo de problema de configuración de red aparte de las configuraciones de App Runner, intenta conectarte al RDS o al servicio descendente externo a App Runner:

   a. Desde una instancia de Amazon EC2 en la misma VPC, intente conectarse a la instancia o servicio de RDS.

   b. Si intenta conectarse a un punto final de VPC de servicio, compruebe la conectividad accediendo al mismo punto final desde una instancia EC2 en la misma VPC.

5. Si alguna de las pruebas de conexión del paso 4 falla, lo más probable es que se trate de un problema ajeno a las configuraciones de App Runner con otro recurso de tu cuenta. AWS Póngase en contacto con AWS Support para obtener ayuda para aislar y solucionar aún más el problema con sus otras configuraciones de red.

6. Si se conecta correctamente a la instancia de RDS o al servicio descendente siguiendo las instrucciones del paso 4, continúe con las instrucciones de este paso. Comprobaremos si el tráfico entra en el ENI activando e inspeccionando los registros de flujo del ENI de Hyperplane.

> **Note**
>
> Para poder completar estos pasos y obtener la información de registro de flujo del ENI requerida, el intento de conexión al RDS o al servicio descendente debe producirse después de que el servicio de App Runner se haya iniciado correctamente. La aplicación debe realizar la operación de conexión al RDS o al servicio descendente cuando está en ejecución. De lo contrario, ENIs podrían limpiarse como parte de los flujos de trabajo de reversión de App Runner. Este enfoque garantiza que ENIs permanezcan disponibles para una mayor investigación.

   a. Desde la AWS consola, inicie la consola EC2.

   b. En el panel de navegación izquierdo, en la agrupación Red y seguridad, seleccione Interfaces de red.

   c. Desplázate hasta las columnas Tipo de interfaz y Descripción para ubicarlas ENIs en las subredes asociadas al conector de VPC. Tendrán los siguientes patrones de nomenclatura.

      - Tipo de interfaz: fargate
      - Descripción: comienza con AWSAppRunner ENI(ejemplo: AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)

   d. Utilice las casillas de verificación al principio de las filas para seleccionar las ENIs que correspondan.

   e. En el menú Acciones, seleccione Crear registro de flujo.

   f. Introduzca la información en las solicitudes y seleccione Crear registro de flujo en la parte inferior de la página.

   g. Inspeccione el registro de flujo generado.

      - Si el tráfico entraba en el ENI cuando estaba probando la conexión, entonces el problema no está relacionado con la configuración del ENI. Es posible que haya problemas de configuración de red con otro recurso de tu AWS cuenta además de los servicios de App Runner. Ponte en contacto con AWS Support para obtener más ayuda.

- Si el tráfico no entraba en el ENI cuando estaba probando la conexión, le recomendamos que se ponga en contacto con AWS Support para comprobar si hay algún problema conocido con el servicio Fargate.

h. Utilice la herramienta Reachability Analyzer de red. Esta herramienta ayuda a determinar las configuraciones incorrectas de la red al identificar los componentes que se bloquean cuando no se puede acceder a una fuente de la ruta de la red virtual. Para obtener más información, consulte ¿Qué es el Reachability Analyzer? en la Guía de Reachability Analyzer de Amazon VPC.

   Introduzca el ENI de App Runner como origen y el ENI de RDS como destino.

7. Si no puede reducir aún más el problema o si sigue sin poder conectarse al RDS o al servicio descendente después de completar los pasos anteriores, le recomendamos que se ponga en contacto con AWS Support para obtener más ayuda.

Cuando no hay suficientes direcciones IP para lanzar instancias o escalar

> **Note**
>
> En el caso de los servicios públicos, App Runner no crea una interfaz de red elástica (ENI) en sus VPCs servidores, por lo que sus servicios públicos no se ven afectados por este cambio.

Esta guía le ayuda a resolver los errores de agotamiento de IP que pueden producirse en los servicios de App Runner con el acceso a la VPC para el tráfico saliente habilitado.

App Runner lanzará instancias en las subredes asociadas al conector de VPC. App Runner crea 1 ENI por instancia en la subred en la que se lanza la instancia. Cada ENI usa una IP privada en esa subred. Las subredes tienen un número fijo de subredes IPs disponibles, según el bloque CIDR asociado a esa subred. Si App Runner no encuentra subredes suficientes IPs para crear un ENI, no podrá lanzar nuevas instancias para su servicio de App Runner. Esto puede provocar problemas a la hora de ampliar los servicios. En esos casos, verás los registros de eventos de App Runner que indican que App Runner no puede encontrar las subredes disponibles IPs. Puede actualizar sus servicios siguiendo las instrucciones que aparecen a continuación para resolver dichos errores.

¿Cómo actualizar tus servicios para tener más disponibles IPs

La cantidad de direcciones IP disponibles en una subred se basa en el bloque CIDR asociado a esa subred. Los bloques CIDR asociados a una subred no se pueden actualizar después de su creación. Los conectores de VPC de App Runner tampoco se pueden actualizar una vez creados. Para ofrecer más servicios de App Runner con el acceso IPs a la VPC para el tráfico saliente habilitado:

1. Cree nuevas subredes con un bloque CIDR más grande.
2. Cree un nuevo conector de VPC con las nuevas subredes.
3. Actualice el servicio App Runner para usar el nuevo conector de VPC.

Cálculo IPs necesario para sus servicios

Antes de intentar crear nuevas subredes con bloques de CIDR más grandes, determina el número de subredes IPs que necesitarás en todos los servicios de App Runner. Te recomendamos que calcules la cantidad de conectores IPs necesaria en tu conector de la siguiente manera:

1. Para cada servicio con acceso a VPC para el tráfico saliente habilitado, anote el tamaño máximo (número máximo de instancias) en la configuración de escalado automático.
2. Sume los valores de todos los servicios.
3. Duplique esta suma para tener en cuenta las nuevas instancias lanzadas durante las implementaciones de color azul verdoso.

