aws

Entwicklerhandbuch

AWS App Runner

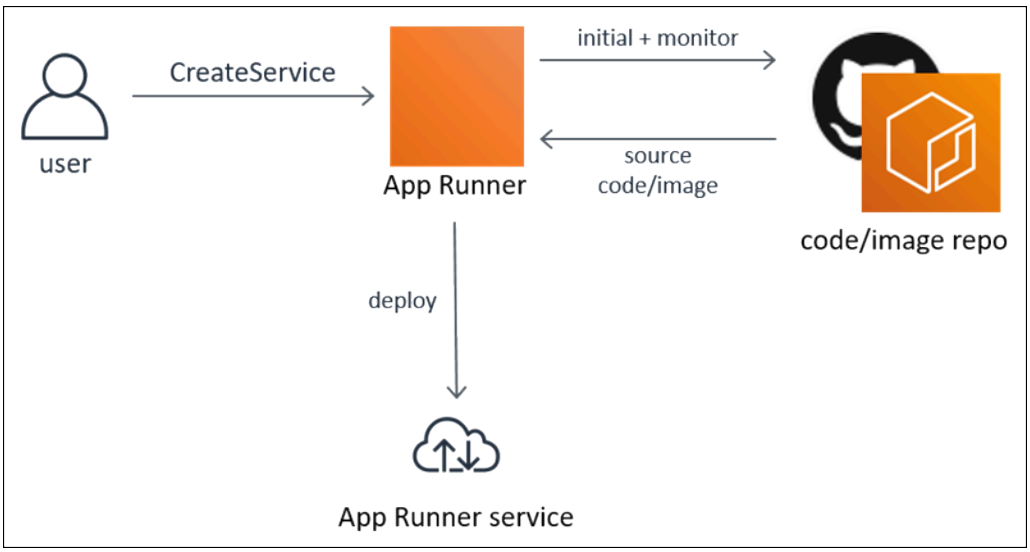
AWS App Runner: Entwicklerhandbuch

Table of Contents

Was ist AWS App Runner?

AWS App Runner ist ein AWS Service, der eine schnelle, einfache und kostengünstige Möglichkeit bietet, Quellcode oder ein Container-Image direkt in einer skalierbaren und sicheren Webanwendung in der AWS Cloud bereitzustellen. Sie müssen sich nicht mit neuen Technologien vertraut machen, entscheiden, welchen Rechendienst Sie verwenden möchten, oder wissen, wie AWS Ressourcen bereitgestellt und konfiguriert werden.

App Runner stellt eine direkte Verbindung zu Ihrem Code- oder Image-Repository her. Es bietet eine automatische Integrations- und Bereitstellungspipeline mit vollständig verwalteten Abläufen, hoher Leistung, Skalierbarkeit und Sicherheit.



Für wen ist App Runner gedacht?

Wenn Sie ein Entwickler sind, können Sie App Runner verwenden, um den Prozess der Bereitstellung einer neuen Version Ihres Code- oder Image-Repositorys zu vereinfachen.

Für Betriebsteams ermöglicht App Runner automatische Bereitstellungen jedes Mal, wenn ein Commit in das Code-Repository oder eine neue Container-Image-Version in das Image-Repository übertragen wird.

Zugreifen auf App Runner

Sie können Ihre App Runner-Dienstbereitstellungen über eine der folgenden Schnittstellen definieren und konfigurieren:

- App Runner-Konsole — Bietet eine Weboberfläche für die Verwaltung Ihrer App Runner-Dienste.
- App Runner-API — Stellt eine RESTful API zur Ausführung von App Runner-Aktionen bereit. Weitere Informationen finden Sie unter AWS App Runner -API-Referenz.
- AWS Befehlszeilenschnittstelle (AWS CLI) — Stellt Befehle für eine Vielzahl von AWS Diensten bereit, einschließlich Amazon VPC, und wird unter Windows, macOS und Linux unterstützt. Weitere Informationen finden Sie unter AWS Command Line Interface.
- AWS SDKs— Bietet sprachspezifische Funktionen APIs und kümmert sich um viele Verbindungsdetails, wie z. B. die Berechnung von Signaturen, die Bearbeitung von Wiederholungsversuchen von Anfragen und die Fehlerbehandlung. Weitere Informationen finden Sie unter AWS SDKs.

Preise für App Runner

App Runner bietet eine kostengünstige Möglichkeit, Ihre Anwendung auszuführen. Sie zahlen nur für Ressourcen, die Ihr App Runner-Dienst verbraucht. Ihr Service wird auf weniger Recheninstanzen herunterskaliert, wenn der Anforderungsverkehr geringer ist. Sie haben die Kontrolle über die Skalierbarkeitseinstellungen: die niedrigste und höchste Anzahl von bereitgestellten Instanzen und die höchste Last, die eine Instanz verarbeitet.

Weitere Informationen zur automatischen Skalierung von App Runner finden Sie unterthe section called “Auto-Scaling”.

Preisinformationen finden Sie unter AWS App Runner Preise.

Was kommt als Nächstes

In den folgenden Themen erfahren Sie, wie Sie mit App Runner beginnen können:

- Einrichtung— Führen Sie die erforderlichen Schritte für die Verwendung von App Runner aus.
- Erste Schritte— Stellen Sie Ihre erste Anwendung in App Runner bereit.

Einrichtung für App Runner

Wenn Sie ein neuer AWS Kunde sind, müssen Sie die auf dieser Seite aufgeführten Einrichtungsvoraussetzungen erfüllen, bevor Sie mit der Nutzung beginnen AWS App Runner.

Für diese Einrichtungsverfahren verwenden Sie den AWS Identity and Access Management (IAM) - Service. Umfassende Informationen zu IAM finden Sie in den folgenden Referenzmaterialien:

- AWS Identity and Access Management (IAM)
- IAM Benutzerhandbuch

Melde dich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Ein Teil des Anmeldevorgangs umfasst den Empfang eines Telefonanrufs oder einer Textnachricht und die Eingabe eines Bestätigungscodes auf der Telefontastatur.

   Wenn Sie sich für eine anmelden AWS-Konto, wird eine Root-Benutzer des AWS-Kontoserstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können Ihre aktuellen Kontoaktivitäten jederzeit einsehen und Ihr Konto verwalten, indem Sie zu https://aws.amazon.com/gehen und Mein Konto auswählen.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerhandbuch.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des interagieren möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Ref erenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldeinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Was kommt als Nächstes

Sie haben die erforderlichen Schritte abgeschlossen. Informationen zur Bereitstellung Ihrer ersten Anwendung in App Runner finden Sie unterErste Schritte.

Erste Schritte mit App Runner

AWS App Runner ist ein AWS Service, der eine schnelle, einfache und kostengünstige Möglichkeit bietet, ein vorhandenes Container-Image oder Quellcode direkt in einen laufenden Webservice in der umzuwandeln AWS Cloud.

In diesem Tutorial erfahren Sie, wie Sie Ihre Anwendung für einen App Runner-Dienst bereitstellen können. AWS App Runner Es führt Sie durch die Konfiguration des Quellcodes und der Bereitstellung, den Service-Build und die Service-Laufzeit. Außerdem wird gezeigt, wie eine Codeversion bereitgestellt, eine Konfigurationsänderung vorgenommen und Protokolle angezeigt werden. Schließlich zeigt das Tutorial, wie Sie die Ressourcen bereinigen, die Sie erstellt haben, während Sie den Anweisungen des Tutorials folgen.

Themen

Voraussetzungen

Bevor Sie mit dem Tutorial beginnen, sollten Sie unbedingt die folgenden Maßnahmen ergreifen:

1. Schließen Sie die Einrichtungsschritte unter abEinrichtung.
2. Entscheide, ob du entweder mit einem GitHub Repository oder einem Bitbucket-Repository arbeiten möchtest.
   - Um mit einem Bitbucket zu arbeiten, erstelle zunächst ein Bitbucket-Konto, falls du noch keines hast. Wenn du neu bei Bitbucket bist, findest du weitere Informationen unter Erste Schritte mit Bitbucket in der Bitbucket Cloud-Dokumentation.
   - Um damit zu arbeiten GitHub, erstelle ein GitHubKonto, falls du noch keines hast. Falls Sie noch nicht damit vertraut sind GitHub, finden Sie weitere Informationen unter Erste Schritte GitHub in der GitHubDokumentation.

> **Note**
>
> Sie können von Ihrem Konto aus Verbindungen zu mehreren Repository-Anbietern herstellen. Wenn du also die Bereitstellung sowohl von einem als auch von einem GitHub Bitbucket-Repository aus durchgehen möchtest, kannst du dieses Verfahren wiederholen. Erstellen Sie beim nächsten Mal einen neuen App Runner-Dienst und erstellen Sie eine neue Kontoverbindung für den anderen Repository-Anbieter.

3. Erstellen Sie ein Repository in Ihrem Repository-Anbieter-Konto. In dieser Anleitung wird der Name des Repositorys verwendet`python-hello`. Erstellen Sie Dateien im Stammverzeichnis des Repositorys mit den in den folgenden Beispielen angegebenen Namen und Inhalten.