Ejemplo

Considere dos servicios A y B que utilizan el mismo conector de VPC.

1. El servicio A tiene el tamaño máximo configurado en 25.
2. El servicio B tiene un tamaño máximo configurado en 15.

IPsNecesario = 2 × (25 + 15) = 80

Asegúrese de que sus subredes tengan al menos 80 subredes disponibles IPs combinadas.

Cree nuevas subredes

1. Determine el tamaño de bloque CIDR necesario para IPv4 usar esta fórmula (tenga en cuenta que AWS reserva 5 IPs : tamaño de subred)

```
Number of available IP addresses = 2^(32 - prefix length) - 5
```

```
Example :
For 192.168.1.0/24:
Prefix length is 24
Number of available IP addresses = 2^(32 - 24) - 5 = 2^8-5 = 251 IP addresses

For 10.0.0.0/16:
Prefix length is 16
Number of available IP addresses = 2^(32 - 16) - 5 = 2^16-5 = 65,531 IP addresses

Quick reference:
/24 = 251 IP addresses
/16 = 65,531 IP addresses
```

2. Cree una nueva subred mediante la AWS EC2 CLI.

```
aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Ejemplo (crea una subred con IPs 4.096):

```
aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
```

3. Cree un nuevo conector de VPC. Consulte: Administrar el acceso a la VPC
4. Actualice sus servicios con el tráfico saliente a la VPC habilitada para usar este nuevo conector de VPC. App Runner empezará a usar las nuevas subredes una vez que se actualice el servicio.

> **Note**
>
> VPCs también están limitadas por el número de unidades disponibles IPs que pueden asignarse a las subredes mediante bloques CIDR. Si no puede crear subredes con bloques CIDR más grandes, es posible que deba actualizar su VPC con bloques CIDR secundarios antes de crear las nuevas subredes.

Adjuntar bloques CIDR secundarios a su VPC

Asocie un bloque CIDR secundario a esta VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Ejemplo:

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

Verificación

Una vez que haya actualizado su servicio. Puedes usar lo siguiente para verificar tu corrección

1. Supervisa los registros de eventos: monitorea los registros de eventos del servicio App Runner para comprobar que no aparecen nuevos errores de falta de disponibilidad de IP o ENI
2. Comprueba el escalado del servicio:
    1. Amplíe completamente el servicio cambiando el recuento mínimo de instancias en su configuración de escalado automático
    2. Compruebe que todas las instancias nuevas se lancen sin errores relacionados con la IP
    3. Supervise varios eventos de escalado para garantizar un rendimiento uniforme
3. Banner de consola: si utiliza la consola de administración de AWS, confirme que App Runner ya no muestre un banner que advierta que es insuficiente IPs.
4. Utilización de VPC e IP de subred:
    1. Utilice el panel de control de la VPC o los comandos de la CLI para comprobar el uso de las direcciones IP en las nuevas subredes.
    2. Confirma que aún queda un margen de disponibilidad adecuado una IPs vez que el servicio se haya ampliado

Dificultades comunes

Cuando abordes el problema del agotamiento de la IP en los servicios de App Runner, ten en cuenta estos posibles problemas:

1. Planificación inadecuada de las direcciones IP: subestimar las necesidades futuras de IP puede provocar problemas de agotamiento recurrentes. Lleve a cabo una planificación exhaustiva de la capacidad, teniendo en cuenta el posible crecimiento del servicio y los escenarios de uso máximo.
2. Omitir el uso de IP en toda la VPC: recuerde que otros servicios de AWS dentro de la misma VPC también consumen direcciones IP. Tenga en cuenta los requisitos de IP de todos los servicios al planificar las configuraciones de VPC y subred.
3. No actualizar los servicios: después de crear nuevas subredes o conectores de VPC, asegúrate de actualizar los servicios de App Runner para usar las nuevas configuraciones. De lo contrario, se seguirá utilizando el rango de IP agotado.
4. Malinterpretar las superposiciones de bloques CIDR: cuando añada bloques CIDR secundarios a una VPC, asegúrese de que no se superpongan con los bloques existentes. La superposición de bloques CIDR puede provocar conflictos de enrutamiento y ambigüedad en las direcciones IP.
5. Superar los límites de la VPC: tenga en cuenta que una VPC puede tener un máximo de 5 bloques CIDR (1 principal y 4 secundarios). Planifique la expansión del espacio de direcciones IP dentro de estas restricciones.
6. Ignorar la distribución de subredes AZ: al crear nuevas subredes, asegúrese de que estén distribuidas en varias zonas de disponibilidad para lograr una alta disponibilidad y tolerancia a los errores.
7. Ignorar los límites de la ENI: recuerde que hay límites en cuanto al número de unidades ENIs que se pueden adjuntar a las instancias. Compruebe que los límites de su cuenta de AWS se ajusten al uso planificado de la interfaz de red.

Si eres consciente de estas dificultades, puedes administrar tus recursos de VPC de manera más eficaz y evitar problemas de agotamiento de IP en tus servicios de App Runner.

Recursos adicionales

1. Documentación de AWS VPC
2. Comprensión de los bloques CIDR
3. Conectores VPC App Runner

Glosario

1. ENI: Elastic Network Interface, una interfaz de red virtual en AWS.
2. CIDR: enrutamiento entre dominios sin clase, un método para asignar direcciones IP.
3. Conector de VPC: un recurso que permite a App Runner conectarse a la VPC.

Seguridad en App Runner

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted 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 AWS los servicios en la Nube de AWS. AWS también le 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 de . Para obtener más información sobre los programas de cumplimiento aplicables AWS App Runner, consulte AWS Servicios incluidos en el ámbito de aplicación por programa de conformidad y AWS servicios incluidos .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a entender cómo aplicar el modelo de responsabilidad compartida al usar App Runner. En los temas siguientes, se muestra cómo configurar App Runner para cumplir con sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger los recursos de App Runner.

Temas

Protección de datos en App Runner

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en AWS App Runner. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre privacidad de datos y los . Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Para fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabajas con App Runner u otra aplicación Servicios de AWS mediante la consola, la API o los SDK. AWS CLI AWS Cualquier dato que introduzca

en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Temas

- Protección de datos mediante cifrado
- Privacidad del tráfico entre redes

Protección de datos mediante cifrado

AWS App Runner lee la fuente de la aplicación (imagen fuente o código fuente) del repositorio que especifique y la almacena para implementarla en su servicio. Para obtener más información, consulte Arquitectura y conceptos.

La protección de datos se refiere a la protección de los datos mientras están en tránsito (cuando viajan hacia y desde App Runner) y en reposo (mientras están almacenados en centros de AWS datos).

Para obtener más información sobre la protección de datos, consulte the section called “Protección de datos”.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Cifrado en tránsito

Puede proteger los datos en tránsito de dos maneras: cifrando la conexión mediante Transport Layer Security (TLS) o utilizando el cifrado del lado del cliente (en el que el objeto se cifra antes de enviarse). Ambos métodos son válidos para proteger los datos de la aplicación. Para proteger la conexión, hay que cifrarla mediante TLS siempre que la aplicación, sus desarrolladores y administradores y los usuarios finales envíen o reciban objetos. App Runner configura tu aplicación para recibir tráfico a través de TLS.

Client-side el cifrado no es un método válido para proteger la imagen o el código fuente que se proporciona a App Runner para su implementación. App Runner necesita acceder a la fuente de la aplicación, por lo que no se puede cifrar. Por lo tanto, asegúrate de proteger la conexión entre tu entorno de desarrollo o despliegue y App Runner.

Cifrado en reposo y administración de claves

Para proteger los datos inactivos de la aplicación, App Runner cifra todas las copias almacenadas de la imagen fuente o del paquete fuente de la aplicación. Al crear un servicio de App Runner, puede proporcionar un AWS KMS key. Si proporciona uno, App Runner utilizará la clave proporcionada para cifrar la fuente. Si no proporciona una, App Runner utilizará una Clave administrada de AWS en su lugar.

Para obtener más información sobre los parámetros de creación de servicios de App Runner, consulte CreateService. Para obtener información sobre AWS Key Management Service (AWS KMS), consulte la Guía para AWS Key Management Service desarrolladores.

Privacidad del tráfico entre redes

App Runner usa Amazon Virtual Private Cloud (Amazon VPC) para crear límites entre los recursos de la aplicación App Runner y controlar el tráfico entre ellos, la red local e Internet. Para obtener más información sobre la seguridad de Amazon VPC, consulte Privacidad del tráfico entre redes en Amazon VPC en la Guía del usuario de Amazon VPC.

Para obtener información sobre cómo asociar la aplicación App Runner a una Amazon VPC personalizada, consulte. the section called “Tráfico saliente”

Para obtener información sobre cómo proteger las solicitudes a App Runner mediante un punto de enlace de VPC, consulte. the section called “Puntos de conexión de VPC”

Para obtener más información sobre la protección de datos, consulte the section called “Protección de datos”.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Administración de identidad y acceso para App Runner

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 de App Runner. Puedes usar IAM sin coste adicional. Servicio de AWS

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Temas

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso a App Runner).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona App Runner con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Ejemplos de políticas basadas en la identidad de App Runner).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear un Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso completo a todos Servicios de AWS los recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver la lista completa de las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

AWS Para controlar el acceso, puede crear políticas y adjuntarlas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección 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 políticas JSON que se asocian a un recurso. Los ejemplos incluyen 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. Debe especificar una entidad principal en una política basada en recursos.

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 () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs 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 compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la descripción general de la 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 que pueden establecer los permisos máximos otorgados por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener 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 se AWS determina 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 funciona App Runner con IAM

Antes de usar IAM para administrar el acceso AWS App Runner, debes entender qué funciones de IAM están disponibles para usar con App Runner. Para obtener una visión general de cómo funcionan App Runner y otros AWS servicios con IAM, consulte AWS Servicios que funcionan con IAM en la Guía del usuario de IAM.

Para ver otros temas de seguridad de App Runner, consulte. Seguridad

Temas

- Políticas de App Runner basadas en la identidad
- Políticas de App Runner basadas en recursos

- Autorización basada en las etiquetas de App Runner
- Permisos de usuario de App Runner
- Funciones de IAM de App Runner

Políticas de App Runner basadas en la identidad

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. App Runner admite acciones, recursos y claves de condición específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

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. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones políticas en App Runner usan el siguiente prefijo antes de la acción:`apprunner:`. Por ejemplo, para conceder a alguien permiso para ejecutar una instancia de Amazon EC2 con la operación `RunInstances` de la API de Amazon EC2, debe incluir la acción `ec2:RunInstances` en la política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. App Runner define su propio conjunto de acciones que describen las tareas que puede realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "apprunner:Describe*"
```

Para ver una lista de las acciones de App Runner, consulte las acciones definidas AWS App Runner en la Referencia de autorización del servicio.

Recursos

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. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

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

Los recursos de App Runner tienen la siguiente estructura de ARN:

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

Para obtener más información sobre el formato de ARNs, consulte Nombres de recursos de Amazon (ARNs) y espacios de nombres de AWS servicios en. Referencia general de AWS

Por ejemplo, para especificar el `my-service` servicio en la declaración, utilice el siguiente ARN:

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

Para especificar todos los servicios que pertenecen a una cuenta específica, utilice el comodín (*):

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

Algunas acciones de App Runner, como las de creación de recursos, no se pueden realizar en un recurso específico. En dichos casos, debe utilizar el carácter comodín (*).

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

Para ver una lista de los tipos de recursos de App Runner y sus tipos ARNs, consulte los recursos definidos AWS App Runner 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 App Runner.

Claves de condición

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` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. 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.

App Runner admite el uso de algunas claves de condición globales. 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.