Dateien für das **`python-hello`** Beispiel-Repository

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)
```

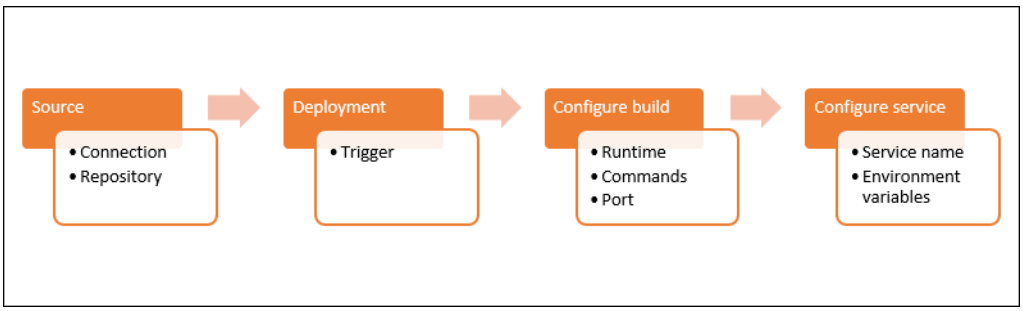
```
    server.serve_forever()
```

Schritt 1: Erstellen Sie einen App Runner-Dienst

In diesem Schritt erstellst du einen App Runner-Dienst auf der Grundlage des Beispiel-Quellcode-Repositorys, das du auf GitHub oder Bitbucket als Teil davon the section called “Voraussetzungen” erstellt hast. Das Beispiel enthält eine einfache Python-Website. Dies sind die wichtigsten Schritte, die Sie unternehmen, um einen Service zu erstellen:

1. Konfigurieren Sie Ihren Quellcode.
2. Konfigurieren Sie die Quellbereitstellung.
3. Konfigurieren Sie den Build der Anwendung.
4. Konfigurieren Sie Ihren Dienst.
5. Überprüfe und bestätige.

Das folgende Diagramm zeigt die Schritte zum Erstellen eines App Runner-Dienstes:

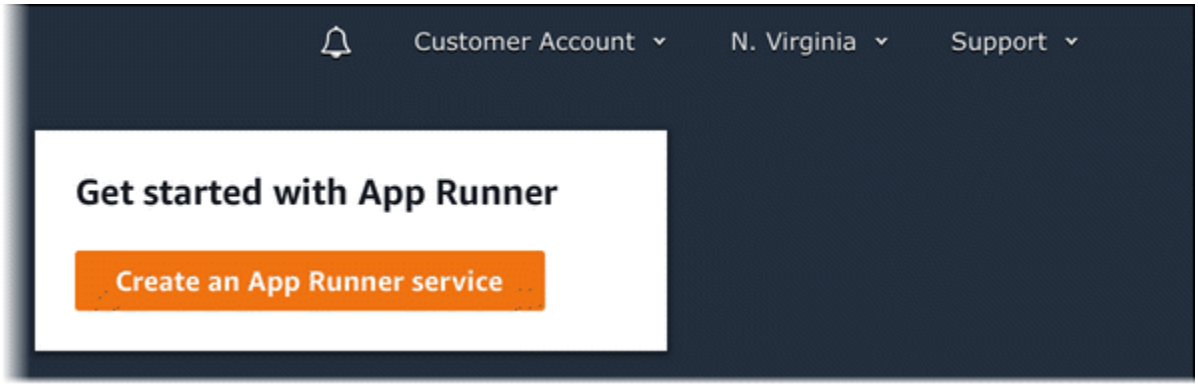


Um einen App Runner-Dienst auf der Grundlage eines Quellcode-Repositorys zu erstellen

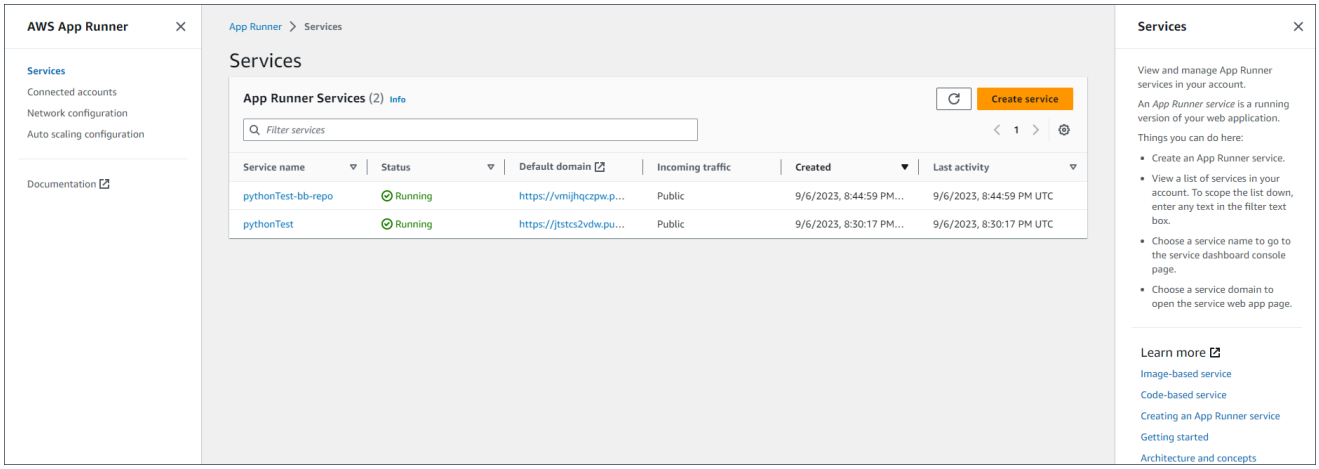
1. Konfigurieren Sie Ihren Quellcode.

   a. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.

   b. Wenn für noch AWS-Konto keine App Runner-Dienste verfügbar sind, wird die Startseite der Konsole angezeigt. Wählen Sie App Runner-Dienst erstellen aus.



Wenn es AWS-Konto bereits Dienste gibt, wird die Seite Dienste mit einer Liste Ihrer Dienste angezeigt. Wählen Sie Create service.



c. Wählen Sie auf der Seite Quelle und Bereitstellung im Abschnitt Quelle für Repository-Typ die Option Quellcode-Repository aus.

d. Wählen Sie einen Anbietertyp aus. Wähle entweder GitHuboder Bitbucket.

e. Wähle als Nächstes „Neu hinzufügen“. Wenn du dazu aufgefordert wirst, gib deine GitHub oder Bitbucket-Anmeldeinformationen ein.

f. Wähle die nächsten Schritte basierend auf dem Anbietertyp, den du zuvor ausgewählt hast.

> **Note**
>
> Die folgenden Schritte zur Installation des AWS-Connectors für GitHub Ihr GitHub Konto sind einmalige Schritte. Sie können die Verbindung wiederverwenden, um mehrere App Runner-Services auf der Grundlage der Repositorys in diesem Konto zu erstellen. Wenn Sie über eine bestehende Verbindung verfügen, wählen Sie diese aus und fahren Sie mit der Repository-Auswahl fort.

Das Gleiche gilt für den AWS-Connector für dein Bitbucket-Konto. Wenn du GitHub sowohl Bitbucket als auch Bitbucket als Quellcode-Repositorys für deine App Runner-Services verwendest, musst du für jeden Anbieter einen AWS-Connector installieren. Anschließend kannst du jeden Connector wiederverwenden, um weitere App Runner-Services zu erstellen.

- Gehen Sie dafür GitHubwie folgt vor.

  i. Geben Sie auf dem nächsten Bildschirm einen Verbindungsnamen ein.

  ii. Wenn Sie App Runner zum ersten Mal verwenden GitHub , wählen Sie Andere installieren aus.

  iii. Wählen Sie im GitHub Dialogfeld AWS Connector für Ihren GitHub Kontonamen aus, wenn Sie dazu aufgefordert werden.

  iv. Wenn Sie aufgefordert werden, den AWS Connector für zu autorisieren GitHub, wählen Sie AWS-Verbindungen autorisieren.

  v. Wählen Sie im GitHub Dialogfeld „ AWS Connector installieren für“ die Option Installieren aus.

     Ihr Kontoname wird als das ausgewählte GitHub Konto/die gewählte Organisation angezeigt. Sie können jetzt ein Repository in Ihrem Konto auswählen.

  vi. Wählen Sie für Repository das von Ihnen erstellte Beispiel-Repository aus`python-hello`. Wählen Sie für Branch den Standard-Branch-Namen Ihres Repositorys (z. B. main).

  vii. Behalten Sie für das Quellverzeichnis den Standardwert bei. Das Verzeichnis ist standardmäßig das Stammverzeichnis des Repositorys. Sie haben Ihren Quellcode in den vorherigen Schritten mit den Voraussetzungen im Stammverzeichnis des Repositorys gespeichert.

- Gehen Sie für Bitbucket wie folgt vor.

  i. Geben Sie auf dem nächsten Bildschirm einen Verbindungsnamen ein.

  ii. Wenn du Bitbucket zum ersten Mal mit App Runner verwendest, wähle Andere installieren aus.

  iii. Im Dialogfeld „AWS CodeStar Zugriffsanfragen“ kannst du deinen Workspace auswählen und Zugriff auf die AWS CodeStar Bitbucket-Integration gewähren. Wähle deinen Workspace aus und wähle dann Zugriff gewähren aus.

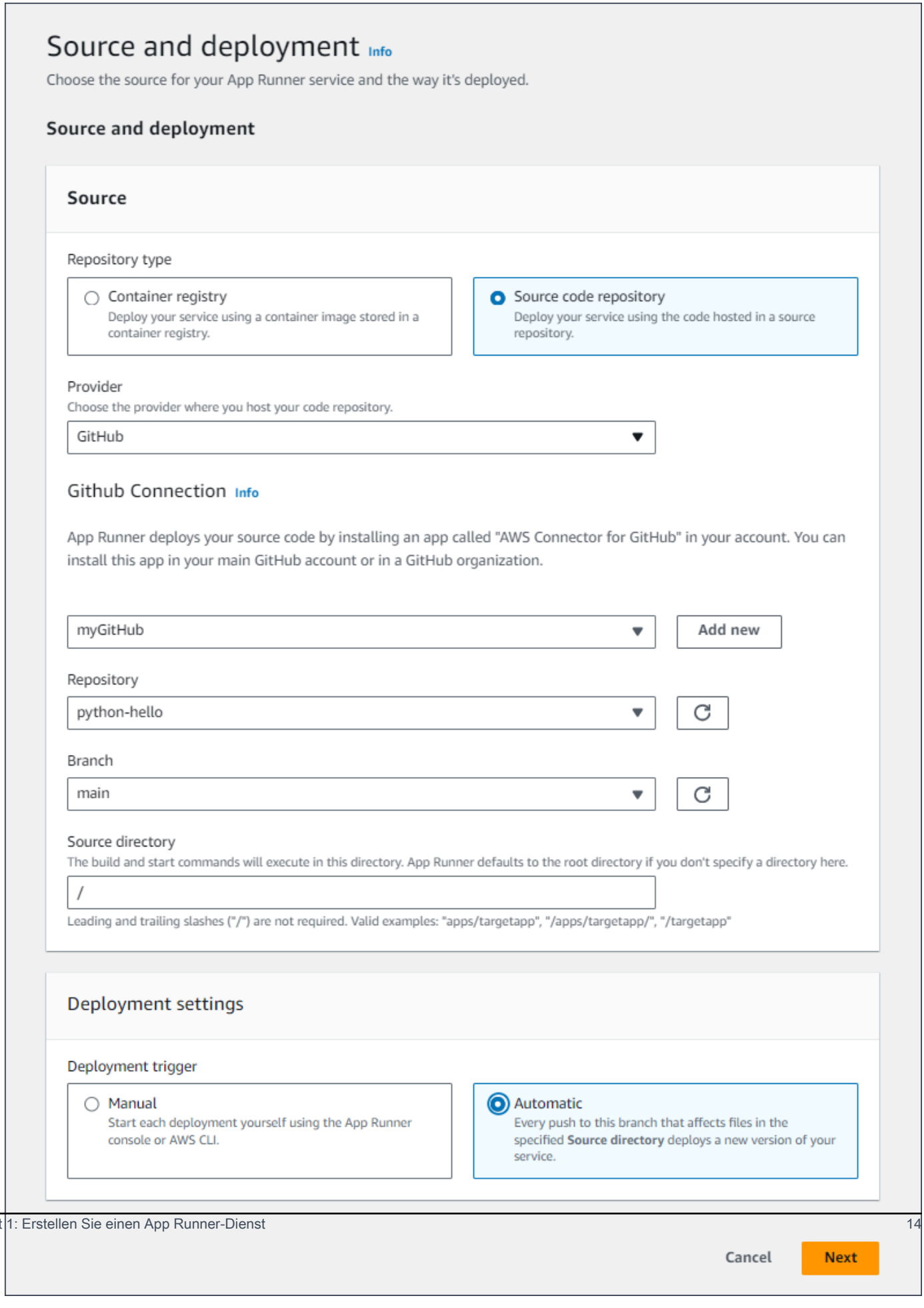
iv. Als Nächstes wirst du zur AWS Konsole weitergeleitet. Vergewissere dich, dass die Bitbucket-Anwendung auf den richtigen Bitbucket-Workspace eingestellt ist, und wähle Weiter aus.

v. Wähle unter Repository das Beispiel-Repository aus, das du erstellt hast,. `python-hello` Wählen Sie für Branch den Standard-Branch-Namen Ihres Repositorys (z. B. main).

vi. Behalten Sie für das Quellverzeichnis den Standardwert bei. Das Verzeichnis ist standardmäßig das Stammverzeichnis des Repositorys. Sie haben Ihren Quellcode in den vorherigen Schritten mit den Voraussetzungen im Stammverzeichnis des Repositorys gespeichert.

2. Konfigurieren Sie Ihre Bereitstellungen: Wählen Sie im Abschnitt Bereitstellungseinstellungen die Option Automatisch und dann Weiter aus.

> **Note**
>
> Bei der automatischen Bereitstellung wird bei jedem neuen Commit in Ihr Repository-Quellverzeichnis automatisch eine neue Version Ihres Dienstes bereitgestellt.


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. Konfigurieren Sie den Anwendungsaufbau.

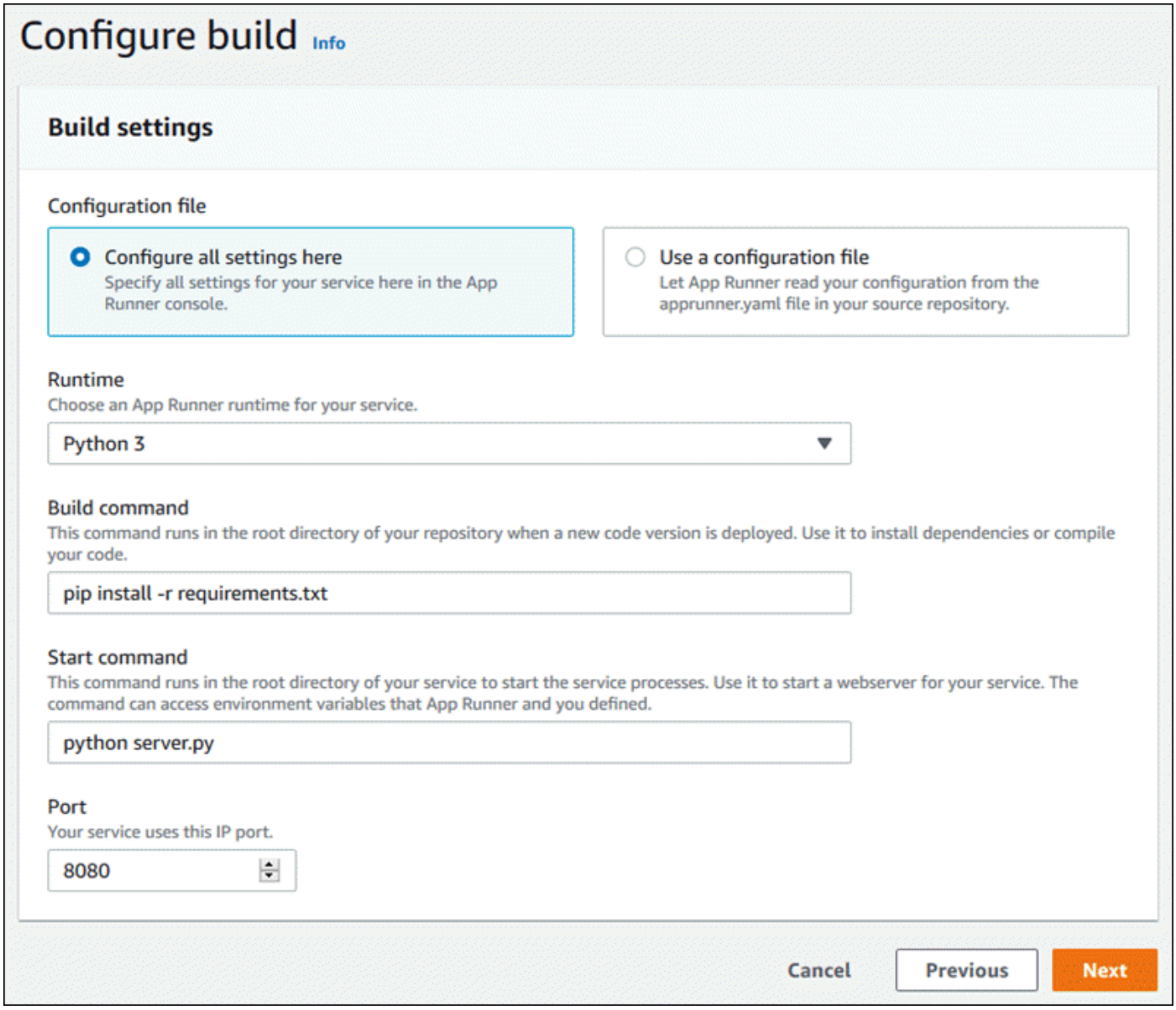
   a. Wählen Sie auf der Seite „Build konfigurieren“ unter Konfigurationsdatei die Option Alle Einstellungen hier konfigurieren aus.

   b. Geben Sie die folgenden Build-Einstellungen an:

      - Runtime — Wählen Sie Python 3.
      - Befehl erstellen — Geben Sie ein**`pip install -r requirements.txt`**.
      - Befehl starten — Geben Sie ein**`python server.py`**.
      - Port — Geben Sie ein**`8080`**.

   c. Wählen Sie Weiter.

> **Note**
>
> Die Python-3-Laufzeit erstellt ein Docker-Image mit einem Python-3-Basisimage und Ihrem Python-Beispielcode. Anschließend wird ein Dienst gestartet, der eine Container-Instance dieses Images ausführt.



4. Konfigurieren Sie Ihren Dienst.

   a. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Diensteinstellungen einen Dienstnamen ein.

   b. Wählen Sie unter Umgebungsvariablen die Option Umgebungsvariable hinzufügen aus. Geben Sie die folgenden Werte für die Umgebungsvariable ein.

      - Quelle — Wählen Sie Klartext
      - Name der Umgebungsvariablen — **`NAME`**
      - Wert der Umgebungsvariablen — beliebiger Name (z. B. Ihr Vorname).

> **Note**
>
> Die Beispielanwendung liest den Namen, den Sie in dieser Umgebungsvariablen festgelegt haben, und zeigt den Namen auf ihrer Webseite an.

c. Wählen Sie Weiter.

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.

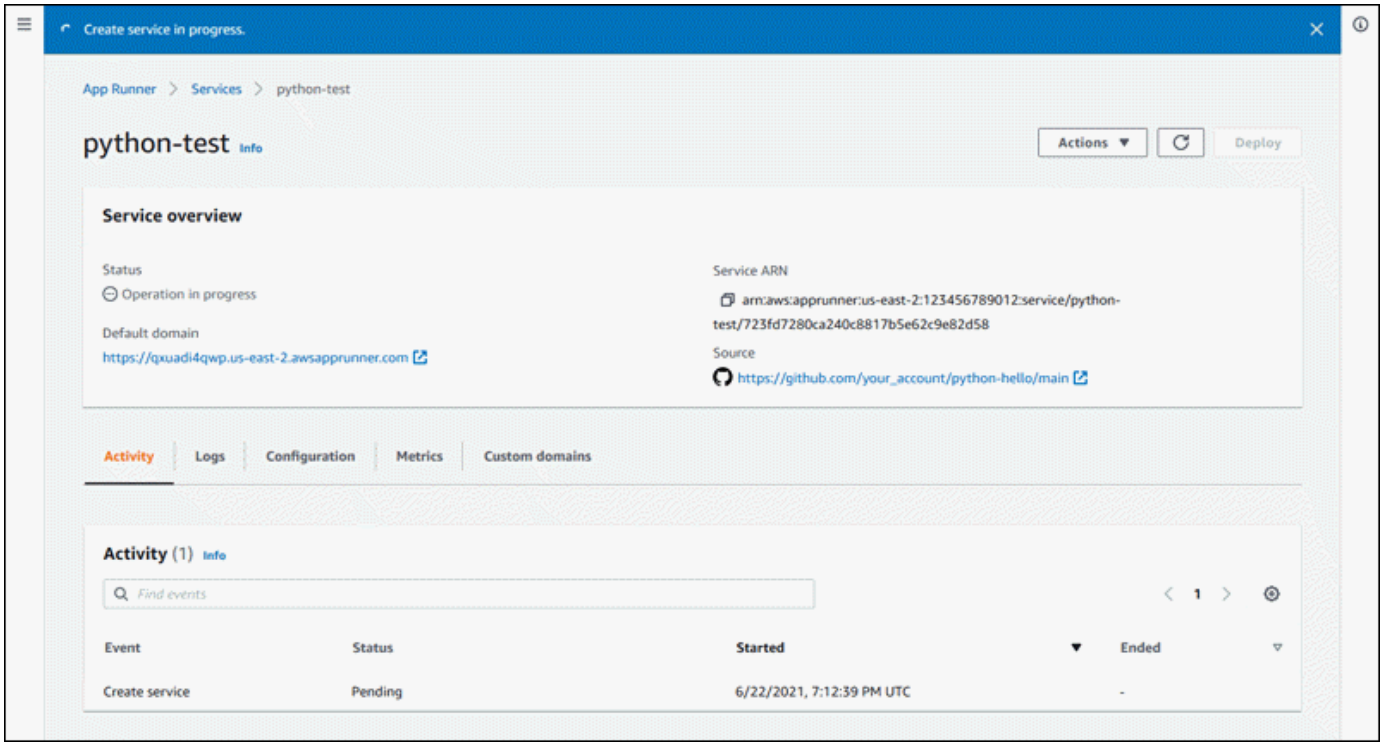
▼ Tags Info

Use tags to search and filter your resources, track your AWS costs, and control access permissions.

Tags — *optional*

5. Überprüfen Sie auf der Seite Überprüfen und erstellen alle von Ihnen eingegebenen Details und wählen Sie dann Erstellen und bereitstellen aus.

   Wenn der Service erfolgreich erstellt wurde, zeigt die Konsole das Service-Dashboard mit einer Serviceübersicht über den neuen Service an.



6. Stellen Sie sicher, dass Ihr Dienst ausgeführt wird.

   a. Warten Sie auf der Service-Dashboard-Seite, bis der Dienststatus Running lautet.

   b. Wählen Sie den Standard-Domainwert — das ist die URL zur Website Ihres Dienstes.

> **Note**
>
> Um die Sicherheit Ihrer App Runner-Anwendungen zu erhöhen, ist die Domain*.awsapprunner.com in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihre App Runner-Anwendungen einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Auf einer Webseite wird angezeigt: Hallo,***your name***!

Schritt 2: Ändern Sie Ihren Servicecode

In diesem Schritt nehmen Sie eine Änderung an Ihrem Code im Quellverzeichnis des Repositorys vor. Die CI/CD-Funktion von App Runner erstellt und implementiert die Änderung automatisch für Ihren Service.

Um eine Änderung an Ihrem Servicecode vorzunehmen

1. Navigieren Sie zu Ihrem Beispiel-Repository.
2. Bearbeiten Sie die Datei mit dem Namen`server.py`.
3. Ändern Sie in dem Ausdruck, der der Variablen zugewiesen ist`message`, `Hello` den Text in`Good morning`.
4. Speichern Sie Ihre Änderungen und übernehmen Sie sie im Repository.
5. Die folgenden Schritte veranschaulichen die Änderung des Servicecodes in einem GitHub Repository.

    a. Navigieren Sie zu Ihrem GitHub Beispiel-Repository.

    b. Wählen Sie den Dateinamen`server.py`, um zu dieser Datei zu navigieren.

    c. Wählen Sie Diese Datei bearbeiten (das Stiftsymbol).

    d. Ändern Sie in dem Ausdruck, der der Variablen zugewiesen ist`message`, `Hello` den Text in`Good morning`.

e. Wählen Sie Commit changes (Änderungen übernehmen) aus.

6. Der neue Commit beginnt mit der Bereitstellung für Ihren App Runner-Dienst. Auf der Service-Dashboard-Seite ändert sich der Dienststatus zu Vorgang in Bearbeitung.

   Warten Sie, bis die Bereitstellung beendet ist. Auf der Service-Dashboard-Seite sollte sich der Dienststatus wieder in Wird ausgeführt ändern.

7. Stellen Sie sicher, dass die Bereitstellung erfolgreich ist: Aktualisieren Sie den Browser-Tab, auf dem die Webseite Ihres Dienstes angezeigt wird.

   Auf der Seite wird jetzt die geänderte Meldung angezeigt: Guten Morgen,***your name***!

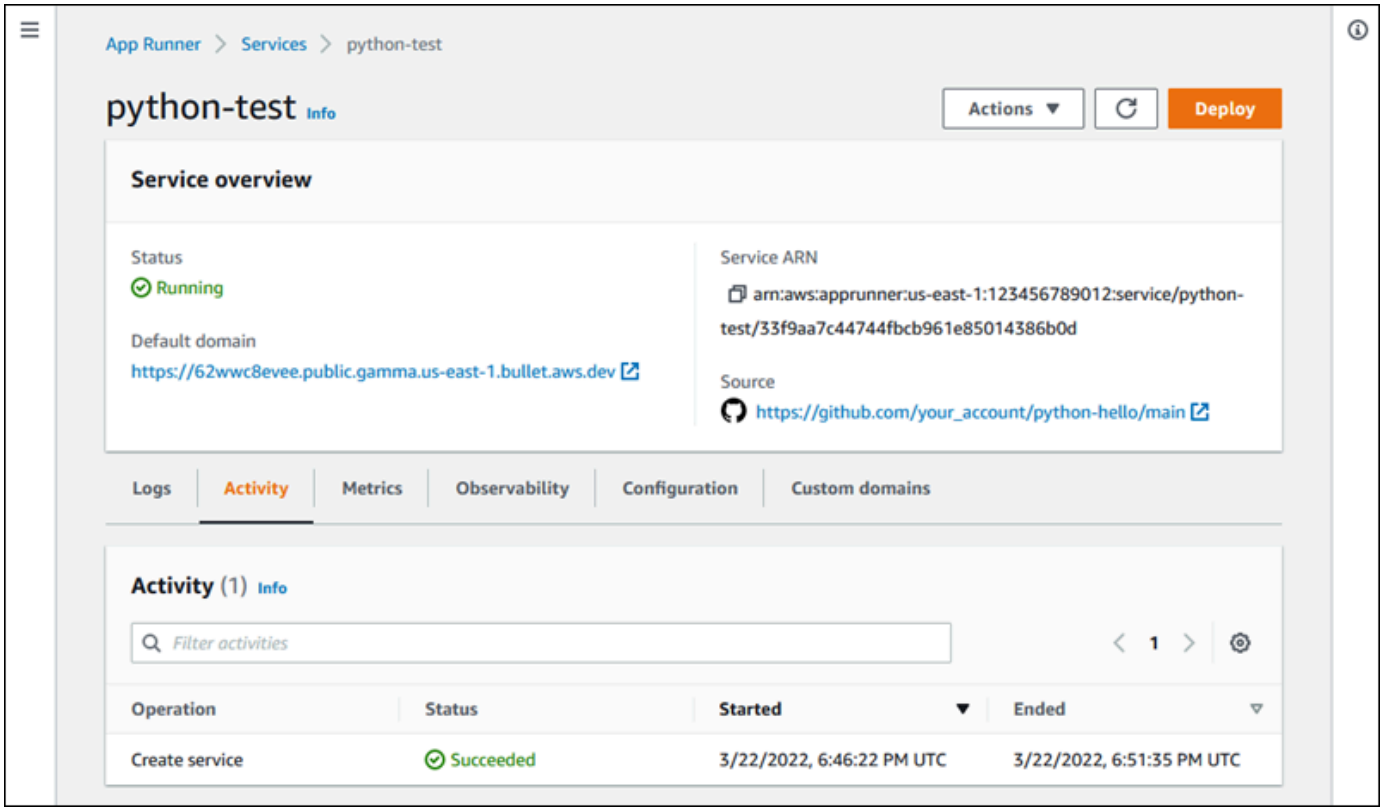
Schritt 3: Nehmen Sie eine Konfigurationsänderung vor

In diesem Schritt nehmen Sie eine Änderung am Wert der **NAME** Umgebungsvariablen vor, um eine Änderung der Dienstkonfiguration zu demonstrieren.

Um den Wert einer Umgebungsvariablen zu ändern

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

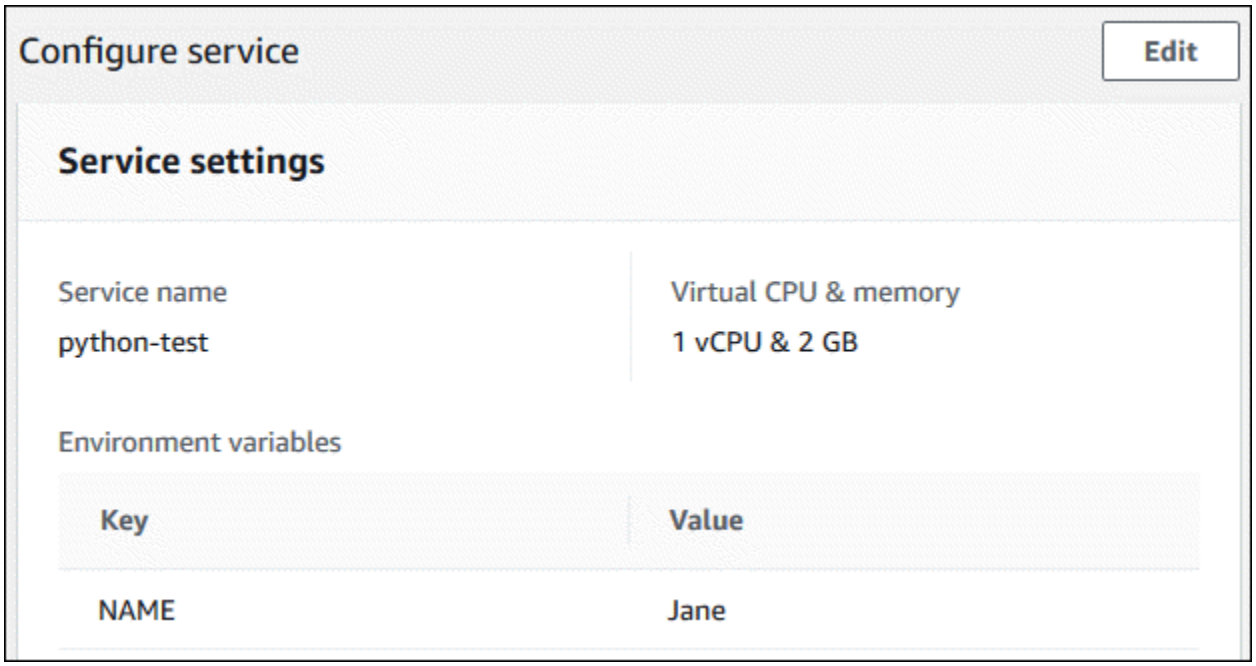
   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite die Registerkarte Konfiguration aus.

   In der Konsole werden Ihre Dienstkonfigurationseinstellungen in mehreren Abschnitten angezeigt.

4. Wählen Sie im Abschnitt Dienst konfigurieren die Option Bearbeiten aus.



5. Ändern Sie für die Umgebungsvariable mit dem Schlüssel **`NAME`** den Wert in einen anderen Namen.
6. Wählen Sie Apply changes.

   App Runner startet den Aktualisierungsvorgang. Auf der Service-Dashboard-Seite ändert sich der Dienststatus in Betrieb.
7. Warten Sie, bis das Update beendet ist. Auf der Service-Dashboard-Seite sollte sich der Dienststatus wieder in Wird ausgeführt ändern.
8. Stellen Sie sicher, dass das Update erfolgreich war: Aktualisieren Sie den Browser-Tab, auf dem die Webseite Ihres Dienstes angezeigt wird.

   Auf der Seite wird jetzt der geänderte Name angezeigt: Guten Morgen,***`new name`***!

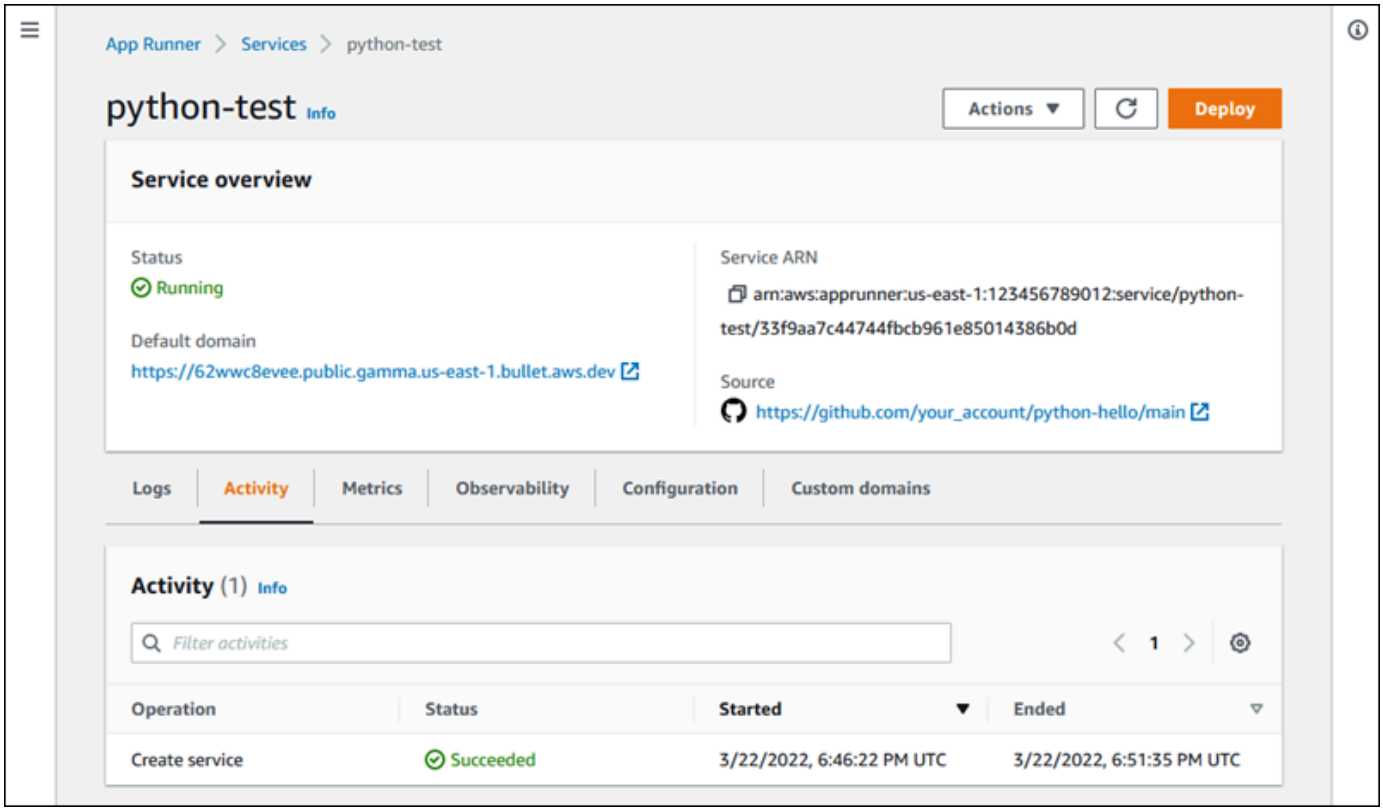
Schritt 4: Logs für Ihren Service anzeigen

In diesem Schritt verwenden Sie die App Runner-Konsole, um Protokolle für Ihren App Runner-Dienst anzuzeigen. App Runner streamt Logs an Amazon CloudWatch CloudWatch Logs (Logs) und zeigt sie auf dem Dashboard Ihres Services an. Informationen zu App Runner-Protokollen finden Sie unterthe section called “Protokolle (CloudWatch Protokolle)”.

So zeigen Sie Protokolle für Ihren Dienst an

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

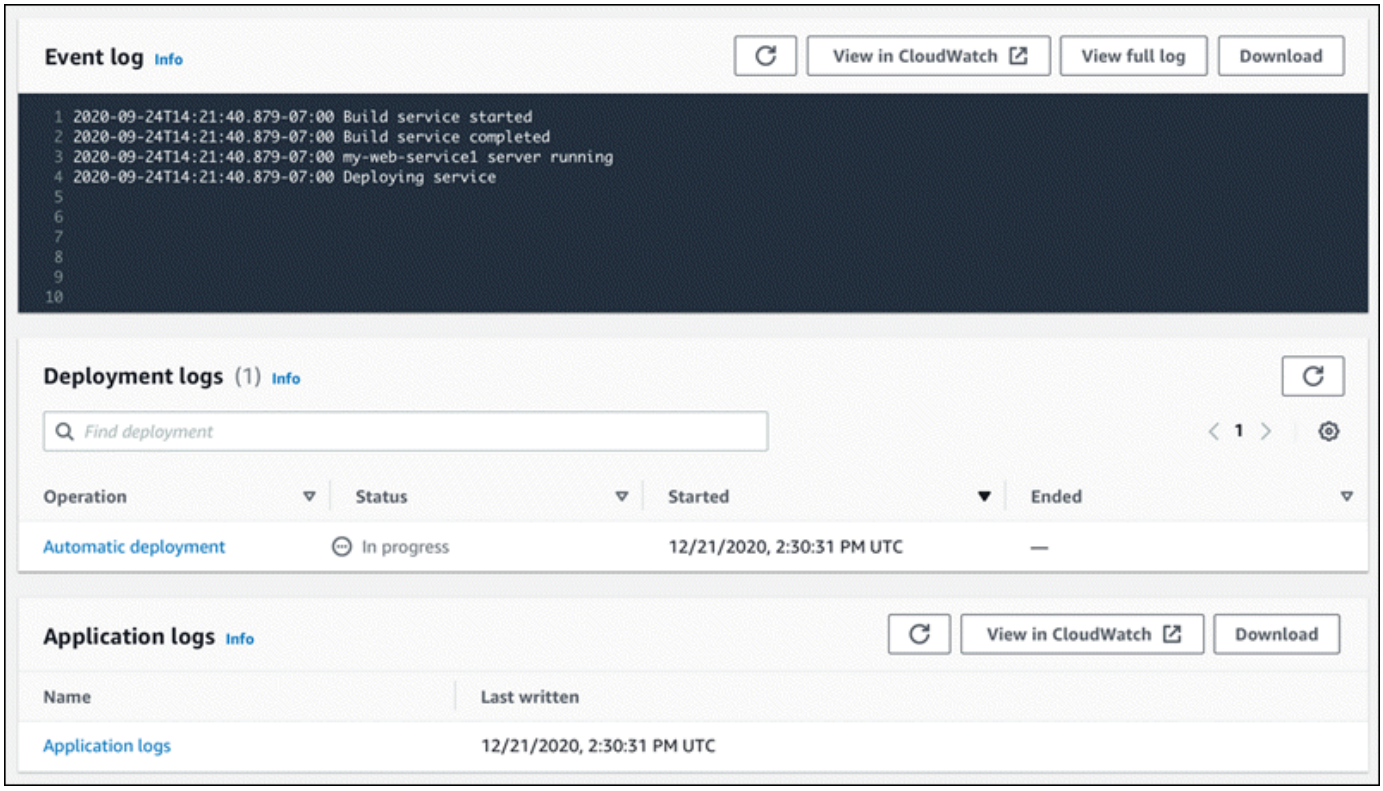
   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite den Tab Logs aus.

   Die Konsole zeigt einige Arten von Protokollen in verschiedenen Abschnitten an:

   - Ereignisprotokoll — Aktivität im Lebenszyklus Ihres App Runner-Dienstes. Die Konsole zeigt die neuesten Ereignisse an.
   - Bereitstellungsprotokolle — Quell-Repository-Bereitstellungen für Ihren App Runner-Dienst. Die Konsole zeigt für jede Bereitstellung einen separaten Protokollstream an.
   - Anwendungsprotokolle — Die Ausgabe der Webanwendung, die für Ihren App Runner-Dienst bereitgestellt wurde. Die Konsole kombiniert die Ausgabe aller laufenden Instanzen in einem einzigen Protokollstream.



4. Um nach bestimmten Bereitstellungen zu suchen, suchen Sie in der Liste der Bereitstellungsprotokolle nach unten, indem Sie einen Suchbegriff eingeben. Sie können nach jedem Wert suchen, der in der Tabelle erscheint.
5. Um den Inhalt eines Protokolls anzuzeigen, wählen Sie Vollständiges Protokoll anzeigen (Ereignisprotokoll) oder den Namen des Protokolldatenstroms (Bereitstellungs- und Anwendungsprotokolle).
6. Wählen Sie Herunterladen, um ein Protokoll herunterzuladen. Wählen Sie für einen Bereitstellungsprotokollstream zunächst einen Protokollstream aus.
7. Wählen Sie Anzeigen in CloudWatch, um die CloudWatch Konsole zu öffnen und alle Funktionen zu nutzen, um Ihre App Runner-Serviceprotokolle zu durchsuchen. Wählen Sie für einen Deployment-Log-Stream zunächst einen Log-Stream aus.

> **Note**
>
> Die CloudWatch Konsole ist besonders nützlich, wenn Sie anstelle des kombinierten Anwendungsprotokolls Anwendungsprotokolle bestimmter Instanzen anzeigen möchten.

Schritt 5: Bereinigen

Sie haben jetzt gelernt, wie Sie einen App Runner-Dienst erstellen, Protokolle anzeigen und einige Änderungen vornehmen. In diesem Schritt löschen Sie den Dienst, um Ressourcen zu entfernen, die Sie nicht mehr benötigen.

Um Ihren Dienst zu löschen

1. Wählen Sie auf der Service-Dashboard-Seite Aktionen und dann Dienst löschen aus.
2. Geben Sie im Bestätigungsdialogfeld den angeforderten Text ein und wählen Sie dann Löschen.

   Ergebnis: Die Konsole navigiert zur Seite „Dienste“. Der Dienst, den Sie gerade gelöscht haben, zeigt den Status LÖSCHEN an. Kurze Zeit später verschwindet er aus der Liste.

Erwägen Sie auch, die GitHub und Bitbucket-Verbindungen zu löschen, die Sie im Rahmen dieses Tutorials erstellt haben. Weitere Informationen finden Sie unter the section called “Verbindungen”.

Was kommt als Nächstes

Nachdem Sie Ihren ersten App Runner-Dienst bereitgestellt haben, erfahren Sie in den folgenden Themen mehr:

- Architektur und Konzepte— Die Architektur, die wichtigsten Konzepte und AWS Ressourcen im Zusammenhang mit App Runner.
- Bildbasierter Dienstund Codebasierter Dienst — Die zwei Arten von Anwendungsquellen, die App Runner bereitstellen kann.
- Entwicklung für App Runner— Dinge, die Sie wissen sollten, wenn Sie Anwendungscode für die Bereitstellung in App Runner entwickeln oder migrieren.
- App Runner-Konsole— Verwalten und überwachen Sie Ihren Service mit der App Runner-Konsole.
- Verwaltung Ihres Dienstes— Verwalten Sie den Lebenszyklus Ihres App Runner-Dienstes.
- Beobachtbarkeit— Verschaffen Sie sich einen Überblick über Ihre App Runner-Servicebetriebe, indem Sie Metriken überwachen, Protokolle lesen, Ereignisse verarbeiten, Service-Aktionsaufrufen verfolgen und Anwendungsereignisse wie HTTP-Aufrufe verfolgen.
- App Runner-Konfigurationsdatei— Eine konfigurationsbasierte Methode, um Optionen für den Build und das Laufzeitverhalten Ihres App Runner-Dienstes festzulegen.

- App Runner-API— Verwenden Sie die App Runner-Anwendungsprogrammierschnittstelle (API), um App Runner-Ressourcen zu erstellen, zu lesen, zu aktualisieren und zu löschen.
- Sicherheit— Die verschiedenen Möglichkeiten AWS und Sie sorgen für Cloud-Sicherheit, während Sie App Runner und andere Dienste nutzen.

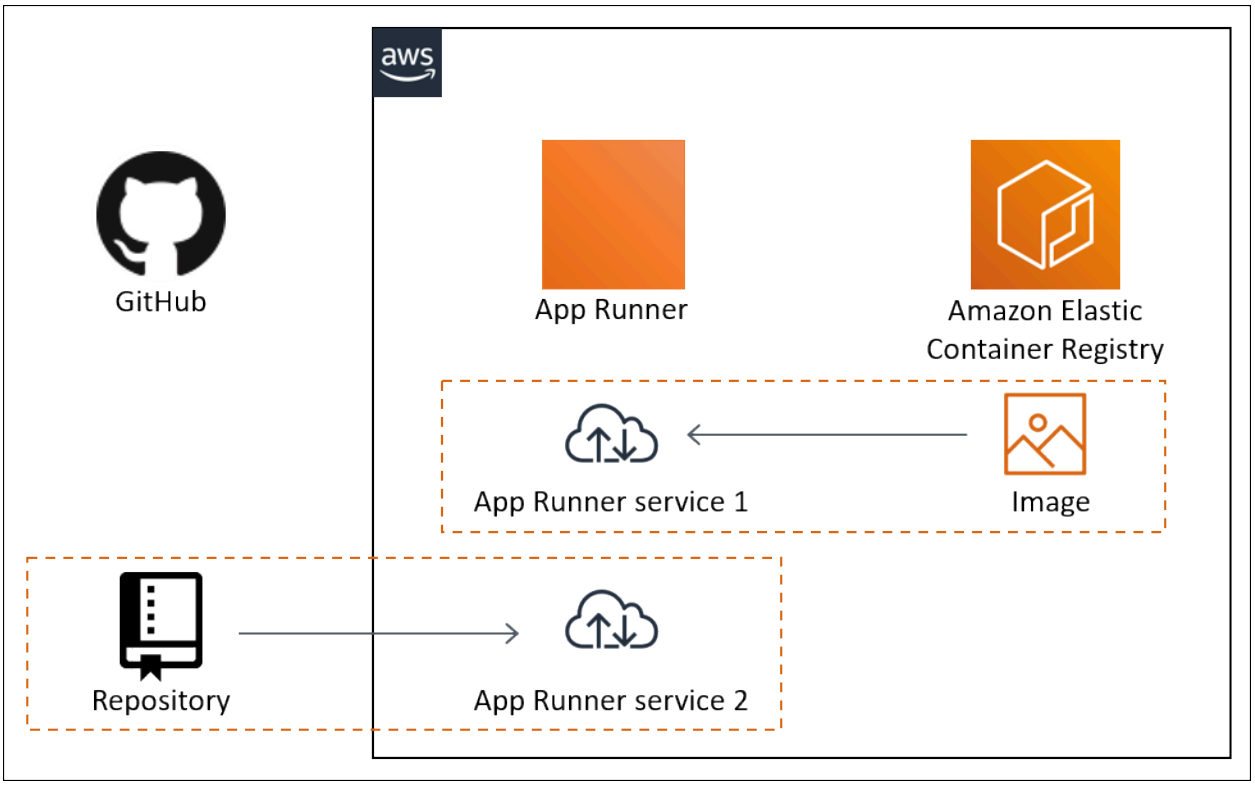
Architektur und Konzepte von App Runner

AWS App Runner entnimmt Ihren Quellcode oder Ihr Quellbild aus einem Repository und erstellt und verwaltet einen laufenden Webservice für Sie in der AWS Cloud. In der Regel müssen Sie nur eine App Runner-Aktion aufrufen CreateService, um Ihren Dienst zu erstellen.