App Runner define un conjunto de claves de condición específicas del servicio. Además, App Runner admite el control de acceso basado en etiquetas, que se implementa mediante claves de condición. Para obtener más información, consulte the section called “Autorización basada en las etiquetas de App Runner”.

Para ver una lista de las claves de condición de App Runner, consulta las claves de condición AWS App Runner en la Referencia de autorización del servicio. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS App Runner.

Ejemplos

Para ver ejemplos de políticas basadas en la identidad de App Runner, consulte. Ejemplos de políticas basadas en la identidad de App Runner

Políticas de App Runner basadas en recursos

App Runner no admite políticas basadas en recursos.

Autorización basada en las etiquetas de App Runner

Puede adjuntar etiquetas a los recursos de App Runner o pasarlas en una solicitud a App Runner. 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 `apprunner:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Para

obtener más información sobre cómo etiquetar los recursos de App Runner, consultethe section called “Configuración”.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Controlar el acceso a los servicios de App Runner en función de las etiquetas.

Permisos de usuario de App Runner

Para usar App Runner, los usuarios de IAM necesitan permisos para las acciones de App Runner. Una forma habitual de conceder permisos a los usuarios es adjuntar una política a los usuarios o grupos de IAM. Para obtener más información sobre la administración de los permisos de los usuarios, consulte Cambiar los permisos de un usuario de IAM en la Guía del usuario de IAM.

App Runner proporciona dos políticas administradas que puede adjuntar a sus usuarios.

- `AWSAppRunnerReadOnlyAccess`— Otorga permisos para enumerar y ver detalles sobre los recursos de App Runner.
- `AWSAppRunnerFullAccess`— Otorga permisos a todas las acciones de App Runner.

Para un control más detallado de los permisos de los usuarios, puedes crear una política personalizada y adjuntarla a tus usuarios. Para obtener más información, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para ver ejemplos de políticas de usuario, consulte. the section called “Políticas de usuario”

Funciones de IAM de App Runner

Un rol de IAM es una entidad dentro de usted Cuenta de AWS que tiene permisos específicos.

Roles vinculados a servicios

Los roles vinculados a un servicio permiten a AWS los servicios acceder a los recursos de otros servicios para completar una acción en tu nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

App Runner admite funciones vinculadas a un servicio. Para obtener información sobre la creación o administración de funciones vinculadas a servicios de App Runner, consulte. the section called “Cómo utilizar roles vinculados a servicios”

Roles de servicio

Esta característica permite que un servicio asuma un rol de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto significa que un usuario de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

App Runner admite algunas funciones de servicio.

Rol de acceso

La función de acceso es una función que App Runner utiliza para acceder a las imágenes de Amazon Elastic Container Registry (Amazon ECR) de su cuenta. Es obligatorio para acceder a una imagen en Amazon ECR y no en Amazon ECR Public.

Antes de crear un servicio basado en una imagen en Amazon ECR, utilice IAM para crear un rol de servicio. Utilice la política gestionada `AWSAppRunnerServicePolicyForECRAccess`en su función de servicio. A continuación, puede transferir esta función a App Runner cuando llame a la CreateServiceAPI del AuthenticationConfigurationmiembro del SourceConfigurationparámetro o cuando utilice la consola de App Runner para crear un servicio.

> **Note**
>
> Si creas tu propia política personalizada para tu función de acceso, asegúrate `"Resource": "*"` de especificar la `ecr:GetAuthorizationToken` acción. Los tokens se pueden usar para acceder a cualquier registro de Amazon ECR al que tenga acceso.

Cuando cree su función de acceso, asegúrese de añadir una política de confianza que declare al responsable del servicio de App Runner `build.apprunner.amazonaws.com` como una entidad de confianza.

Política de confianza para un rol de acceso

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "build.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Si utilizas la consola de App Runner para crear un servicio, la consola puede crear automáticamente un rol de acceso para ti y elegirlo para el nuevo servicio. La consola también muestra otras funciones de tu cuenta y, si lo deseas, puedes seleccionar una función diferente.

Rol de instancia

El rol de instancia es un rol opcional que App Runner usa para proporcionar permisos a las acciones de AWS servicio que necesitan las instancias informáticas de tu servicio. Debes proporcionar un rol de instancia a App Runner si el código de tu aplicación llama a AWS actions (APIs). Incorpora los permisos necesarios en tu rol de instancia o crea tu propia política personalizada y úsala en el rol de instancia. No tenemos forma de anticipar qué llamadas utilizará tu código. Por lo tanto, no ofrecemos una política gestionada para este fin.

Antes de crear un servicio de App Runner, usa IAM para crear un rol de servicio con las políticas personalizadas o integradas necesarias. A continuación, puedes transferir esta función a App Runner como función de instancia cuando llames a la CreateServiceAPI del `InstanceRoleArn` miembro del InstanceConfigurationparámetro o cuando utilices la consola de App Runner para crear un servicio.

Al crear el rol de instancia, asegúrate de agregar una política de confianza que declare al principal del servicio de App Runner `tasks.apprunner.amazonaws.com` como una entidad de confianza.

Política de confianza para un rol de instancia

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "tasks.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Si utilizas la consola de App Runner para crear un servicio, la consola mostrará una lista de las funciones de tu cuenta y podrás seleccionar la función que creaste para ello.

Para obtener información sobre la creación de un servicio, consultethe section called “Creación”.

Ejemplos de políticas basadas en la identidad de App Runner

De forma predeterminada, los usuarios y roles de IAM no tienen permiso para crear o modificar recursos. AWS App Runner Tampoco pueden realizar tareas con la AWS API Consola de administración de AWS AWS CLI, o. Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Temas

- Prácticas recomendadas sobre las políticas
- Políticas de usuario
- Controlar el acceso a los servicios de App Runner en función de las etiquetas

Prácticas recomendadas sobre las políticas

Las políticas basadas en la identidad determinan si alguien puede crear, acceder o eliminar los recursos de App Runner de su cuenta. 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 tarea 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 CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza 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 Validación de políticas con el Analizador de acceso de IAM en la Guía del 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 exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Políticas de usuario

Para acceder a la consola de App Runner, los usuarios de IAM deben tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de App Runner que tiene en su Cuenta de AWS cuenta. Si creas una política basada en la identidad que sea más restrictiva que los permisos mínimos requeridos, la consola no funcionará según lo previsto para los usuarios con esa política.