Mit einem Quell-Image-Repository stellen Sie ein ready-to-use Container-Image bereit, das App Runner für die Ausführung Ihres Webdienstes bereitstellen kann. Mit einem Quellcode-Repository stellen Sie Ihren Code und Anweisungen zum Erstellen und Ausführen eines Webdienstes bereit und zielen auf eine bestimmte Laufzeitumgebung ab. App Runner unterstützt mehrere Programmierplattformen mit jeweils einer oder mehreren verwalteten Laufzeiten für Plattform-Hauptversionen.

Derzeit kann App Runner Ihren Quellcode entweder aus einem Bitbucket oder einem GitHubRepository abrufen, oder es kann Ihr Quell-Image aus der Amazon Elastic Container Registry (Amazon ECR) in Ihrem abrufen. AWS-Konto

Das folgende Diagramm zeigt einen Überblick über die App Runner-Servicearchitektur. In dem Diagramm gibt es zwei Beispieldienste: Einer stellt Quellcode von Amazon ECR bereit GitHub, und der andere stellt ein Quell-Image von Amazon ECR bereit. Derselbe Ablauf gilt für das Bitbucket-Repository.



App Runner-Konzepte

Im Folgenden finden Sie die wichtigsten Konzepte im Zusammenhang mit Ihrem Webservice, der in App Runner ausgeführt wird:

- App Runner-Dienst — Eine AWS Ressource, die App Runner verwendet, um Ihre Anwendung auf der Grundlage des Quellcode-Repositorys oder Container-Images bereitzustellen und zu verwalten. Ein App Runner-Dienst ist eine laufende Version Ihrer Anwendung. Weitere Informationen zum Erstellen eines Dienstes finden Sie unterthe section called “Erstellung”.
- Quelltyp — Der Typ des Quell-Repositorys, das Sie für die Bereitstellung Ihres App Runner-Dienstes bereitstellen: Quellcode oder Quellbild.
- Repository-Anbieter — Der Repository-Service, der Ihre Anwendungsquelle enthält (z. GitHubB. Bitbucket oder Amazon ECR).
- App Runner-Verbindung — Eine AWS Ressource, mit der App Runner auf ein Konto eines Repository-Anbieters zugreifen kann (z. B. ein GitHub Konto oder eine Organisation). Weitere Informationen zu Verbindungen finden Sie unter the section called “Verbindungen”.
- Runtime — Ein Basis-Image für die Bereitstellung eines Quellcode-Repositorys. App Runner bietet eine Vielzahl von verwalteten Laufzeiten für verschiedene Programmierplattformen und Versionen. Weitere Informationen finden Sie unter Codebasierter Dienst.
- Bereitstellung — Eine Aktion, die eine Version Ihres Quell-Repositorys (Code oder Bild) auf einen App Runner-Dienst anwendet. Die erste Bereitstellung für den Dienst erfolgt im Rahmen der Diensterstellung. Spätere Bereitstellungen können auf zwei Arten erfolgen:
  - Automatische Bereitstellung — Eine CI/CD-Funktion. Sie können einen App Runner-Dienst so konfigurieren, dass jede Version Ihrer Anwendung automatisch erstellt (für den Quellcode) und bereitgestellt wird, so wie sie im Repository erscheint. Dies kann ein neuer Commit in einem Quellcode-Repository oder eine neue Image-Version in einem Quell-Image-Repository sein.
  - Manuelle Bereitstellung — Eine Bereitstellung für Ihren App Runner-Dienst, die Sie explizit starten.
- Benutzerdefinierte Domain — Eine Domain, die Sie mit Ihrem App Runner-Dienst verknüpfen. Benutzer Ihrer Webanwendung können diese Domain anstelle der standardmäßigen App Runner-Subdomain verwenden, um auf Ihren Webservice zuzugreifen. Weitere Informationen finden Sie unter the section called “Benutzerdefinierte Domainnamen”.

> **Note**
>
> Um die Sicherheit Ihrer App Runner-Anwendungen zu erhöhen, ist die Domain*.awsapprunner.com in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihre App Runner-Anwendungen einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

- Wartung — Eine Aktivität, die App Runner gelegentlich in der Infrastruktur ausführt, auf der Ihr App Runner-Dienst ausgeführt wird. Während der Wartung ändert sich der Dienststatus vorübergehend für einige Minuten auf `OPERATION_IN_PROGRESS` (Vorgang wird in der Konsole ausgeführt). Aktionen an Ihrem Service (z. B. Bereitstellung, Konfigurationsupdate, Pause/Wiederaufnahme oder Löschen) werden während dieser Zeit blockiert. Führen Sie die Aktion einige Minuten später erneut aus, wenn der Dienststatus wieder auf eingestellt ist. `RUNNING`

  > **Note**
  >
  > Wenn Ihre Aktion fehlschlägt, bedeutet das nicht, dass Ihr App Runner-Dienst ausgefallen ist. Ihre Anwendung ist aktiv und bearbeitet weiterhin Anfragen. Es ist unwahrscheinlich, dass es bei Ihrem Dienst zu Ausfallzeiten kommt.

  Insbesondere migriert App Runner Ihren Dienst, wenn Probleme in der zugrunde liegenden Hardware festgestellt werden, auf der der Dienst gehostet wird. Um Serviceausfälle zu vermeiden, stellt App Runner Ihren Dienst auf einer neuen Gruppe von Instanzen bereit und leitet den Datenverkehr auf diese um (eine blaugrüne Bereitstellung). Gelegentlich kann es zu einer leichten vorübergehenden Erhöhung der Gebühren kommen.

Von App Runner unterstützte Konfigurationen

Wenn Sie einen App Runner-Dienst konfigurieren, geben Sie die virtuelle CPU- und Speicherkonfiguration an, die Ihrem Dienst zugewiesen werden soll. Sie zahlen auf der Grundlage

der von Ihnen ausgewählten Rechenkonfiguration. Weitere Informationen zu Preisen finden Sie unter AWS Resource Groups -Preise.

Die folgende Tabelle enthält Informationen zu den vCPU- und Speicherkonfigurationen, die App Runner unterstützt:

CPU	Arbeitsspeicher
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

App Runner-Ressourcen

Wenn Sie App Runner verwenden, erstellen und verwalten Sie einige Arten von Ressourcen in Ihrem AWS-Konto. Diese Ressourcen werden verwendet, um auf Ihren Code zuzugreifen und Ihre Dienste zu verwalten.

Die folgende Tabelle bietet einen Überblick über diese Ressourcen:

Ressourcenname	Beschreibung
Service	Stellt eine laufende Version Ihrer Anwendung dar. Ein Großteil des restlichen Handbuchs beschreibt Diensttypen, Verwaltung, Konfiguration und Überwachung. ARN: `arn:aws:apprunner:` *region*`:`*account-id* `:service/` *service-name* `[/`*service-id* `]`
Connection	Ermöglicht Ihren App Runner-Diensten Zugriff auf private Repositorys, die bei Drittanbietern gespeichert sind. Existiert als separate Ressource für die gemeinsame Nutzung durch mehrere Dienste. Weitere Informationen zu Verbindungen finden Sie unter the section called “Verbindungen”. ARN: `arn:aws:apprunner:` *region*`:`*account-id* `:connection/` *connection-name* `[/`*connection-id* `]`
AutoScalingConfiguration	Stellt Ihren App Runner-Diensten Einstellungen zur Verfügung, die die automatische Skalierung Ihrer Anwendung steuern. Existiert als separate Ressource für die gemeinsame Nutzung durch mehrere Dienste. Weitere Informationen zum Auto Scaling finden Sie unter the section called “Auto-Scaling”. ARN: `arn:aws:apprunner:` *region*`:`*account-id* `:autoscalingconfiguration/` *config-name* `[/`*config-revision* `[/`*config-id* `]]`
ObservabilityConfiguration	Konfiguriert zusätzliche Funktionen zur Anwendungsbeobachtung für Ihre App Runner-Dienste. Existiert als separate Ressource für die gemeinsame Nutzung durch mehrere Dienste. Weitere Hinweise zur Konfiguration der Observability finden Sie unterthe section called “Konfiguration der Beobachtbarkeit”. ARN: `arn:aws:apprunner:` *region*`:`*account-id* `:observabilityconfiguration/` *config-name* `[/`*config-revision* `[/`*config-id* `]]`

Ressourcenname	Beschreibung
VpcConnector	Konfiguriert VPC-Einstellungen für Ihre App Runner-Dienste. Existiert als separate Ressource für die gemeinsame Nutzung durch mehrere Dienste. Weitere Informationen zur VPC-Funktionalität finden Sie unterthe section called “Ausgehender Verkehr”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :vpcconnector/` *`connector-name`* `[/`*`connector-revision`*` [/`*`connector-id`*` ]]`
VpcIngressConnection	Es handelt sich um eine AWS App Runner Ressource, die zur Konfiguration des eingehenden Datenverkehrs verwendet wird. Es stellt eine Verbindung zwischen einem VPC-Schnittstellenendpunkt und dem App Runner-Service her, sodass Ihr App Runner-Service nur von einer Amazon VPC aus zugänglich ist. Weitere Informationen zur Funktionalität von VPCIngress Connection finden Sie unter. the section called “Privaten Endpunkt aktivieren” ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :vpcingressconnection/` *`vpc-ingress-connection-name`* *`[/connector-id ]]`*

App Runner-Ressourcenkontingente

AWS legt Ihrem Konto einige Kontingente (auch als Limits bezeichnet) für die jeweilige AWS AWS-Region Ressourcennutzung fest. In der folgenden Tabelle sind Kontingente für App Runner-Ressourcen aufgeführt. Kontingente sind auch unter AWS App Runner Endpoints und Kontingente in der Allgemeine AWS-Referenzaufgeführt.

Ressourcenkontingent	Beschreibung	Standardwert	Einstellbar?
Services	Die maximale Anzahl von Diensten, die Sie in Ihrem Konto für jeden Dienst erstellen können AWS-Region.	30	✓ Ja

Ressourcenkontingent		Beschreibung	Standardwert	Einstellbar?
Connections		Die maximale Anzahl von Verbindungen, die Sie in Ihrem Konto für jeden erstellen können AWS-Region. Sie können eine einzelne Verbindung in mehreren Services verwenden.	10	✓ Ja
Auto scaling configurations	Namen	Die maximale Anzahl eindeutiger Namen, die Sie in Auto Scaling-Konfigurationen haben können, die Sie jeweils in Ihrem Konto erstellen AWS-Region. Sie können eine einzelne Auto Scaling-Konfiguration in mehreren Services verwenden.	10	✓ Ja
	Revisionen pro Name	Die maximale Anzahl von Auto Scaling-Konfigurationsrevisionen, die Sie in Ihrem Konto AWS-Region für jeden eindeutigen Namen erstellen können. Sie können eine einzelne Version der Auto Scaling-Konfiguration in mehreren Diensten verwenden.	5	✗ Nein
Observability configurations	Namen	Die maximale Anzahl eindeutiger Namen, die Sie in Observability-Konfigurationen verwenden können, die Sie jeweils AWS-Region in Ihrem Konto erstellen. Sie können eine einzelne Beobachtbarkeitskonfiguration in mehreren Services verwenden.	10	✓ Ja
	Revisionen pro Name	Die maximale Anzahl von Revisionen der Observability-Konfiguration, die Sie in Ihrem Konto AWS-Region für jeden eindeutigen Namen erstellen können. Sie können eine einzelne Version der Observability-Konfiguration in mehreren Diensten verwenden.	10	✗ Nein

Ressourcenkontingent	Beschreibung	Standardwert	Einstellbar?
VPC connectors	Die maximale Anzahl von VPC-Connectoren, die Sie für jeden AWS-Region in Ihrem Konto erstellen können. Sie können einen einzelnen VPC-Konnektor in mehreren Services verwenden.	10	✓ Ja
VPC Ingress Connection	Die maximale Anzahl von VPC-Ingress-Verbindungen, die Sie in Ihrem Konto für jede Verbindung erstellen können. AWS-Region Sie können eine einzelne VPC-Ingress-Verbindung verwenden, um auf mehrere App Runner-Dienste zuzugreifen.	1	✗ Nein

Die meisten Kontingente sind anpassbar, und Sie können für sie eine Erhöhung des Kontingents beantragen. Weitere Informationen dazu finden Sie unter Beantragen einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch.

App Runner-Dienst, der auf einem Quellbild basiert

Sie können AWS App Runner damit Dienste erstellen und verwalten, die auf zwei grundlegend unterschiedlichen Typen von Dienstquellen basieren: Quellcode und Quell-Image. Unabhängig vom Quelltyp kümmert sich App Runner um das Starten, Ausführen, Skalieren und den Lastenausgleich Ihres Dienstes. Sie können die CI/CD Funktionen von App Runner nutzen, um Änderungen an Ihrem Quell-Image oder Code nachzuverfolgen. Wenn App Runner eine Änderung entdeckt, erstellt es automatisch (für den Quellcode) und stellt die neue Version für Ihren App Runner-Dienst bereit.

In diesem Kapitel werden Dienste beschrieben, die auf einem Quell-Image basieren. Informationen zu Diensten, die auf Quellcode basieren, finden Sie unterCodebasierter Dienst.

Ein Quell-Image ist ein öffentliches oder privates Container-Image, das in einem Image-Repository gespeichert ist. Sie verweisen App Runner auf ein Image und es startet einen Dienst, der einen Container ausführt, der auf diesem Image basiert. Es ist keine Erstellungsphase erforderlich. Vielmehr stellen Sie ein ready-to-deploy Bild bereit.

> **Note**
>
> Bei der Bereitstellung von Container-Images sind Sie dafür verantwortlich, diese Images regelmäßig zu aktualisieren und zu patchen. App Runner verwaltet zwar die Infrastruktur, Sie sollten jedoch die Sicherheit und den up-to-date Status der bereitgestellten Container-Images sicherstellen. Weitere Informationen finden Sie in der AWS App Runner-Dokumentation

Anbieter von Bild-Repositorien

App Runner unterstützt die folgenden Image-Repository-Anbieter:

- Amazon Elastic Container Registry (Amazon ECR) — Speichert Bilder, die privat für einen AWS-Konto sind.
- Amazon Elastic Container Registry Public (Amazon ECR Public) — Speichert Bilder, die öffentlich lesbar sind.

Anwendungsfälle für Anbieter

- Verwenden eines in Amazon ECR gespeicherten Bildes in Ihrem Konto AWS

- Verwenden eines in Amazon ECR gespeicherten Bildes in einem anderen Konto AWS
- Verwenden eines in Amazon ECR Public gespeicherten Bildes

Verwenden eines in Amazon ECR gespeicherten Bildes in Ihrem Konto AWS

Amazon ECR speichert Bilder in Repositorys. Es gibt private und öffentliche Repositorien. Um Ihr Image aus einem privaten Repository für einen App Runner-Service bereitzustellen, benötigt App Runner die Erlaubnis, Ihr Image aus Amazon ECR zu lesen. Um App Runner diese Berechtigung zu erteilen, müssen Sie App Runner eine Zugriffsrolle zuweisen. Dies ist eine AWS Identity and Access Management (IAM-) Rolle, die über die erforderlichen Amazon ECR-Aktionsberechtigungen verfügt. Wenn Sie die App Runner-Konsole verwenden, um den Service zu erstellen, können Sie eine bestehende Rolle in Ihrem Konto auswählen. Alternativ können Sie die IAM-Konsole verwenden, um eine neue benutzerdefinierte Rolle zu erstellen. Sie können auch festlegen, dass die App Runner-Konsole auf der Grundlage verwalteter Richtlinien eine Rolle für Sie erstellt.

Wenn Sie die App Runner API oder die verwenden AWS CLI, führen Sie einen zweistufigen Prozess durch. Zunächst verwenden Sie die IAM-Konsole, um eine Zugriffsrolle zu erstellen. Sie können eine verwaltete Richtlinie verwenden, die App Runner bereitstellt, oder Ihre eigenen benutzerdefinierten Berechtigungen eingeben. Anschließend geben Sie die Zugriffsrolle bei der Diensterstellung mithilfe der CreateServiceAPI-Aktion an.

Informationen zur Erstellung des App Runner-Dienstes finden Sie unterthe section called “Erstellung”.

Verwenden eines in Amazon ECR gespeicherten Bildes in einem anderen Konto AWS

Wenn Sie einen App Runner-Service erstellen, können Sie ein Bild verwenden, das in einem Amazon ECR-Repository gespeichert ist und zu einem anderen AWS Konto gehört als dem, in dem sich Ihr Service befindet. Bei der Verwendung eines kontoübergreifenden Images sind zusätzlich zu den im vorherigen Abschnitt über ein Bild für dasselbe Konto aufgeführten Überlegungen einige weitere Überlegungen zu beachten.

- Dem kontoübergreifenden Repository sollte eine Richtlinie beigefügt sein. Die Repository-Richtlinie gewährt Ihrer Zugriffsrolle Berechtigungen zum Lesen von Bildern im Repository. Verwenden Sie zu diesem Zweck die folgende Richtlinie. *access-role-arn*Ersetzen Sie es durch den Amazon-Ressourcennamen (ARN) Ihrer Zugriffsrolle.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "access-role-arn"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}
```