App Runner proporciona dos políticas administradas que puedes adjuntar a tus usuarios.

- `AWSAppRunnerReadOnlyAccess`— Otorga permisos para enumerar y ver detalles sobre los recursos de App Runner.
- `AWSAppRunnerFullAccess`— Otorga permisos a todas las acciones de App Runner.

Para garantizar que los usuarios puedan usar la consola de App Runner, adjunta, como mínimo, la política `AWSAppRunnerReadOnlyAccess` administrada a los usuarios. En su lugar, puede adjuntar la política `AWSAppRunnerFullAccess` administrada o agregar permisos adicionales específicos para permitir a los usuarios crear, modificar y eliminar recursos. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite el acceso únicamente a las acciones que coincidan con la operación de la API que deseas que realicen los usuarios.

Los siguientes ejemplos muestran las políticas de usuario personalizadas. Puede utilizarlos como puntos de partida para definir sus propias políticas de usuario personalizadas. Copie el ejemplo o elimine las acciones, reduzca los recursos y añada condiciones.

Ejemplo: política de usuario de administración de consolas y conexiones

Este ejemplo de política permite el acceso a la consola y permite la creación y administración de conexiones. No permite la creación ni la administración del servicio App Runner. Se puede adjuntar a un usuario cuya función sea administrar el acceso del servicio App Runner a los activos de código fuente.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner:DeleteConnection"
      ],
      "Resource": "*"
    }
  ]
}
```

Ejemplo: políticas de usuario que usan claves de condición

Los ejemplos de esta sección muestran los permisos condicionales que dependen de algunas propiedades de los recursos o parámetros de acción.

Este ejemplo de política permite crear un servicio de App Runner, pero deniega el uso de una conexión denominada`prod`.

JSON

```
  { "Version":"2012-10-17",
  "Statement":
     [ { "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
         "Effect": "Allow",
         "Action": "apprunner:CreateService",
         "Resource": "*",
         "Condition":
            { "ArnNotLike":
               {"apprunner:ConnectionArn":"arn:aws:apprunner:*:*:connection/prod/
*"}
            }
       }
     ]
   }
```

Esta política de ejemplo permite actualizar un servicio de App Runner nombrado `preprod` únicamente con una configuración de autoescalado denominada`preprod`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
            "Effect": "Allow",
            "Action": "apprunner:UpdateService",
            "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
            "Condition": {
                "ArnLike": {
                    "apprunner:AutoScalingConfigurationArn":
 "arn:aws:apprunner:us-east-1:*:autoscalingconfiguration/preprod/*"
                }
            }
        }
    ]
}
```

Controlar el acceso a los servicios de App Runner en función de las etiquetas

Puedes usar las condiciones de tu política basada en la identidad para controlar el acceso a los recursos de App Runner en función de las etiquetas. En este ejemplo, se muestra cómo se puede crear una política que permita eliminar un servicio de App Runner. Sin embargo, los permisos solo se conceden si la etiqueta de servicio `Owner` tiene el valor del nombre de usuario de dicho usuario. Esta política también proporciona los permisos necesarios para llevar a cabo esta acción en la consola.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
```

```
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:us-east-1:*:service/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

También puede asociar esta política al usuario de IAM en su cuenta. Si un usuario llamado `richard-roe` intenta eliminar un servicio de App Runner, el servicio debe estar etiquetado `0wner=richard-roe` o`owner=richard-roe`. De lo contrario, se le deniega el acceso. La clave de la etiqueta de condición `0wner` coincide con los nombres de las claves de condición `0wner` y `owner` porque no distinguen entre mayúsculas y minúsculas. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

Uso de funciones vinculadas a servicios para App Runner

AWS App Runner 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 que está vinculado directamente a App Runner. App Runner predefine las funciones vinculadas al servicio e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en tu nombre.

Temas

- Uso de roles para la administración
- Uso de roles para la creación de redes

Uso de roles para la administración

AWS App Runner 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 que está vinculado directamente a App

Runner. App Runner predefine las funciones vinculadas al servicio e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en tu nombre.

Un rol vinculado a un servicio facilita la configuración de App Runner, ya que no es necesario añadir manualmente los permisos necesarios. App Runner define los permisos de sus funciones vinculadas al servicio y, a menos que se defina lo contrario, solo App Runner 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 tus recursos de App Runner, ya que no puedes eliminar inadvertidamente el permiso de acceso a los recursos.

Para obtener información acerca de 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 Rol vinculado a un servicio. 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 App Runner

App Runner usa el rol vinculado al servicio denominado. AWSServiceRoleForAppRunner

El rol permite a App Runner realizar las siguientes tareas:

- Envía los registros a los grupos de CloudWatch registros de Amazon Logs.
- Crea reglas de Amazon CloudWatch Events para suscribirte a los envíos de imágenes del Amazon Elastic Container Registry (Amazon ECR).
- Envíe la información de rastreo a. AWS X-Ray

El rol AWSService RoleForAppRunner vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `apprunner.amazonaws.com`

Las políticas de permisos del rol AWSService RoleForAppRunner vinculado al servicio contienen todos los permisos que App Runner necesita para realizar acciones en tu nombre:

- Política gestionada `AppRunnerServiceRolePolicy`
- Política de rastreo de rayos X: consulte el siguiente contenido de la política.

Política de rastreo por rayos X

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "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.

Crear un rol vinculado a un servicio para App Runner

No necesita crear manualmente un rol vinculado a servicios. Al crear un servicio de App Runner en la Consola de administración de AWS, la o la AWS API AWS CLI, App Runner 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 creas un servicio de App Runner, App Runner vuelve a crear el rol vinculado al servicio para ti.

Edición de un rol vinculado a un servicio para App Runner

App Runner no permite editar el rol vinculado al AWSService RoleForAppRunner 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 Edición de un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminar un rol vinculado a un servicio para App Runner

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 limpiar el rol vinculado a servicios antes de eliminarlo manualmente.

Limpiar un rol vinculado a servicios

Antes de que pueda utilizar IAM para eliminar un rol vinculado a servicios, primero debe eliminar los recursos que utiliza el rol.

En App Runner, esto significa eliminar todos los servicios de App Runner de tu cuenta. Para obtener información sobre cómo eliminar los servicios de App Runner, consultethe section called “Eliminación”.

> **Note**
>
> Si el servicio App Runner utiliza el rol al intentar eliminar los recursos, es posible que la eliminación no se realice correctamente. En tal caso, espere unos minutos e intente de nuevo la operación.

Eliminar manualmente el rol vinculado al servicio

Usa la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSService RoleForAppRunner servicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones compatibles con las funciones vinculadas al servicio de App Runner

App Runner admite el uso de roles vinculados al servicio en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Puntos de conexión y cuotas de AWS App Runner en la Referencia general de AWS.

Uso de roles para la creación de redes

AWS App Runner 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 que está vinculado directamente a App Runner. App Runner predefine las funciones vinculadas al servicio e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en tu nombre.

Un rol vinculado a un servicio facilita la configuración de App Runner, ya que no es necesario añadir manualmente los permisos necesarios. App Runner define los permisos de sus funciones vinculadas al servicio y, a menos que se defina lo contrario, solo App Runner 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 tus recursos de App Runner, ya que no puedes eliminar inadvertidamente el permiso de acceso a los recursos.

Para obtener información acerca de 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 Rol vinculado a un servicio. 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 App Runner

App Runner usa el rol vinculado al servicio denominado. AWSServiceRoleForAppRunnerNetworking

El rol permite a App Runner realizar las siguientes tareas:

- Adjunta una VPC a tu servicio de App Runner y administra las interfaces de red.

El rol AWSService RoleForAppRunnerNetworking vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `networking.apprunner.amazonaws.com`

La política de permisos de roles denominada `AppRunnerNetworkingServiceRolePolicy`contiene todos los permisos que App Runner necesita para completar acciones en tu nombre.

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.

Crear un rol vinculado a un servicio para App Runner

No necesita crear manualmente un rol vinculado a servicios. Al crear un conector de VPC en la Consola de administración de AWS, la o la AWS API AWS CLI, App Runner 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. Al crear un conector de VPC, App Runner vuelve a crear el rol vinculado al servicio para usted.

Edición de un rol vinculado a un servicio para App Runner

App Runner no permite editar el rol vinculado al AWSService RoleForAppRunnerNetworking 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 Edición de un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminar un rol vinculado a un servicio para App Runner

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 limpiar el rol vinculado a servicios antes de eliminarlo manualmente.

Limpiar un rol vinculado a un servicio

Antes de que pueda utilizar IAM para eliminar un rol vinculado a servicios, primero debe eliminar los recursos que utiliza el rol.

En App Runner, esto significa desasociar los conectores de VPC de todos los servicios de App Runner de su cuenta y eliminar los conectores de VPC. Para obtener más información, consulte the section called “Tráfico saliente”.

> **Note**
>
> Si el servicio App Runner utiliza el rol cuando intentas eliminar los recursos, es posible que la eliminación no se realice correctamente. En tal caso, espere unos minutos e intente de nuevo la operación.

Eliminar manualmente la función vinculada al servicio

Usa la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al AWSService RoleForAppRunnerNetworking servicio. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones compatibles con las funciones vinculadas al servicio de App Runner

App Runner admite el uso de roles vinculados al servicio en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Puntos de conexión y cuotas de AWS App Runner en la Referencia general de AWS.

AWS políticas gestionadas para AWS App Runner

Una política AWS administrada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

App Runner actualiza las políticas AWS administradas

Consulta los detalles sobre las actualizaciones de las políticas AWS administradas de App Runner desde que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS de la página del historial de documentos de App Runner.

Cambio	Descripción	Fecha
AWSAppRunnerReadOnlyAccess: política nueva	App Runner agregó una nueva política que permite a los usuarios enumerar y ver detalles sobre los recursos de App Runner.	24 de febrero de 2022
AWSAppRunnerFullAccess: actualización de una política actual	App Runner actualizó la lista de recursos de la `iam:CreateServiceLinkedRole` acción para permitir la creación de un rol `AWSServiceRoleForAppRunnerNetworking` vinculado al servicio.	8 de febrero de 2022
AppRunnerNetworkingServiceRolePolicy: política nueva	App Runner agregó una nueva política que permite a App Runner realizar llamadas a Amazon Virtual Private Cloud para conectar una VPC a su servicio de App Runner y administrar las interfaces de red en nombre de los servicios de App Runner. La política se usa en la función vinculada al `AWSServiceRoleForAppRunnerNetworking` servicio.	8 de febrero de 2022
AWSAppRunnerFullAccess: política nueva	App Runner agregó una nueva política que permite a los usuarios realizar todas las acciones de App Runner.	10 de enero de 2022

Cambio	Descripción	Fecha
AppRunnerServiceRolePolicy: política nueva	App Runner agregó una nueva política que permite a App Runner realizar llamadas a Amazon CloudWatch Logs y Amazon CloudWatch Events en nombre de los servicios de App Runner. La política se usa en la función `AWSServiceRoleForAppRunner` vinculada al servicio.	1 de marzo de 2021
AWSAppRunnerServicePolicyForECRAccess: política nueva	App Runner agregó una nueva política que permite a App Runner acceder a las imágenes de Amazon Elastic Container Registry (Amazon ECR) de su cuenta.	1 de marzo de 2021
App Runner comenzó a rastrear los cambios	App Runner comenzó a realizar un seguimiento de los cambios en sus políticas AWS gestionadas.	1 de marzo de 2021

Solución de problemas de identidad y acceso a App Runner

Usa la siguiente información para ayudarte a diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con un AWS App Runner IAM.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Temas

- No estoy autorizado a realizar ninguna acción en App Runner
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de App Runner

No estoy autorizado a realizar ninguna acción en App Runner

Si Consola de administración de AWS te indica que no estás autorizado a realizar una acción, ponte en contacto con tu administrador para obtener ayuda. Su administrador es la persona que le proporcionó sus credenciales de AWS inicio de sesión.