Informationen zum Anhängen einer Repository-Richtlinie an ein Amazon ECR-Repository finden Sie unter Setting a repository policy statement im Amazon Elastic Container Registry-Benutzerhandbuch.

- App Runner unterstützt keine automatische Bereitstellung von Amazon ECR-Images in einem anderen Konto als dem, in dem sich Ihr Service befindet.

Verwenden eines in Amazon ECR Public gespeicherten Bildes

Amazon ECR Public speichert öffentlich lesbare Bilder. Dies sind die Hauptunterschiede zwischen Amazon ECR und Amazon ECR Public, die Sie im Zusammenhang mit App Runner-Diensten beachten sollten:

- Öffentliche Amazon ECR Images sind öffentlich lesbar. Sie müssen keine Zugriffsrolle angeben, wenn Sie einen Service erstellen, der auf einem Amazon ECR Public Image basiert. An das Repository müssen keine Richtlinien angehängt werden.
- App Runner unterstützt keine automatische (kontinuierliche) Bereitstellung von Amazon ECR Public Images.

Starten Sie einen Service direkt von Amazon ECR Public

Sie können Container-Images kompatibler Webanwendungen, die in der Amazon ECR Public Gallery gehostet werden, direkt als Webservices starten, die auf App Runner ausgeführt werden. Suchen Sie beim Durchsuchen der Galerie auf der Galerie-Seite nach Launch with App Runner nach einem Bild. Ein Bild mit dieser Option ist mit App Runner kompatibel. Weitere Informationen zur Galerie finden Sie unter Verwenden der Amazon ECR Public Gallery im Amazon ECR Public-Benutzerhandbuch.

Um ein Galeriebild als App Runner-Service zu starten

1. Wählen Sie auf der Galerieseite eines Bilds die Option Mit App Runner starten aus.

   Ergebnis: Die App Runner-Konsole wird in einem neuen Browser-Tab geöffnet. In der Konsole wird der Assistent zum Erstellen von Diensten angezeigt, in dem die meisten erforderlichen neuen Dienstdetails bereits ausgefüllt sind.
2. Wenn Sie Ihren Service in einer anderen AWS Region als der, die in der Konsole angezeigt wird, erstellen möchten, wählen Sie die Region aus, die in der Kopfzeile der Konsole angezeigt wird. Wählen Sie dann eine andere Region aus.
3. Geben Sie für Port die Portnummer ein, auf der die Image-Anwendung lauscht. Sie finden ihn normalerweise auf der Galerieseite für das Bild.
4. Ändern Sie optional alle anderen Konfigurationsdetails.
5. Wählen Sie Weiter, überprüfen Sie die Einstellungen und wählen Sie dann Create & Deploy aus.

Beispiel für ein Bild

Das App Runner-Team verwaltet das hello-app-runnerBeispielbild in einer Amazon ECR Public Gallery. Sie können dieses Beispiel verwenden, um mit der Erstellung eines imagebasierten App Runner-Dienstes zu beginnen. Weitere Informationen finden Sie unter hello-app-runner.

App Runner-Dienst basiert auf Quellcode

Sie können AWS App Runner damit Dienste erstellen und verwalten, die auf zwei grundlegend unterschiedlichen Typen von Dienstquellen basieren: Quellcode und Quell-Image. Unabhängig vom Quelltyp kümmert sich App Runner um das Starten, Ausführen, Skalieren und den Lastenausgleich Ihres Dienstes. Sie können die CI/CD-Funktion von App Runner verwenden, um Änderungen an Ihrem Quell-Image oder Code nachzuverfolgen. Wenn App Runner eine Änderung entdeckt, erstellt es automatisch die neue Version (für den Quellcode) und stellt sie für Ihren App Runner-Dienst bereit.

In diesem Kapitel werden Dienste beschrieben, die auf Quellcode basieren. Informationen zu Diensten, die auf einem Quell-Image basieren, finden Sie unterBildbasierter Dienst.

Quellcode ist Anwendungscode, den App Runner für Sie erstellt und bereitstellt. Sie verweisen App Runner auf ein Quellverzeichnis in einem Code-Repository und wählen eine geeignete Laufzeit aus, die einer Programmierplattformversion entspricht. App Runner erstellt ein Image, das auf dem Basis-Image der Runtime und Ihrem Anwendungscode basiert. Anschließend wird ein Dienst gestartet, der einen auf diesem Image basierenden Container ausführt.

App Runner bietet praktische plattformspezifische verwaltete Laufzeiten. Jede dieser Laufzeiten erstellt ein Container-Image aus Ihrem Quellcode und fügt Ihrem Image Laufzeitabhängigkeiten hinzu. Sie müssen keine Container-Konfiguration und keine Build-Anweisungen wie ein Dockerfile angeben.

In den Unterthemen dieses Kapitels werden die verschiedenen Plattformen behandelt, die App Runner unterstützt — verwaltete Plattformen, die verwaltete Laufzeiten für verschiedene Programmierumgebungen und Versionen bereitstellen.

Themen

- Verwenden der -.NET-Plattform
- Verwenden der -PHP-Plattform
- Verwenden der -Ruby-Plattform
- Verwenden der Go-Plattform von

Anbieter von Quellcode-Repositorys

App Runner stellt Ihren Quellcode bereit, indem er ihn aus einem Quellcode-Repository liest. App Runner unterstützt zwei Quellcode-Repository-Anbieter: GitHubund Bitbucket.

Bereitstellung über Ihren Quellcode-Repository-Anbieter

Um Ihren Quellcode aus einem Quellcode-Repository für einen App Runner-Dienst bereitzustellen, stellt App Runner eine Verbindung zu diesem Dienst her. Wenn Sie die App Runner-Konsole verwenden, um einen Dienst zu erstellen, geben Sie Verbindungsdetails und ein Quellverzeichnis für App Runner an, um Ihren Quellcode bereitzustellen.

Verbindungen

Sie geben Verbindungsdetails im Rahmen der Diensterstellung an. Wenn Sie die App Runner-API oder die verwenden AWS CLI, ist eine Verbindung eine separate Ressource. Zunächst erstellen Sie die Verbindung mithilfe der CreateConnectionAPI-Aktion. Anschließend geben Sie den ARN der Verbindung während der Diensterstellung mithilfe der CreateServiceAPI-Aktion an.

Quellverzeichnis

Wenn Sie einen Service erstellen, geben Sie auch ein Quellverzeichnis an. Standardmäßig verwendet App Runner das Stammverzeichnis Ihres Repositorys als Quellverzeichnis. Das Quellverzeichnis ist der Speicherort in Ihrem Quellcode-Repository, in dem der Quellcode und die Konfigurationsdateien Ihrer Anwendung gespeichert werden. Die Befehle build und start werden ebenfalls vom Quellverzeichnis aus ausgeführt. Wenn Sie die App Runner-API oder die verwenden AWS CLI , um einen Service zu erstellen oder zu aktualisieren, geben Sie das Quellverzeichnis in den Aktionen CreateServiceund UpdateServiceAPI an. Weitere Informationen finden Sie im nachfolgenden Quellverzeichnis-Abschnitt.

Weitere Informationen zur Erstellung des App Runner-Dienstes finden Sie unterthe section called “Erstellung”. Weitere Informationen zu App Runner-Verbindungen finden Sie unterthe section called “Verbindungen”.

Quellverzeichnis

Wenn Sie einen App Runner-Dienst erstellen, können Sie das Quellverzeichnis zusammen mit dem Repository und dem Branch angeben. Setzen Sie den Wert des Felds Quellverzeichnis auf den Repository-Verzeichnispfad, in dem der Quellcode und die Konfigurationsdateien der Anwendung gespeichert sind. App Runner führt die Befehle Build und Start über den von Ihnen angegebenen Quellverzeichnispfad aus.

Geben Sie den Wert für den Quellverzeichnispfad als absoluten Wert aus dem Stammverzeichnis des Repositorys ein. Wenn Sie keinen Wert angeben, wird standardmäßig das Verzeichnis der obersten Ebene des Repositorys verwendet, das auch als Repository-Stammverzeichnis bezeichnet wird.

Sie haben auch die Möglichkeit, neben dem Repository-Verzeichnis der obersten Ebene auch andere Quellverzeichnispfade anzugeben. Dies unterstützt eine Monorepo-Repository-Architektur, was bedeutet, dass der Quellcode für mehrere Anwendungen in einem Repository gespeichert wird. Um mehrere App Runner-Dienste von einem einzigen Monorepo aus zu erstellen und zu unterstützen, geben Sie bei der Erstellung der einzelnen Dienste unterschiedliche Quellverzeichnisse an.

> **Note**
>
> Wenn Sie dasselbe Quellverzeichnis für mehrere App Runner-Dienste angeben, werden beide Dienste einzeln bereitgestellt und ausgeführt.

Wenn Sie sich dafür entscheiden, eine `apprunner.yaml` Konfigurationsdatei zur Definition Ihrer Serviceparameter zu verwenden, platzieren Sie diese im Quellverzeichnisordner des Repositorys.

Wenn die Option „Deployment-Trigger“ auf „Automatisch“ gesetzt ist, lösen die Änderungen, die Sie im Quellverzeichnis vornehmen, eine automatische Bereitstellung aus. Nur die Änderungen im Quellverzeichnispfad lösen eine automatische Bereitstellung aus. Es ist wichtig zu verstehen, wie sich der Speicherort des Quellverzeichnisses auf den Umfang einer automatischen Bereitstellung auswirkt. Weitere Informationen finden Sie unter Automatisierte Bereitstellungen unterBereitstellungsmethoden.

> **Note**
>
> Wenn Ihr App Runner-Dienst die verwalteten PHP-Runtimes verwendet und Sie ein anderes Quellverzeichnis als das Standard-Root-Repository angeben möchten, ist es wichtig,

die richtige PHP-Laufzeitversion zu verwenden. Weitere Informationen finden Sie unter Verwenden der -PHP-Plattform.

Von App Runner verwaltete Plattformen

Von App Runner verwaltete Plattformen bieten verwaltete Laufzeiten für verschiedene Programmierumgebungen. Jede verwaltete Laufzeit macht es einfach, Container zu erstellen und auszuführen, die auf einer Version einer Programmiersprache oder Laufzeitumgebung basieren. Wenn Sie eine verwaltete Runtime verwenden, startet App Runner mit einem verwalteten Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält ein Sprach-Runtime-Paket sowie einige Tools und beliebte Abhängigkeitspakete. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*<language-name><major-version>*.

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Verwaltete Runtime-Versionen und der App Runner-Build

App Runner bietet einen aktualisierten Build-Prozess für Anwendungen, die auf den Laufzeiten der neueren Hauptversionen ausgeführt werden. Dieser überarbeitete Erstellungsprozess ist schneller und effizienter. Außerdem wird ein endgültiges Image mit geringerem Platzbedarf erstellt, das nur Ihren Quellcode, Build-Artefakte und Laufzeiten enthält, die für die Ausführung Ihrer Anwendung erforderlich sind.

Wir bezeichnen den neueren Build-Prozess als den überarbeiteten App Runner-Build und den ursprünglichen Build-Prozess als den ursprünglichen App Runner-Build. Um grundlegende Änderungen an früheren Versionen von Runtime-Plattformen zu vermeiden, wendet App Runner den überarbeiteten Build nur auf bestimmte Runtime-Versionen an, in der Regel auf neu veröffentlichte Hauptversionen.

Wir haben der `apprunner.yaml` Konfigurationsdatei eine neue Komponente hinzugefügt, um den überarbeiteten Build für einen ganz bestimmten Anwendungsfall abwärtskompatibel zu machen und um auch mehr Flexibilität bei der Konfiguration des Builds Ihrer Anwendung zu bieten. Dies ist der optionale `pre-run`Parameter. In den folgenden Abschnitten erklären wir, wann dieser Parameter zusammen mit anderen nützlichen Informationen zu den Builds verwendet werden sollte.

Die folgende Tabelle zeigt, welche Version des App Runner-Builds für bestimmte verwaltete Runtime-Versionen gilt. Wir werden dieses Dokument weiterhin aktualisieren, um Sie über unsere aktuellen Laufzeiten auf dem Laufenden zu halten.

Plattform	Ursprünglicher Aufbau	Überarbeiteter Aufbau
Python – Informationen zur Veröffentlichung	• Python 3.8 • Python 3.7	• Python 3.11 (!)
Node.js – Informationen zur Veröffentlichung	• Node.js 16 • Node.js 14 • Node.js 12	• Node.js 22 • Node.js 18
Corretto — Informationen zur Veröffentlichung	• Corretto 11 • Corretto 8	
.NET – Informationen zur Veröffentlichung	• .NET 6	
PHP – Informationen zur Veröffentlichung	• PHP 8.1	
Ruby – Informationen zur Veröffentlichung	• Rubin 3.1	

Plattform	Ursprünglicher Aufbau	Überarbeiteter Aufbau
Go – Informationen zur Veröffentlichung	• Go 1	

> **Important**
>
> Python 3.11 — Wir haben spezifische Empfehlungen für die Build-Konfiguration von Diensten, die die verwaltete Python 3.11-Runtime verwenden. Weitere Informationen finden Sie Callouts für bestimmte Runtime-Versionen im Thema Python-Plattform.

Weitere Informationen zu App Runner-Builds und -Migration

Wenn Sie Ihre Anwendung auf eine neuere Runtime migrieren, die den überarbeiteten Build verwendet, müssen Sie möglicherweise Ihre Build-Konfiguration geringfügig ändern.

Um den Kontext für Migrationsüberlegungen zu schaffen, beschreiben wir zunächst die allgemeinen Prozesse sowohl für den ursprünglichen App Runner-Build als auch für den überarbeiteten Build. Im Anschluss finden Sie einen Abschnitt, in dem spezifische Merkmale Ihres Dienstes beschrieben werden, für die möglicherweise einige Konfigurationsupdates erforderlich sind.

Der ursprüngliche App Runner-Build

Der ursprüngliche App Runner-Anwendungsentwicklungsprozess nutzt den AWS CodeBuild Service. Die ersten Schritte basieren auf Bildern, die vom Service kuratiert wurden. CodeBuild Es folgt ein Docker-Build-Prozess, der das entsprechende verwaltete Runtime-Image von App Runner als Basis-Image verwendet.

Die allgemeinen Schritte sind die folgenden:

1. Führen Sie `pre-build` Befehle in einem CodeBuild -kuratierten Bild aus.

   Die `pre-build` Befehle sind optional. Sie können nur in der `apprunner.yaml` Konfigurationsdatei angegeben werden.
2. Führen Sie die `build` Befehle mit CodeBuild demselben Bild aus dem vorherigen Schritt aus.

Die `build` Befehle sind erforderlich. Sie können in der App Runner-Konsole, der App Runner-API oder in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

3. Führen Sie einen Docker-Build aus, um ein Image zu generieren, das auf dem von App Runner verwalteten Runtime-Image für Ihre spezifische Plattform und Laufzeitversion basiert.
4. Kopieren Sie das `/app` Verzeichnis aus dem Image, das wir in Schritt 2 generiert haben. Das Ziel ist das Image, das auf dem von App Runner verwalteten Runtime-Image basiert, das wir in Schritt 3 generiert haben.
5. Führen Sie die `build` Befehle erneut auf dem generierten, von App Runner verwalteten Runtime-Image aus. Wir führen die Build-Befehle erneut aus, um Build-Artefakte aus dem Quellcode in dem `/app` Verzeichnis zu generieren, das wir in Schritt 4 in das Verzeichnis kopiert haben. Dieses Image wird später von App Runner bereitgestellt, um Ihren Webservice in einem Container auszuführen.

   Die `build` Befehle sind erforderlich. Sie können in der App Runner-Konsole, der App Runner-API oder in der `apprunner.yaml` Konfigurationsdatei angegeben werden.
6. Führen Sie die `post-build` Befehle im CodeBuild Bild aus Schritt 2 aus.

   Die `post-build` Befehle sind optional. Sie können nur in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

Nach Abschluss des Builds stellt App Runner das generierte verwaltete Runtime-Image von App Runner aus Schritt 5 bereit, um Ihren Webservice in einem Container auszuführen.

Der überarbeitete App Runner-Build

Der überarbeitete Build-Prozess ist schneller und effizienter als der ursprüngliche Build-Prozess, der im vorherigen Abschnitt beschrieben wurde. Dadurch entfällt die Duplizierung der Build-Befehle, die im Build der vorherigen Version aufgetreten sind. Außerdem wird ein endgültiges Image mit geringerem Platzbedarf erstellt, das nur Ihren Quellcode, Build-Artefakte und Laufzeiten enthält, die für die Ausführung Ihrer Anwendung erforderlich sind.

Dieser Build-Prozess verwendet einen mehrstufigen Docker-Build. Die allgemeinen Prozessschritte sind die folgenden:

1. Erstellungsphase — Startet einen Docker-Build-Prozess, der `build` Befehle `pre-build` und Befehle auf den App Runner-Build-Images ausführt.

a. Kopieren Sie den Quellcode der Anwendung in das `/app` Verzeichnis.

> **Note**
>
> Dieses `/app` Verzeichnis wird in jeder Phase des Docker-Builds als Arbeitsverzeichnis bezeichnet.

b. `pre-build`-Befehle ausführen

Die `pre-build` Befehle sind optional. Sie können nur in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

c. Führen Sie die `build` Befehle aus.

Die `build` Befehle sind erforderlich. Sie können in der App Runner-Konsole, der App Runner-API oder in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

2. Paketierungsphase — Generiert das endgültige Container-Image für den Kunden, das ebenfalls auf dem App Runner-Run-Image basiert.

a. Kopieren Sie das `/app` Verzeichnis aus der vorherigen Build-Phase in das neue Run-Image. Dazu gehören der Quellcode Ihrer Anwendung und die Build-Artefakte aus der vorherigen Phase.

b. Führen Sie die `pre-run` Befehle aus. Wenn Sie das Runtime-Image außerhalb des `/app` Verzeichnisses mithilfe der `build` Befehle ändern müssen, fügen Sie diesem Segment der `apprunner.yaml` Konfigurationsdatei dieselben oder die erforderlichen Befehle hinzu.

Dies ist ein neuer Parameter, der eingeführt wurde, um den überarbeiteten App Runner-Build zu unterstützen.

Die `pre-run` Befehle sind optional. Sie können nur in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

> **Hinweise**
>
> - Die `pre-run` Befehle werden nur vom überarbeiteten Build unterstützt. Fügen Sie sie nicht der Konfigurationsdatei hinzu, wenn Ihr Service Laufzeitversionen verwendet, die den ursprünglichen Build verwenden.
> - Wenn Sie mit den `build` Befehlen nichts außerhalb des `/app` Verzeichnisses ändern müssen, müssen Sie keine `pre-run` Befehle angeben.

3. Post-Build-Phase — Diese Phase wird nach der Build-Phase fortgesetzt und führt `post-build` Befehle aus.
   a. Führen Sie die `post-build` Befehle innerhalb des Verzeichnisses aus`/app`.

      Die `post-build` Befehle sind optional. Sie können nur in der `apprunner.yaml` Konfigurationsdatei angegeben werden.

Nach Abschluss des Builds stellt App Runner dann das Run-Image bereit, um Ihren Webservice in einem Container auszuführen.

> **Note**
>
> Lassen Sie sich bei der Konfiguration des Build-Prozesses nicht von den `env` Einträgen im Abschnitt „Ausführen“ in die `apprunner.yaml` Irre führen. Auch wenn sich der `pre-run` Befehlsparameter, auf den in Schritt 2 (b) verwiesen wird, im Abschnitt Run befindet, verwenden Sie den `env` Parameter im Abschnitt Run nicht, um Ihren Build zu konfigurieren. Die `pre-run` Befehle verweisen nur auf die `env` Variablen, die im Abschnitt Build der Konfigurationsdatei definiert sind. Weitere Informationen finden Sie Abschnitt „Ausführen“ im Kapitel App Runner-Konfigurationsdatei.

Berücksichtigung der Serviceanforderungen für die Migration

Wenn Ihre Anwendungsumgebung eine dieser beiden Anforderungen erfüllt, müssen Sie Ihre Build-Konfiguration überarbeiten, indem Sie `pre-run` Befehle hinzufügen.

- Wenn Sie mit den `build` Befehlen etwas außerhalb des `/app` Verzeichnisses ändern müssen.
- Wenn Sie die `build` Befehle zweimal ausführen müssen, um die erforderliche Umgebung zu erstellen. Dies ist eine sehr ungewöhnliche Anforderung. Die überwiegende Mehrheit der Builds wird dies nicht tun.