El siguiente ejemplo de error se produce cuando un usuario de IAM llamado `marymajor` intenta usar la consola para ver detalles sobre un servicio de App Runner pero no tiene `apprunner:DescribeService` permisos.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

En este caso, Mary pide al administrador que actualice sus políticas para poder acceder al *my-example-service* recurso mediante la `apprunner:DescribeService` acción.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de App Runner

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si App Runner es compatible con estas funciones, consulte. Cómo funciona App Runner 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 conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y supervisión en App Runner

El monitoreo es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de su AWS App Runner servicio. La recopilación de datos de supervisión de todas las partes de la AWS solución le permite depurar más fácilmente un error en caso de que se produzca. App Runner se integra con varias AWS herramientas para monitorear tus servicios de App Runner y responder a posibles incidentes.

CloudWatch Alarmas Amazon

Con CloudWatch las alarmas de Amazon, puedes ver una métrica de servicio durante un período de tiempo que especifiques. Si la métrica supera un umbral determinado durante un número determinado de períodos, recibirás una notificación.

App Runner recopila una variedad de métricas sobre el servicio en su conjunto y las instancias (unidades de escalado) en las que se ejecuta el servicio web. Para obtener más información, consulte Métricas (CloudWatch).

Registros de aplicaciones

App Runner recopila el resultado del código de la aplicación y lo transmite a Amazon CloudWatch Logs. El contenido de este resultado depende de usted. Por ejemplo, puede incluir registros detallados de las solicitudes realizadas a su servicio web. Estos registros pueden resultar útiles en las auditorías de seguridad y acceso. Para obtener más información, consulte Registros (CloudWatch registros).

AWS CloudTrail registros de acciones

App Runner está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en App Runner. CloudTrail captura todas las llamadas a la API de App Runner como eventos. Puede ver los eventos más recientes en la CloudTrail consola y crear un registro para permitir la entrega continua de CloudTrail eventos a un bucket de Amazon Simple Storage Service (Amazon S3). Para obtener más información, consulte Acciones de la API (CloudTrail).

Validación de conformidad para App Runner

Third-party los auditores evalúan la seguridad y el cumplimiento AWS App Runner como parte de varios programas de AWS cumplimiento. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al utilizarlos Servicios de AWS, consulte AWS la documentación de seguridad.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Resiliencia en App Runner

La infraestructura AWS global se basa en zonas Regiones de AWS de disponibilidad. Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad 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 zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

AWS App Runner administra y automatiza el uso de la infraestructura AWS global en su nombre. Al utilizar App Runner, se beneficia de los mecanismos de disponibilidad y tolerancia a errores que AWS ofrece.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Seguridad de la infraestructura en AWS App Runner

Como servicio gestionado, AWS App Runner está protegido por los procedimientos de seguridad de red AWS global que se describen en el documento técnico Amazon Web Services: Overview of Security Processes.

Utiliza las llamadas a la API AWS publicadas para administrar y operar App Runner a través de la red. Los clientes que llamen a App Runner APIs deben ser compatibles con Transport Layer Security (TLS) 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos. Estos requisitos no se aplican a los puntos finales de las aplicaciones de App Runner.

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 puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Uso de App Runner con puntos finales de VPC

Es posible que su AWS aplicación integre AWS App Runner servicios con otros Servicios de AWS que se ejecuten en una VPC desde Amazon Virtual Private Cloud (Amazon VPC). Es posible que algunas partes de la aplicación realicen solicitudes a App Runner desde la VPC. Por ejemplo, puedes utilizarla AWS CodePipeline para realizar una implementación continua en tu servicio de App Runner. Una forma de mejorar la seguridad de tu aplicación es enviar estas solicitudes de App Runner (y las solicitudes a otras Servicios de AWS) a través de un punto final de VPC.

Con un punto de enlace de VPC, puede conectar de forma privada su VPC a los servicios de punto final de Servicios de AWS VPC compatibles y con la tecnología de. AWS PrivateLink No necesitas una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. Direct Connect

Los recursos de tu VPC no utilizan direcciones IP públicas para interactuar con los recursos de App Runner. El tráfico entre tu VPC y App Runner no sale de la red de Amazon. Para obtener más información sobre los puntos de enlace de la VPC, consulte los puntos de enlace de la VPC en la guía.AWS PrivateLink

> **Note**
>
> De forma predeterminada, la aplicación web del servicio App Runner se ejecuta en una VPC que App Runner proporciona y configura. Esta VPC es pública. Significa que está conectada

a Internet. Si lo desea, puede asociar su aplicación a una VPC personalizada. Para obtener más información, consulte the section called “Tráfico saliente”.
Puede configurar sus servicios para acceder a Internet AWS APIs, incluso cuando su servicio esté conectado a una VPC. Para obtener instrucciones sobre cómo habilitar el acceso público a Internet para el tráfico saliente de la VPC, consulte. the section called “Consideraciones a la hora de seleccionar una subred”
App Runner no admite la creación de un punto final de VPC para tu aplicación.

Configuración de un punto final de VPC para App Runner

Para crear el punto de enlace de la VPC de la interfaz para el servicio App Runner en su VPC, siga el procedimiento de creación de un punto final de interfaz de la guía.AWS PrivateLink En Nombre de servicio, elija `com.amazonaws.region.apprunner`.