Änderungen außerhalb des **`/app`** Verzeichnisses

- Der überarbeitete App Runner-Build geht davon aus, dass Ihre Anwendung keine Abhängigkeiten außerhalb des `/app` Verzeichnisses hat.
- Die Befehle, die Sie entweder mit der `apprunner.yaml` Datei, der App Runner-API oder der App Runner-Konsole bereitstellen, müssen Build-Artefakte im `/app` Verzeichnis generieren.

- Sie können die `post-build` Befehle`pre-build`, und ändern`build`, um sicherzustellen, dass sich alle Build-Artefakte im `/app` Verzeichnis befinden.
- Wenn Ihre Anwendung erfordert, dass der Build das generierte Image für Ihren Service außerhalb des `/app` Verzeichnisses weiter modifiziert, können Sie die neuen `pre-run` Befehle in der verwendenapprunner.yaml. Weitere Informationen finden Sie unter App Runner-Dienstoptionen mithilfe einer Konfigurationsdatei einrichten.

Die **`build`** Befehle zweimal ausführen

- Der ursprüngliche App Runner-Build führt die `build` Befehle zweimal aus, zuerst in Schritt 2, dann erneut in Schritt 5. Der überarbeitete App Runner-Build behebt diese Redundanz und führt die `build` Befehle nur einmal aus. Falls für Ihre Anwendung eine ungewöhnliche Anforderung besteht, dass die `build` Befehle zweimal ausgeführt werden müssen, bietet der überarbeitete App Runner-Build die Möglichkeit, dieselben Befehle mithilfe des `pre-run` Parameters anzugeben und erneut auszuführen. Dabei wird das gleiche Double-Build-Verhalten beibehalten.

Verwenden der Python-Plattform von

Die AWS App Runner Python-Plattform bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer Python-Version basieren. Wenn Sie eine Python-Laufzeit verwenden, startet App Runner mit einem verwalteten Python-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von Python sowie einige Tools und beliebte Abhängigkeitspakete. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*`<language-name><major-version>`*.

Gültige Namen und Versionen der Python-Laufzeit finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für Python-Laufzeiten: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Beispiel: `3.8.5`

Die folgenden Beispiele demonstrieren das Sperren von Versionen:

- `3.8`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `3.8.5`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

- Python-Laufzeitkonfiguration
- Callouts für bestimmte Runtime-Versionen
- Beispiele für Python-Runtime
- Informationen zur Python-Runtime-Version

Python-Laufzeitkonfiguration

Wenn Sie sich für eine verwaltete Laufzeit entscheiden, müssen Sie mindestens auch Befehle zum Erstellen und Ausführen konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.
- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Callouts für bestimmte Runtime-Versionen

> **Note**
>
> App Runner führt jetzt einen aktualisierten Build-Prozess für Anwendungen aus, die auf den folgenden Runtime-Versionen basieren: Python 3.11, Node.js 22 und Node.js 18. Wenn Ihre Anwendung auf einer dieser Runtime-Versionen ausgeführt wird, finden Sie weitere Informationen Verwaltete Runtime-Versionen und der App Runner-Build zum überarbeiteten Build-Prozess unter. Anwendungen, die alle anderen Runtime-Versionen verwenden, sind nicht betroffen und sie verwenden weiterhin den ursprünglichen Build-Prozess.

Python 3.11 (überarbeiteter App Runner-Build)

Verwenden Sie die folgenden Einstellungen in der Datei apprunner.yaml für die verwaltete Python 3.11-Laufzeit.

- Stellen Sie den Schlüssel im oberen Bereich auf `runtime` `python311`

  Example

  ```
  runtime: python311
  ```

- Verwenden Sie `pip3` statt von`pip`, um Abhängigkeiten zu installieren.
- Verwenden Sie den `python3` Interpreter anstelle von. `python`
- Führen Sie das `pip3` Installationsprogramm als `pre-run` Befehl aus. Python installiert Abhängigkeiten außerhalb des `/app` Verzeichnisses. Da App Runner den überarbeiteten App Runner-Build für Python 3.11 ausführt, geht alles verloren, was außerhalb des `/app` Verzeichnisses über Befehle im Build-Abschnitt der `apprunner.yaml` Datei installiert wurde. Weitere Informationen finden Sie unter Der überarbeitete App Runner-Build.

  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 -
```

Weitere Informationen finden Sie auch im Beispiel einer erweiterten Konfigurationsdatei für Python 3.11 weiter unten in diesem Thema.

Beispiele für Python-Runtime

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines Python-Dienstes. Das letzte Beispiel ist der Quellcode für eine vollständige Python-Anwendung, die Sie in einem Python-Laufzeitdienst bereitstellen können.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist `3.7.7` und`3.11`.
> Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Python-Laufzeitversion finden Sie unterthe section called “Informationen zur Veröffentlichung”.

Minimale Python-Konfigurationsdatei

Dieses Beispiel zeigt eine minimale Konfigurationsdatei, die Sie mit einer von Python verwalteten Laufzeit verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

Python 3.11 verwendet die `python3` Befehle `pip3` und. Weitere Informationen finden Sie im Beispiel einer erweiterten Konfigurationsdatei für Python 3.11 weiter unten in diesem Thema.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
```

```
      - pipenv install
run:
  command: python app.py
```

Erweiterte Python-Konfigurationsdatei

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer von Python verwalteten Laufzeit.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist`3.7.7`. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Python-Laufzeitversion finden Sie unterthe section called “Informationen zur Veröffentlichung”. Python 3.11 verwendet die `python3` Befehle `pip3` und. Weitere Informationen finden Sie im Beispiel einer erweiterten Konfigurationsdatei für Python 3.11 weiter unten in diesem Thema.

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"
```

Erweiterte Python-Konfigurationsdatei — Python 3.11 (verwendet einen überarbeiteten Build)

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer verwalteten Python 3.11-Laufzeit in der`apprunner.yaml`. Dieses Beispiel enthält einen `pre-run` Abschnitt, da diese Version von Python den überarbeiteten App Runner-Build verwendet.

Der `pre-run` Parameter wird nur vom überarbeiteten App Runner-Build unterstützt. Fügen Sie diesen Parameter nicht in Ihre Konfigurationsdatei ein, wenn Ihre Anwendung Runtime-Versionen verwendet, die vom ursprünglichen App Runner-Build unterstützt werden. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist`3.11`. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Python-Laufzeitversion finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige Python-Anwendungsquelle

Dieses Beispiel zeigt den Quellcode für eine vollständige Python-Anwendung, die Sie in einem Python-Runtime-Service bereitstellen können.

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
```

Informationen zur Python-Runtime-Version

In diesem Thema werden die vollständigen Informationen zu den Python-Laufzeitversionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — überarbeiteter App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Python 3.11 (Python311)	3.11.13	SQLite 3,50,1
	3.11,12	SQLite 3,50,0

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	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

Hinweise

- Python 3.11 — Wir haben spezifische Empfehlungen für die Build-Konfiguration von Diensten, die die verwaltete Python 3.11-Runtime verwenden. Weitere Informationen finden Sie Callouts für bestimmte Runtime-Versionen im Thema Python-Plattform.
- App Runner bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Python3 (Python3)	3.8.20	SQLite 3,50,1
	3.8,20	SQLite 3,50,0
	3.8,16	SQLite 3,46,1
	3.8,15	SQLite 3,40,0
	3.7.16	SQLite 3,50,0

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	3.7,15	SQLite 3,40,0
	3.7.10	SQLite 3,40,0

> **Note**
>
> App Runner bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Verwenden der Node.js-Plattform von

Die Plattform AWS App Runner Node.js bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer Version von Node.js basieren. Wenn Sie eine Node.js Runtime verwenden, startet App Runner mit einem verwalteten Node.js Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von Node.js und einige Tools. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*<language-name><major-version>*.

Gültige Laufzeitnamen und Versionen von Node.js finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten

Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für die Laufzeiten von Node.js: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Beispiel: `22.14.0`

Die folgenden Beispiele veranschaulichen das Sperren von Versionen:

- `22.14`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `22.14.0`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

- Laufzeitkonfiguration von Node.js
- Callouts für bestimmte Runtime-Versionen
- Beispiele für die Laufzeit von Node.js
- Informationen zur Runtime-Version von Node.js

Laufzeitkonfiguration von Node.js

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.
- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Speziell bei den Laufzeiten von Node.js können Sie den Build und die Laufzeit auch mithilfe einer JSON-Datei konfigurieren, die `package.json` im Stammverzeichnis Ihres Quell-Repositorys benannt ist. Mithilfe dieser Datei können Sie die Engine-Version von Node.js, die Abhängigkeitspakete und verschiedene Befehle (Befehlszeilenanwendungen) konfigurieren. Paketmanager wie npm oder yarn interpretieren diese Datei als Eingabe für ihre Befehle.

Zum Beispiel:

- npm installinstalliert Pakete, die durch den `devDependencies` Knoten `dependencies` und in `package.json` definiert sind.
- npm startoder npm run start führt den durch den `scripts/start` Knoten in definierten Befehl aus`package.json`.

Im Folgenden sehen Sie ein Beispiel für eine `package.json`-Datei.

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"
  }
```

```
}
```

Weitere Informationen *`package.json`* dazu finden Sie unter Erstellen einer package.json-Datei auf der npm Docs-Website.

Tipps

- Wenn Ihre `package.json` Datei einen start Befehl definiert, können Sie ihn als run Befehl in Ihrer App Runner-Konfigurationsdatei verwenden, wie das folgende Beispiel zeigt.

  Example

  package.json

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

  apprunner.yaml

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

- Wenn Sie npm install in Ihrer Entwicklungsumgebung ausführen, erstellt npm die Datei. `package-lock.json` Diese Datei enthält einen Snapshot der Paketversionen, die npm gerade installiert hat. Wenn npm anschließend Abhängigkeiten installiert, verwendet es genau diese Versionen. Wenn Sie Garn installieren, wird eine `yarn.lock` Datei erstellt. Übergeben Sie diese Dateien in Ihr Quellcode-Repository, um sicherzustellen, dass Ihre Anwendung mit den Versionen der Abhängigkeiten installiert wird, mit denen Sie sie entwickelt und getestet haben.
- Sie können auch eine App Runner-Konfigurationsdatei verwenden, um die Version von Node.js und den Startbefehl zu konfigurieren. Wenn Sie dies tun, überschreiben diese Definitionen die Definitionen in`package.json`. Ein Konflikt zwischen der `node` Version in `package.json` und dem `runtime-version` Wert in der App Runner-Konfigurationsdatei führt dazu, dass die App Runner-Buildphase fehlschlägt.

Callouts für bestimmte Runtime-Versionen

Node.js 22 und Node.js 18 (überarbeiteter App Runner-Build)

App Runner führt jetzt einen aktualisierten Build-Prozess für Anwendungen aus, die auf den folgenden Laufzeitversionen basieren: Python 3.11, Node.js 22 und Node.js 18. Wenn Ihre Anwendung auf einer dieser Runtime-Versionen ausgeführt wird, finden Sie weitere Informationen Verwaltete Runtime-Versionen und der App Runner-Build zum überarbeiteten Build-Prozess unter. Anwendungen, die alle anderen Runtime-Versionen verwenden, sind nicht betroffen und sie verwenden weiterhin den ursprünglichen Build-Prozess.

Beispiele für die Laufzeit von Node.js

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines Node.js -Dienstes.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist`22.14.0`. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Laufzeitversion von Node.js finden Sie unterthe section called “Informationen zur Veröffentlichung”.

Minimale Konfigurationsdatei Node.js

Dieses Beispiel zeigt eine Minimalkonfigurationsdatei, die Sie mit einer verwalteten Laufzeit von Node.js verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    build:
      - npm install --production
run:
```

```
  command: node app.js
```

Erweiterte Konfigurationsdatei Node.js

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer verwalteten Laufzeit von Node.js.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist*22.14.0*. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Laufzeitversion von Node.js finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Erweiterte Konfigurationsdatei Node.js — Node.js 22 (verwendet einen überarbeiteten Build)

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer von Node.js verwalteten Laufzeit in der`apprunner.yaml`. Dieses Beispiel enthält einen `pre-run` Abschnitt, da diese Version von Node.js den überarbeiteten App Runner-Build verwendet.

Der `pre-run` Parameter wird nur vom überarbeiteten App Runner-Build unterstützt. Fügen Sie diesen Parameter nicht in Ihre Konfigurationsdatei ein, wenn Ihre Anwendung Runtime-Versionen verwendet, die vom ursprünglichen App Runner-Build unterstützt werden. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist*22.14.0*. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Laufzeitversion von Node.js finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

App Node.js mit Grunt

Dieses Beispiel zeigt, wie eine Node.js -Anwendung konfiguriert wird, die mit Grunt entwickelt wurde. Grunt ist ein JavaScript Task-Runner über die Befehlszeile. Es führt sich wiederholende Aufgaben aus und verwaltet die Prozessautomatisierung, um menschliche Fehler zu reduzieren. Die Plugins Grunt und Grunt werden mit npm installiert und verwaltet. Sie konfigurieren Grunt, indem Sie die `Gruntfile.js` Datei in das Stammverzeichnis Ihres Quell-Repositorys aufnehmen.

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**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist. *22.14.0* Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Laufzeitversion von Node.js finden Sie unterthe section called “Informationen zur Veröffentlichung”.

```
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
```

Informationen zur Runtime-Version von Node.js

> **Note**
>
> Die Standardrichtlinie von App Runner sieht vor, dass eine Laufzeit als veraltet eingestuft wird, wenn für eine wichtige Komponente der Laufzeit der Community Long-Term Support (LTS) ausläuft und keine Sicherheitsupdates mehr verfügbar sind. In einigen Fällen kann App Runner das Verfallen einer Laufzeit für einen begrenzten Zeitraum hinauszögern, und zwar über das end-of-support Datum der von der Laufzeit unterstützten Sprachversion hinaus. Ein Beispiel für einen solchen Fall könnte die Erweiterung des Supports für eine Runtime sein, um Kunden Zeit für die Migration zu geben.

In diesem Thema werden die vollständigen Informationen zu den Runtime-Versionen von Node.js aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — überarbeiteter App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Node.js 22 (nodejs22)	2217,0	npm 10.9.2, Garn 1.22.22
	22.16.0	npm 10.9.2, Garn 1.22.22
	22.14.0	npm 10.9.2, Garn 1.22.22

> **Note**
>
> App Runner bietet einen überarbeiteten Build-Prozess für bestimmte Haupt-Runtimes, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Unterstützte Runtime-Versionen — überarbeiteter App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Node.js 18 (nodejs18)	18.20.8	npm 10.8.2, Garn 1.22.22
	18.20.7	npm 10.8.2, Garn 1.22.22
	18.20.6	npm 10.8.2, Garn 1.22.22
	18.20.5	npm 10.8.2, Garn 1.22.22
	18.20.4	npm 10.7.0, Garn 1.22.22
	18.20.3	npm 10.7.0, Garn 1.22.22
	18.20.2	npm 10, Garn *
	18.19.1	npm 10, Garn *
	18.19.0	npm 10, Garn *

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Node.js 16 (nodejs16)	16.20,2	npm 8.19.4, Garn 1.22.22
	16.20.1	npm 8.19.4, Garn *
	16.20.0	npm 8.19.4, Garn *
	16.19.1	npm 8.19.4, Garn *
	16.19.0	npm 8.19.4, Garn *
	16.18.1	npm 8.19.4, Garn *
	16.17.1	npm 8.19.4, Garn *
	16.17.0	npm 8.19.4, Garn *

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Node.js 14 (nodejs14)	14,21,3	npm 6.14.18, Garn 1.22.22
	14.21.2	npm 6.14.18, Garn *
	14.21.1	npm 6.14.18, Garn *
	14.20.1	npm 6.14.18, Garn *
	14.19.0	npm 6.14.18, Garn *
Node.js 12 (nodejs12)	12.22.12	npm 6.14.16, Garn 1.22.22
	12.21.0	npm 6.14.16, Garn *

Verwendung der Java-Plattform

Die AWS App Runner Java-Plattform bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer Java-Version basieren. Wenn Sie eine Java-Runtime verwenden, startet App Runner mit einem verwalteten Java-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von Java und einige Tools. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend stellt es dieses Image bereit, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*`<language-name><major-version>`*.

Derzeit basieren alle unterstützten Java-Laufzeiten auf Amazon Corretto. Gültige Namen und Versionen der Java-Laufzeit finden Sie unter. the section called “Informationen zur Veröffentlichung”

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene

beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für Amazon Corretto-Laufzeiten:

Laufzeit	Syntax	Beispiel
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`

Die folgenden Beispiele veranschaulichen das Sperren von Versionen:

- `11.0.13`— Sperrt die Open JDK-Update-Version. App Runner aktualisiert nur Open JDK- und Amazon Corretto-Builds auf niedrigerer Ebene.
- `11.0.13.08.1`— Auf eine bestimmte Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

Konfiguration der Java-Laufzeit

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.

- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Beispiele für Java-Runtime

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines Java-Dienstes. Das letzte Beispiel ist der Quellcode für eine vollständige Java-Anwendung, die Sie auf einem Corretto 11-Laufzeitdienst bereitstellen können.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist. *`11.0.13.08.1`*
> Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Java-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

Minimale Corretto 11-Konfigurationsdatei

Dieses Beispiel zeigt eine minimale Konfigurationsdatei, die Sie mit einer von Corretto 11 verwalteten Runtime verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unter.

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 .
```

Erweiterte Corretto 11-Konfigurationsdatei

Dieses Beispiel zeigt, wie Sie alle Konfigurationsschlüssel mit einer von Corretto 11 verwalteten Laufzeit verwenden können.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist. *11.0.13.08.1*
> Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Java-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige Corretto 11-Anwendungsquelle

Dieses Beispiel zeigt den Quellcode für eine vollständige Java-Anwendung, die Sie auf einem Corretto 11-Laufzeitdienst bereitstellen können.

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 apprunner.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>
```

Informationen zur Java-Runtime-Version

In diesem Thema werden die vollständigen Informationen zu den Java-Runtime-Versionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Corretto 11 (Korrekt-11)	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
	11.0.25.9.1	Maven 3.9.9, Gradle 6.9.4

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	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 (Korretto 8)	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
	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

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	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 bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Verwenden der -.NET-Plattform

AWS App Runner Die.NET-Plattform bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer .NET-Version basieren. Wenn Sie eine.NET-Runtime verwenden, startet App Runner mit einem verwalteten.NET-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von.NET sowie einige Tools und beliebte Abhängigkeitspakete. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-

Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet`<language-name><major-version>`.

Gültige Namen und Versionen von .NET-Runtime finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für .NET-Laufzeiten: `major[.minor[.patch]]`

Zum Beispiel: `6.0.9`

Die folgenden Beispiele veranschaulichen das Sperren von Versionen:

- `6.0`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `6.0.9`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

- .NET-Laufzeitkonfiguration
- .NET-Runtime-Beispiele
- Informationen zur Veröffentlichung von.NET-Runtime

.NET-Laufzeitkonfiguration

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.

- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

.NET-Runtime-Beispiele

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines .NET-Dienstes. Das letzte Beispiel ist der Quellcode für eine vollständige.NET-Anwendung, die Sie in einem.NET-Runtime-Service bereitstellen können.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist`6.0.9`. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte.NET-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

Minimale .NET-Konfigurationsdatei

Dieses Beispiel zeigt eine Minimalkonfigurationsdatei, die Sie mit einer verwalteten.NET-Laufzeit verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
```

```
run:
  command: dotnet out/HelloWorldDotNetApp.dll
```

Erweiterte.NET-Konfigurationsdatei

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer verwalteten.NET-Laufzeit.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist*6.0.9*. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte.NET-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige .NET-Anwendungsquelle

Dieses Beispiel zeigt den Quellcode für eine vollständige.NET-Anwendung, die Sie in einem.NET-Runtime-Service bereitstellen können.

> **Note**
>
> - Führen Sie den folgenden Befehl aus, um eine einfache.NET 6-Web-App zu erstellen: `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - Fügen Sie der `apprunner.yaml` erstellten.NET 6-Web-App die hinzu.

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"
```

Informationen zur Veröffentlichung von.NET-Runtime

In diesem Thema werden die vollständigen Informationen zu den.NET-Laufzeitversionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
.NET 6 (dotnet6)	6.0.36	.NET SDK 6.0.428
	6.0.33	.NET SDK 6.0.425

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	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
	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 bietet einen überarbeiteten Build-Prozess für bestimmte Haupt-Runtimes, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Verwenden der -PHP-Plattform

Die AWS App Runner PHP-Plattform bietet verwaltete Laufzeiten. Sie können jede Laufzeit verwenden, um Container mit Webanwendungen zu erstellen und auszuführen, die auf einer PHP-Version basieren. Wenn Sie eine PHP-Runtime verwenden, startet App Runner mit einem verwalteten PHP-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von PHP und einige Tools. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*`<language-name><major-version>`*.

Gültige PHP-Laufzeitnamen und -versionen finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Service bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für PHP-Laufzeiten: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Zum Beispiel: `8.1.10`

Im Folgenden finden Sie Beispiele für das Sperren von Versionen:

- `8.1`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `8.1.10`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

> **Important**
>
> Wenn Sie das Quellverzeichnis des Code-Repositorys für Ihren App Runner-Dienst an einem anderen Ort als dem Standard-Repository-Stammverzeichnis angeben möchten, muss Ihre verwaltete PHP-Runtime-Version PHP `8.1.22` oder höher sein. Ältere PHP-Runtime-Versionen verwenden `8.1.22` möglicherweise nur das Standard-Root-Quellverzeichnis.

Themen

PHP-Laufzeitkonfiguration

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.
- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Kompatibilität

Sie können Ihre App Runner-Dienste auf der PHP-Plattform mit einem der folgenden Webserver ausführen:

- Apache HTTP Server
- NGINX

Apache HTTP Server and NGINX sind mit PHP-FPM kompatibel. Sie können das starten Apache HTTP Server und NGINXindem Sie eine der folgenden Optionen verwenden:

- Supervisord — Weitere Informationen zum Ausführen eines supervisord, siehe Supervisord ausführen.
- Startskript

Beispiele zur Konfiguration Ihres App Runner-Dienstes mit der PHP-Plattform mithilfe von Apache HTTP Server oder NGINX finden Sie unter. the section called “Vollständige PHP-Anwendungsquelle”

Dateistruktur

Der `index.php` muss im `public` Ordner unter dem `root` Verzeichnis des Webservers installiert sein.

> **Note**
>
> Wir empfehlen, die `supervisord.conf` Dateien `startup.sh` oder im Stammverzeichnis des Webservers zu speichern. Stellen Sie sicher, dass der `start` Befehl auf den Speicherort verweist, an dem die `startup.sh` `supervisord.conf` Oder-Dateien gespeichert sind.

Im Folgenden finden Sie ein Beispiel für die Dateistruktur, wenn Sie supervisord.

```
/
```

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

Das Folgende ist ein Beispiel für die Dateistruktur, wenn Sie ein Startskript verwenden.

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

Wir empfehlen, diese Dateistrukturen im Quellverzeichnis des Code-Repositorys zu speichern, das für den App Runner-Dienst vorgesehen ist.

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

> **Important**
>
> Wenn Sie das Code-Repository-Quellverzeichnis für Ihren App Runner-Dienst an einem anderen Ort als dem Standard-Repository-Stammverzeichnis angeben möchten, muss Ihre von PHP verwaltete Runtime-Version PHP `8.1.22` oder höher sein. Ältere PHP-Runtime-Versionen verwenden `8.1.22` möglicherweise nur das Standard-Root-Quellverzeichnis. App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Ihr Dienst verwendet standardmäßig die neuesten Laufzeiten, es sei denn, Sie haben die Versionssperre mit dem `runtime-version` Schlüsselwort in der App Runner-Konfigurationsdatei angegeben.

Beispiele für PHP-Runtime

Im Folgenden finden Sie Beispiele für App Runner-Konfigurationsdateien, die zum Erstellen und Ausführen eines PHP-Dienstes verwendet werden.

Minimale PHP-Konfigurationsdatei

Das folgende Beispiel ist eine Minimalkonfigurationsdatei, die Sie mit einer von PHP verwalteten Laufzeit verwenden können. Weitere Hinweise zu einer Minimalkonfigurationsdatei finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

Example apprunner.yaml

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

Erweiterte PHP-Konfigurationsdatei

Das folgende Beispiel verwendet alle Konfigurationsschlüssel mit einer von PHP verwalteten Laufzeit.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist*8.1.10*. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte PHP-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige PHP-Anwendungsquelle

Die folgenden Beispiele zeigen den Quellcode von PHP-Anwendungen, den Sie für die Bereitstellung in einem PHP-Laufzeitdienst verwenden können Apache HTTP Server oder NGINX. Bei diesen Beispielen wird davon ausgegangen, dass Sie die Standarddateistruktur verwenden.

Ausführen der PHP-Plattform mit Apache HTTP Server verwenden supervisord

Example Dateistruktur

> **Note**
>
> - Die `supervisord.conf` Datei kann an einer beliebigen Stelle im Repository gespeichert werden. Stellen Sie sicher, dass der `start` Befehl auf den Speicherort der `supervisord.conf` Datei verweist.
> - Das `index.php` muss in dem `public` Ordner unter dem `root` Verzeichnis installiert sein.

```
/
## 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>
```

Ausführen der PHP-Plattform mit Apache HTTP Server verwenden startup script

Example Dateistruktur

> **Note**
>
> - Die `startup.sh` Datei kann an einer beliebigen Stelle im Repository gespeichert werden. Stellen Sie sicher, dass der `start` Befehl auf den Speicherort der `startup.sh` Datei verweist.
> - Das `index.php` muss in dem `public` Ordner unter dem `root` Verzeichnis installiert sein.

```
/
## 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**
>
> - Achte darauf, die `startup.sh` Datei als ausführbare Datei zu speichern, bevor du sie in ein Git-Repository übergibst. Verwenden Sie `chmod +x startup.sh` diese Option, um die Ausführungsberechtigung für Ihre `startup.sh` Datei festzulegen.
> - Wenn Sie die `startup.sh` Datei nicht als ausführbare Datei speichern, geben Sie den Befehl `chmod +x startup.sh` als `build` Befehl in Ihre `apprunner.yaml` Datei ein.

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>
```

Ausführen der PHP-Plattform mit NGINX verwenden supervisord

Example Dateistruktur

> **Note**
>
> - Die `supervisord.conf` Datei kann an einer beliebigen Stelle im Repository gespeichert werden. Stellen Sie sicher, dass der `start` Befehl auf den Speicherort der `supervisord.conf` Datei verweist.
> - Das `index.php` muss in dem `public` Ordner unter dem `root` Verzeichnis installiert sein.

```
/
## 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>
```

Ausführen der PHP-Plattform mit NGINX verwenden startup script

Example Dateistruktur

> **Note**
>
> - Die `startup.sh` Datei kann an einer beliebigen Stelle im Repository gespeichert werden. Stellen Sie sicher, dass der `start` Befehl auf den Speicherort der `startup.sh` Datei verweist.
> - Das `index.php` muss in dem `public` Ordner unter dem `root` Verzeichnis installiert sein.

```
/
```

```
## 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**
>
> - Achte darauf, die `startup.sh` Datei als ausführbare Datei zu speichern, bevor du sie in ein Git-Repository übergibst. Verwenden Sie `chmod +x startup.sh` diese Option, um die Ausführungsberechtigung für Ihre `startup.sh` Datei festzulegen.
> - Wenn Sie die `startup.sh` Datei nicht als ausführbare Datei speichern, geben Sie den Befehl `chmod +x startup.sh` als `build` Befehl in Ihre `apprunner.yaml` Datei ein.

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>
```

Informationen zur PHP-Runtime-Version

In diesem Thema werden die vollständigen Informationen zu den PHP-Laufzeitversionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
PHP 8.1 (php81)	8.1.32	
	8.1.31	
	8.1.29	
	8.1.28	
	8.1.27	
	8.1.26	
	8.1.24	
	8.1.22	
	8.1.21	

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
	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 bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Verwenden der -Ruby-Plattform

Die AWS App Runner Ruby-Plattform bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer Ruby-Version basieren. Wenn Sie eine Ruby-Runtime verwenden, startet App Runner mit einem verwalteten Ruby-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von Ruby und einige Tools. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet*<language-name><major-version>*.

Gültige Namen und Versionen der Ruby-Runtime finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Dienst bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für Ruby-Laufzeiten: *major*`[.`*minor*`[.`*patch*`]]`

Zum Beispiel: `3.1.2`

Die folgenden Beispiele demonstrieren das Sperren von Versionen:

- `3.1`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `3.1.2`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

- Ruby-Laufzeitkonfiguration
- Beispiele für Ruby-Runtime
- Informationen zur Ruby-Runtime-Version

Ruby-Laufzeitkonfiguration

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.
- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Beispiele für Ruby-Runtime

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines Ruby-Dienstes.

Minimale Ruby-Konfigurationsdatei

Dieses Beispiel zeigt eine minimale Konfigurationsdatei, die Sie mit einer von Ruby verwalteten Runtime verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

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
```

Erweiterte Ruby-Konfigurationsdatei

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer von Ruby verwalteten Laufzeit.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist`3.1.2`. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Ruby-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige Ruby-Anwendungsquelle

Diese Beispiele zeigen den Quellcode für eine vollständige Ruby-Anwendung, die Sie in einem Ruby-Runtime-Service bereitstellen können.

Example server.rb

```
# server.rb
require 'sinatra'
```

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

Example config.ru

```
# config.ru

require './server'

run Sinatra::Application
```

Example 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
```

Informationen zur Ruby-Runtime-Version

In diesem Thema werden die vollständigen Informationen zu den Ruby-Runtime-Versionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Ruby 3.1 (Ruby31)	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
	3.1.3	SQLite 3,41,0
	3.1.2	SQLite 3,39,4

> **Note**
>
> App Runner bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Verwenden der Go-Plattform von

Die AWS App Runner Go-Plattform bietet verwaltete Laufzeiten. Jede Laufzeit macht es einfach, Container mit Webanwendungen zu erstellen und auszuführen, die auf einer Go-Version basieren. Wenn Sie eine Go-Runtime verwenden, beginnt App Runner mit einem verwalteten Go-Runtime-Image. Dieses Image basiert auf dem Amazon Linux Docker-Image und enthält das Runtime-Paket für eine Version von Go und einige Tools. App Runner verwendet dieses verwaltete Runtime-Image als Basis-Image und fügt Ihren Anwendungscode hinzu, um ein Docker-Image zu erstellen. Anschließend wird dieses Image bereitgestellt, um Ihren Webservice in einem Container auszuführen.

Sie geben eine Laufzeit für Ihren App Runner-Dienst an, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder des CreateServiceAPI-Vorgangs erstellen. Sie können auch eine Laufzeit als Teil Ihres Quellcodes angeben. Verwenden Sie das `runtime` Schlüsselwort in einer App Runner-

Konfigurationsdatei, die Sie in Ihr Code-Repository aufnehmen. Die Benennungskonvention einer verwalteten Laufzeit lautet`<language-name><major-version>`.

Gültige Namen und Versionen von Go-Runtime finden Sie unterthe section called “Informationen zur Veröffentlichung”.

App Runner aktualisiert die Laufzeit für Ihren Service bei jeder Bereitstellung oder jedem Service-Update auf die neueste Version. Wenn Ihre Anwendung eine bestimmte Version einer verwalteten Laufzeit benötigt, können Sie diese mithilfe des `runtime-version` Schlüsselworts in der App Runner-Konfigurationsdatei angeben. Sie können sich auf eine beliebige Versionsebene beschränken, einschließlich einer Haupt- oder Nebenversion. App Runner aktualisiert die Laufzeit Ihres Dienstes nur auf niedrigerer Ebene.

Versionssyntax für Go-Laufzeiten: `major[.minor[.patch]]`

Zum Beispiel: `1.18.7`

Die folgenden Beispiele demonstrieren das Sperren von Versionen:

- `1.18`— Sperren Sie die Haupt- und Nebenversionen. App Runner aktualisiert nur Patch-Versionen.
- `1.18.7`— Auf eine bestimmte Patch-Version festlegen. App Runner aktualisiert Ihre Runtime-Version nicht.

Themen

- Gehen Sie zur Laufzeitkonfiguration
- Gehen Sie zu Runtime-Beispielen
- Informationen zur Go Runtime-Version

Gehen Sie zur Laufzeitkonfiguration

Wenn Sie sich für eine verwaltete Runtime entscheiden, müssen Sie mindestens auch Build- und Run-Befehle konfigurieren. Sie konfigurieren sie bei der Erstellung oder Aktualisierung Ihres App Runner-Dienstes. Sie können dies mit einer der folgenden Methoden tun:

- Verwenden der App Runner-Konsole — Geben Sie die Befehle im Abschnitt Build konfigurieren des Erstellungsprozesses oder der Registerkarte Konfiguration an.

- Verwenden der App Runner-API — Rufen Sie den UpdateServiceAPI-Vorgang CreateServiceoder auf. Geben Sie die Befehle mithilfe der `StartCommand` Elemente `BuildCommand` und des CodeConfigurationValuesDatentyps an.
- Mithilfe einer Konfigurationsdatei — Geben Sie einen oder mehrere Build-Befehle in bis zu drei Build-Phasen sowie einen einzelnen Run-Befehl an, der zum Starten Ihrer Anwendung dient. Es gibt zusätzliche optionale Konfigurationseinstellungen.

Die Bereitstellung einer Konfigurationsdatei ist optional. Wenn Sie einen App Runner-Dienst mithilfe der Konsole oder der API erstellen, geben Sie an, ob App Runner Ihre Konfigurationseinstellungen direkt bei der Erstellung oder aus einer Konfigurationsdatei bezieht.

Gehen Sie zu Runtime-Beispielen

Die folgenden Beispiele zeigen App Runner-Konfigurationsdateien zum Erstellen und Ausführen eines Go-Dienstes.

Minimale Go-Konfigurationsdatei

Dieses Beispiel zeigt eine minimale Konfigurationsdatei, die Sie mit einer von Go verwalteten Runtime verwenden können. Informationen zu den Annahmen, die App Runner mit einer minimalen Konfigurationsdatei trifft, finden Sie unterthe section called “Beispiele für Konfigurationsdateien”.

Example apprunner.yaml

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

Erweiterte Go-Konfigurationsdatei

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel mit einer von Go verwalteten Laufzeit.

> **Note**
>
> Die Runtime-Version, die in diesen Beispielen verwendet wird, ist*1.18.7*. Sie können sie durch eine Version ersetzen, die Sie verwenden möchten. Die neueste unterstützte Go-Runtime-Version finden Sie unterthe section called “Informationen zur Veröffentlichung”.

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"
```

Vollständige Go-Anwendungsquelle

Diese Beispiele zeigen den Quellcode für eine vollständige Go-Anwendung, die Sie in einem Go-Runtime-Service bereitstellen können.

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
```

Informationen zur Go Runtime-Version

In diesem Thema werden die vollständigen Informationen zu den Go-Runtime-Versionen aufgeführt, die App Runner unterstützt.

Unterstützte Runtime-Versionen — originaler App Runner-Build

Laufzeitname	Kleinere Versionen	Enthaltene Pakete
Go 1 (go1)	1.18.10	
	1,18,9	
	1.18,8	
	1.18,7	

> **Note**
>
> App Runner bietet einen überarbeiteten Build-Prozess für bestimmte Hauptlaufzeiten, die in jüngerer Zeit veröffentlicht wurden. Aus diesem Grund finden Sie in bestimmten Abschnitten dieses Dokuments Verweise auf den überarbeiteten App Runner-Build und den ursprünglichen App Runner-Build. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Anwendungscode für App Runner entwickeln

In diesem Kapitel werden Laufzeitinformationen und Entwicklungsrichtlinien behandelt, die Sie bei der Entwicklung oder Migration von Anwendungscode für die Bereitstellung berücksichtigen sollten. AWS App Runner

Informationen zur Laufzeit

Egal, ob Sie ein Container-Image bereitstellen oder App Runner eines für Sie erstellt, App Runner führt Ihren Anwendungscode in einer Container-Instance aus. Im Folgenden sind einige wichtige Aspekte der Laufzeitumgebung für Container-Instances aufgeführt.

- Framework-Unterstützung — App Runner unterstützt jedes Image, das eine Webanwendung implementiert. Es ist unabhängig von der Programmiersprache, die Sie wählen, und unabhängig vom Webanwendungsserver oder Framework, das Sie verwenden, falls Sie eines verwenden. Der Einfachheit halber bieten wir plattformspezifische verwaltete Laufzeiten für verschiedene Programmierplattformen an, um den Prozess der Anwendungsentwicklung und die Erstellung abstrakter Images zu optimieren.
- Webanfragen — App Runner bietet Unterstützung für HTTP 1.0 und HTTP 1.1 für die Container-Instances. Weitere Informationen zur Konfiguration Ihres Dienstes finden Sie unterthe section called “Konfiguration”. Sie müssen die Verarbeitung von sicherem HTTPS-Verkehr nicht implementieren. App Runner leitet alle eingehenden HTTP-Anfragen an die entsprechenden HTTPS-Endpunkte weiter. Sie müssen keine Einstellungen konfigurieren, um die Umleitung der HTTP-Webanfragen zu aktivieren. App Runner beendet das TLS, bevor Anfragen an Ihre Anwendungscontainer-Instance weitergeleitet werden.

> **Note**
>
> - Für die HTTP-Anfragen gilt ein Zeitlimit von insgesamt 120 Sekunden. Die 120 Sekunden beinhalten die Zeit, die die Anwendung benötigt, um die Anfrage einschließlich des Hauptteils zu lesen und das Schreiben der HTTP-Antwort abzuschließen.
> - Das Limit für das Lesen und Beantworten von Anfragen hängt von den Anwendungen ab, die Sie verwenden. Diese Anwendungen können ihre eigenen internen Timeouts haben, z. B. hat der HTTP-Server für Python, Gunicorn, ein Standard-Timeout-Limit von 30 Sekunden. In solchen Fällen überschreibt das Timeout-Limit der Anwendung das App Runner-Timeout-Limit von 120 Sekunden.