Consideraciones sobre la privacidad de la red de VPC

Important

El uso de un punto de enlace de VPC para App Runner no garantiza que todo el tráfico de la VPC permanezca fuera de Internet. La VPC puede ser pública. Además, es posible que algunas partes de la solución no usen puntos de enlace de VPC para realizar AWS llamadas a la API. Por ejemplo, Servicios de AWS podría llamar a otros servicios mediante sus puntos de enlace públicos. Si la solución de su VPC requiere privacidad de tráfico, lea esta sección.

Para garantizar la privacidad del tráfico de red en su VPC, tenga en cuenta lo siguiente:

- Habilite el nombre DNS: es posible que algunas partes de la aplicación sigan enviando solicitudes a App Runner a través de Internet mediante el dispositivo de punto final público `apprunner.region.amazonaws.com`. Si su VPC está configurada con acceso a Internet, estas solicitudes se realizan correctamente sin ninguna indicación para usted. Para evitarlo, asegúrate de que la opción Habilitar el nombre DNS esté habilitada al crear el punto final. De forma predeterminada, se establece en true. Esto añade una entrada DNS en la VPC que asigna el punto de conexión del servicio público al punto de conexión de la VPC de tipo interfaz.
- Configure los puntos finales de la VPC para servicios adicionales: es posible que su solución envíe solicitudes a otros. Servicios de AWS Por ejemplo, AWS CodePipeline podría enviar solicitudes a.

AWS CodeBuild Configure los puntos de enlace de VPC para estos servicios y habilite los nombres de DNS en estos puntos de enlace.

- Configurar una VPC privada: si es posible (si la solución no necesita acceso a Internet en absoluto), configure la VPC como privada, lo que significa que no tiene conexión a Internet. Esto garantiza que un punto final de VPC faltante provoque un error visible, de modo que se pueda añadir el punto final que falta.

Uso de políticas de punto de conexión para controlar el acceso con puntos de conexión de la VPC

App Runner admite políticas de puntos finales de VPC. De forma predeterminada, se permite el acceso total a App Runner a través del punto final de la interfaz. Las políticas de puntos de enlace de la VPC se pueden usar para controlar qué directores de AWS pueden acceder al punto de enlace de App Runner. Como alternativa, puede asociar un grupo de seguridad a las interfaces de red de los puntos finales para controlar el tráfico a App Runner a través del punto final de la interfaz.

Integración con el punto final de la interfaz

Compatible con App Runner AWS PrivateLink, que proporciona conectividad privada a App Runner y elimina la exposición del tráfico a Internet. Para permitir que su aplicación envíe solicitudes a App Runner mediante AWS PrivateLink, configure un tipo de punto final de VPC conocido como punto final de interfaz. Para obtener más información, consulte Puntos de conexión de VPC de interfaz (AWS PrivateLink) en la Guía de AWS PrivateLink .

Análisis de configuración y vulnerabilidad en App Runner

AWS y nuestros clientes comparten la responsabilidad de lograr un alto nivel de seguridad y conformidad de los componentes de software. Para obtener más información, consulte el AWS Modelo de responsabilidad compartida de https://aws.amazon.com/compliance/shared-responsibility-model/.

Imágenes de contenedores de parches

Aplicar parches a la imagen del contenedor forma parte de la responsabilidad del cliente en el modelo de seguridad compartida. El propietario de la imagen es responsable de actualizar y corregir periódicamente la imagen del contenedor. Recomendamos establecer un programa de

rutina para comprobar y aplicar las actualizaciones a las imágenes del contenedor. Para obtener más información sobre cómo escanear sus imágenes en busca de vulnerabilidades, consulte la documentación de AWS App Runner

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Mejores prácticas de seguridad para App Runner

AWS App Runner proporciona varias características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen 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, no como normas.

Para ver otros temas de seguridad de App Runner, consulteSeguridad.

Prácticas recomendadas de seguridad preventiva

Los controles de seguridad preventivos intentan evitar incidentes antes de que ocurran.

Implementación del acceso a los privilegios mínimos

App Runner proporciona políticas gestionadas AWS Identity and Access Management (IAM) para los usuarios de IAM y el rol de acceso. Estas políticas administradas especifican todos los permisos que pueden ser necesarios para el correcto funcionamiento del servicio de App Runner.

Es posible que su aplicación no requiera todos los permisos de nuestras políticas administradas. Puede personalizarlas y conceder solo los permisos necesarios para que sus usuarios y su servicio de App Runner realicen sus tareas. Esto es especialmente importante para las políticas de usuario, donde diferentes roles de usuario pueden tener necesidades de permiso distintas. La implementación del acceso con privilegios mínimos es esencial a la hora de reducir los riesgos de seguridad y el impacto que podrían causar los errores o los intentos malintencionados.

Escanear las imágenes para detectar vulnerabilidades

Puede utilizar los ECR de Amazon APIs para identificar las vulnerabilidades de software en las imágenes de sus contenedores. Para obtener más información, consulte la documentación de Amazon ECR.

Prácticas recomendadas de detección de seguridad

Los controles de detección de seguridad identifican las infracciones de seguridad tras producirse. Pueden ayudarte a detectar una posible amenaza o incidente de seguridad.

Implementar la supervisión

La supervisión es una parte importante del mantenimiento de la fiabilidad, la seguridad, la disponibilidad y el rendimiento de las soluciones de App Runner. AWS proporciona varias herramientas y servicios para ayudarte a supervisar tus AWS servicios.

A continuación se muestran algunos ejemplos de elementos que supervisar:

- CloudWatch Métricas de Amazon para App Runner: configura alarmas para las métricas clave de App Runner y para las métricas personalizadas de tu aplicación. Para obtener más información, consulte Métricas (CloudWatch).
- AWS CloudTrail entradas: realiza un seguimiento de las acciones que podrían afectar a la disponibilidad, como `PauseService` o`DeleteConnection`. Para obtener información, consulte Acciones de la API (CloudTrail).

AWS Glosario

Para obtener la AWS terminología más reciente, consulte el AWS glosario de la Glosario de AWS Referencia.