- Sie müssen keine TLS-Verschlüsselungssammlungen oder andere Parameter konfigurieren, da App Runner ein vollständig verwalteter Dienst ist und die TLS-Terminierung für Sie verwaltet.

- Zustandslose Apps — Derzeit unterstützt App Runner keine statusbehafteten Apps. Daher garantiert App Runner nicht, dass der Status über die Dauer der Verarbeitung einer einzelnen eingehenden Webanfrage hinaus bestehen bleibt.
- Speicher — App Runner skaliert die Instanzen für Ihre App Runner-Anwendung automatisch entsprechend dem eingehenden Datenverkehrsvolumen nach oben oder unten. Sie können Auto-Scaling-Optionen für Ihre App Runner-Anwendung konfigurieren. Da die Anzahl der derzeit aktiven Instanzen, die die Webanfragen verarbeiten, vom eingehenden Datenverkehrsvolumen abhängt, kann App Runner nicht garantieren, dass die Dateien über die Verarbeitung einer einzelnen Anfrage hinaus bestehen bleiben können. Daher implementiert App Runner das Dateisystem in Ihrer Container-Instance als kurzlebigen Speicher, was bedeutet, dass die Dateien vorübergehend sind. Die Dateien bleiben beispielsweise nicht erhalten, wenn Sie Ihren App Runner-Dienst anhalten und wieder aufnehmen.

  App Runner stellt Ihnen 3 GB kurzlebigen Speicher zur Verfügung und verwendet einen Teil der 3 GB kurzlebigen Speicher für das gepullte, komprimierte und unkomprimierte Container-Image auf der Instanz. Der verbleibende flüchtige Speicher kann von Ihrem App Runner-Dienst verwendet werden. Dies ist jedoch kein dauerhafter Speicher, da er staatenlos ist.

  > **Note**
  >
  > Es kann Szenarien geben, in denen die Speicherdateien über mehrere Anfragen hinweg bestehen bleiben. Wenn beispielsweise die nächste Anfrage auf derselben Instanz landet, bleiben die Speicherdateien bestehen. Die Persistenz von Speicherdateien über Anfragen hinweg kann in bestimmten Situationen nützlich sein. Wenn Sie beispielsweise eine Anfrage bearbeiten, können Sie Dateien zwischenspeichern, die Ihre Anwendung herunterlädt, falls future Anfragen sie möglicherweise benötigen. Dies könnte die future Bearbeitung von Anfragen beschleunigen, kann aber die Geschwindigkeitssteigerung nicht garantieren. Ihr Code sollte nicht davon ausgehen, dass eine Datei, die in einer früheren Anfrage heruntergeladen wurde, noch existiert.
  > Verwenden Sie einen Service wie Amazon, um garantiertes Caching mit einem In-Memory-Datenspeicher mit hohem Durchsatz und niedriger Latenz zu gewährleisten. ElastiCache

- Umgebungsvariablen — Standardmäßig stellt App Runner die `PORT` Umgebungsvariable in Ihrer Container-Instance zur Verfügung. Sie können den Variablenwert mit Portinformationen konfigurieren und benutzerdefinierte Umgebungsvariablen und Werte hinzufügen. Sie können auch sensible Daten, die in AWS Secrets Manageroder AWS Systems Manager Parameter Store gespeichert sind, als Umgebungsvariablen referenzieren. Weitere Hinweise zum Erstellen von Umgebungsvariablen finden Sie unterReferenz-Umgebungsvariablen.
- Instanzrolle — Wenn Ihr Anwendungscode AWS Dienste aufruft, die den Dienst APIs oder einen der Dienste verwenden AWS SDKs, erstellen Sie mithilfe von AWS Identity and Access Management (IAM) eine Instanzrolle. Hängen Sie sie dann bei der Erstellung an Ihren App Runner-Dienst an. Nehmen Sie alle AWS Serviceaktionsberechtigungen, die Ihr Code benötigt, in Ihre Instanzrolle auf. Weitere Informationen finden Sie unter the section called “Instanzrolle”.

Richtlinien für die Codeentwicklung

Beachten Sie diese Richtlinien, wenn Sie Code für eine App Runner-Webanwendung entwickeln.

- Container-Images patchen — Wenn Sie Container-Images bereitstellen, sind Sie dafür verantwortlich, diese Images regelmäßig zu aktualisieren und zu patchen. App Runner verwaltet zwar die Infrastruktur, Sie sollten jedoch die Sicherheit und den up-to-date Status der bereitgestellten Container-Images sicherstellen. Weitere Informationen finden Sie in der AWS App Runner-Dokumentation
- Zustandslosen Code entwerfen — Gestalten Sie die Webanwendung, die Sie für Ihren App Runner-Service bereitstellen, so, dass sie zustandslos ist. Ihr Code sollte davon ausgehen, dass kein Status über die Dauer der Verarbeitung einer einzelnen eingehenden Webanfrage hinaus bestehen bleibt.
- Temporäre Dateien löschen — Wenn Sie Dateien erstellen, werden sie in einem Dateisystem gespeichert und nehmen einen Teil der Speicherzuweisung Ihres Dienstes in Anspruch. Um out-of-storage Fehler zu vermeiden, sollten Sie temporäre Dateien nicht für längere Zeit aufbewahren. Richten Sie bei Entscheidungen zum Zwischenspeichern von Dateien die Speichergröße mit der Geschwindigkeit der Bearbeitung von Anfragen ab.
- Instanzstart — App Runner bietet eine Startzeit von fünf Minuten. Ihre Instance muss auf ihren konfigurierten Listening-Ports auf Anfragen warten und innerhalb von fünf Minuten nach dem Start wieder funktionsfähig sein. Während der Startzeit wird App Runner-Instanzen eine virtuelle CPU (vCPU) basierend auf Ihrer vCPU-Konfiguration zugewiesen. Weitere Informationen zur verfügbaren vCPU-Konfiguration finden Sie unterthe section called “Von App Runner unterstützte Konfigurationen”.

Nachdem die Instanz erfolgreich gestartet wurde, wechselt sie in einen Ruhezustand und wartet auf Anfragen. Sie zahlen auf der Grundlage der Startdauer der Instance, wobei die Mindestgebühr eine Minute pro Instance-Start beträgt. Informationen zu Preisen finden Sie unter AWS App Runner -Preise.

Verwenden der App Runner-Konsole

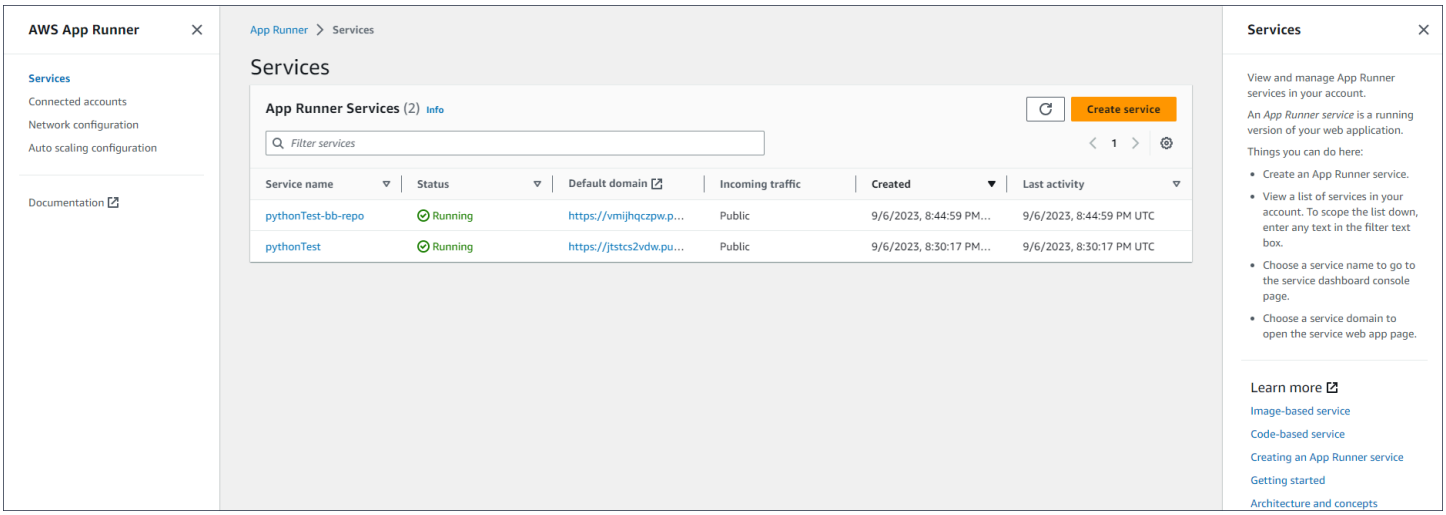
Verwenden Sie die AWS App Runner Konsole, um Ihre App Runner-Dienste und zugehörige Ressourcen wie verbundene Konten zu erstellen, zu verwalten und zu überwachen. Sie können bestehende Dienste anzeigen, neue erstellen und einen Dienst konfigurieren. Sie können den Status eines App Runner-Dienstes sowie Protokolle einsehen, Aktivitäten überwachen und Messwerte verfolgen. Sie können auch zur Website Ihres Dienstes oder zu Ihrem Quell-Repository navigieren.

In den folgenden Abschnitten werden das Layout und die Funktionen der Konsole beschrieben und Sie finden weiterführende Informationen.

Allgemeines Konsolenlayout

Die App Runner-Konsole besteht aus drei Bereichen. Von links nach rechts:

- Navigationsbereich — Ein Seitenbereich, der reduziert oder erweitert werden kann. Verwenden Sie es, um die Konsolenseite der obersten Ebene auszuwählen, die Sie verwenden möchten.
- Inhaltsbereich — Der Hauptteil der Konsolenseite. Verwenden Sie ihn, um Informationen anzuzeigen und Ihre Aufgaben auszuführen.
- Hilfebereich — Ein Seitenbereich mit weiteren Informationen. Erweitern Sie es, um Hilfe zu der Seite zu erhalten, auf der Sie sich befinden. Oder wählen Sie einen beliebigen Informations-Link auf einer Konsolenseite, um kontextbezogene Hilfe zu erhalten.



Die Seite „Dienste“

Auf der Seite Dienste werden die App Runner-Dienste in Ihrem Konto aufgeführt. Sie können die Liste mithilfe des Filtertextfeldes einschränken.

Um zur Seite „Dienste“ zu gelangen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Services.

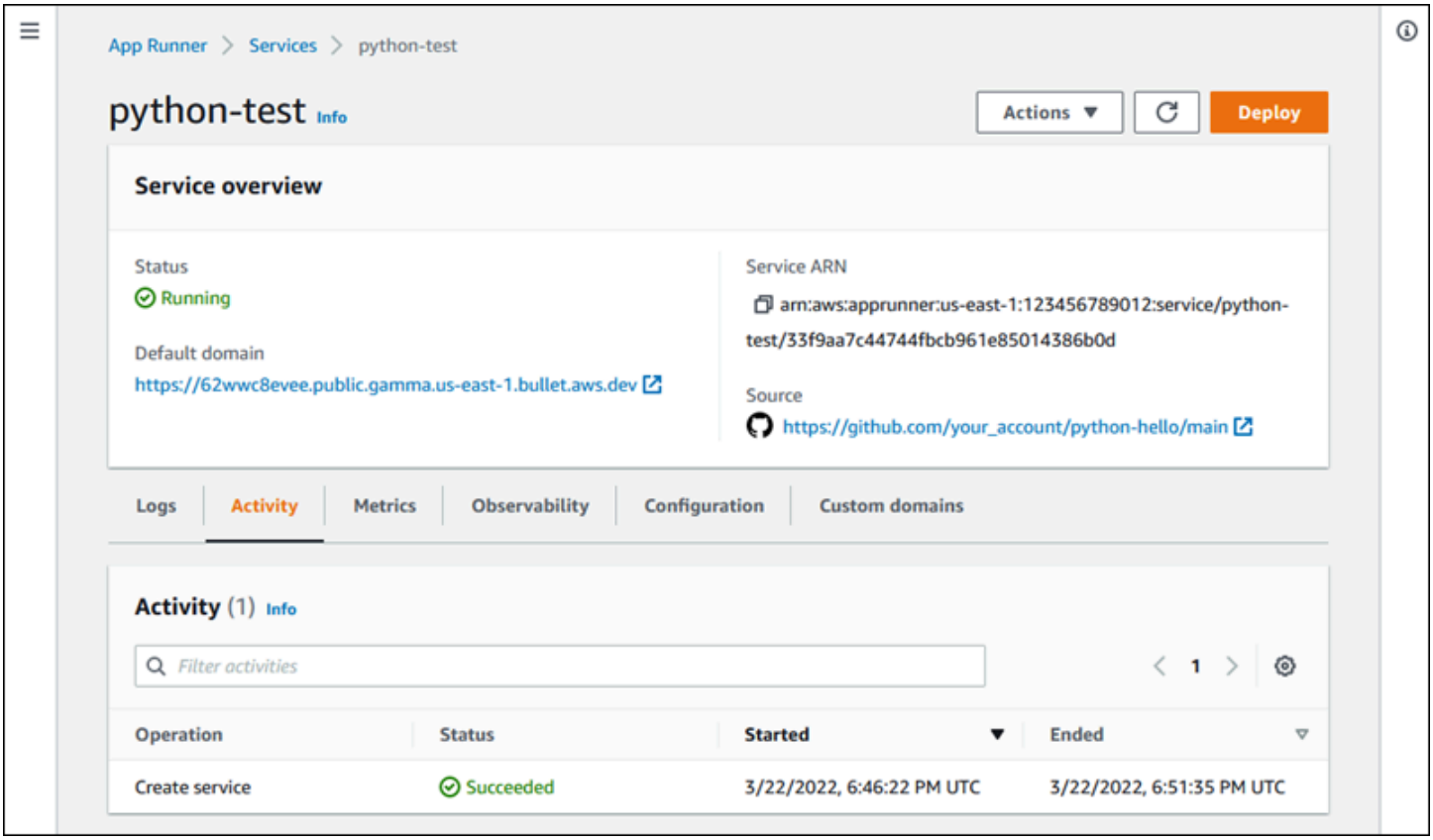
Dinge, die du hier tun kannst:

- Erstellen Sie einen App Runner-Dienst. Weitere Informationen finden Sie unter the section called “Erstellung”.
- Wählen Sie einen Dienstnamen, um zur Konsolenseite des Service-Dashboards zu gelangen.
- Wählen Sie eine Dienstdomäne aus, um die Service-Web-App-Seite zu öffnen.

Die Service-Dashboard-Seite

Auf der Service-Dashboard-Seite können Sie Informationen zu einem App Runner-Dienst anzeigen und ihn verwalten. Oben auf der Seite können Sie den Dienstnamen sehen.

Um zum Service-Dashboard zu gelangen, navigieren Sie zur Seite Dienste (siehe vorheriger Abschnitt) und wählen Sie dann Ihren App Runner-Dienst aus.



Der Abschnitt Serviceübersicht enthält grundlegende Informationen zum App Runner-Dienst und Ihrer Anwendung. Dinge, die du hier tun kannst:

- Zeigen Sie Servicedetails wie Status, Zustand und ARN an.
- Navigieren Sie zur Standarddomäne — der Domäne, die App Runner für die Webanwendung bereitstellt, die in Ihrem Dienst ausgeführt wird. Dies ist eine Subdomain in der `awsapprunner.com` Domain, die App Runner gehört.
- Navigieren Sie zu dem Quell-Repository, das für den Service bereitgestellt wurde.
- Starten Sie eine Bereitstellung des Quell-Repositorys für Ihren Service.
- Unterbrechen Sie Ihren Service, setzen Sie ihn fort und löschen Sie ihn.

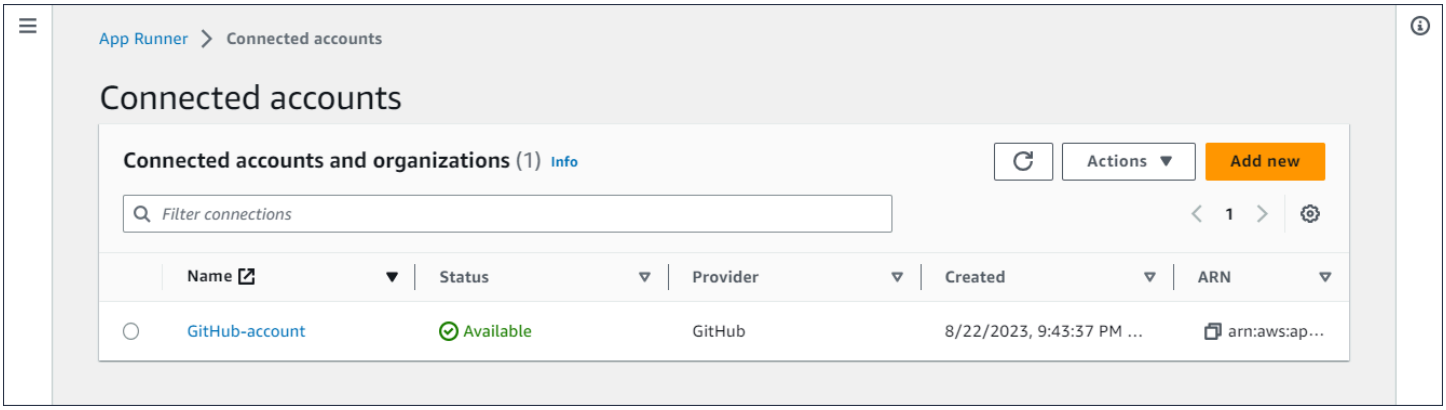
Die Tabs unter der Serviceübersicht dienen der Serviceverwaltung und -beobachtbarkeit.

Die Seite Verbundene Konten

Auf der Seite Verbundene Konten werden App Runner-Verbindungen zu Quellcode-Repository-Anbietern in Ihrem Konto aufgeführt. Sie können die Liste mithilfe des Filtertextfeldes einschränken. Weitere Informationen zu verbundenen Konten finden Sie unterthe section called “Verbindungen”.

Um zur Seite Verbundene Konten zu gelangen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Verbundene Konten aus.



Dinge, die du hier tun kannst:

- Sehen Sie sich eine Liste der Verbindungen mit Repository-Anbietern in Ihrem Konto an. Um die Liste einzugrenzen, geben Sie einen beliebigen Text in das Filtertextfeld ein.
- Wählen Sie einen Verbindungsnamen, um zum entsprechenden Anbieterkonto oder zur entsprechenden Organisation zu gelangen.
- Wählen Sie eine Verbindung aus, um den Handshake für eine Verbindung abzuschließen, die Sie gerade hergestellt haben (als Teil der Erstellung eines Dienstes), oder um die Verbindung zu löschen.

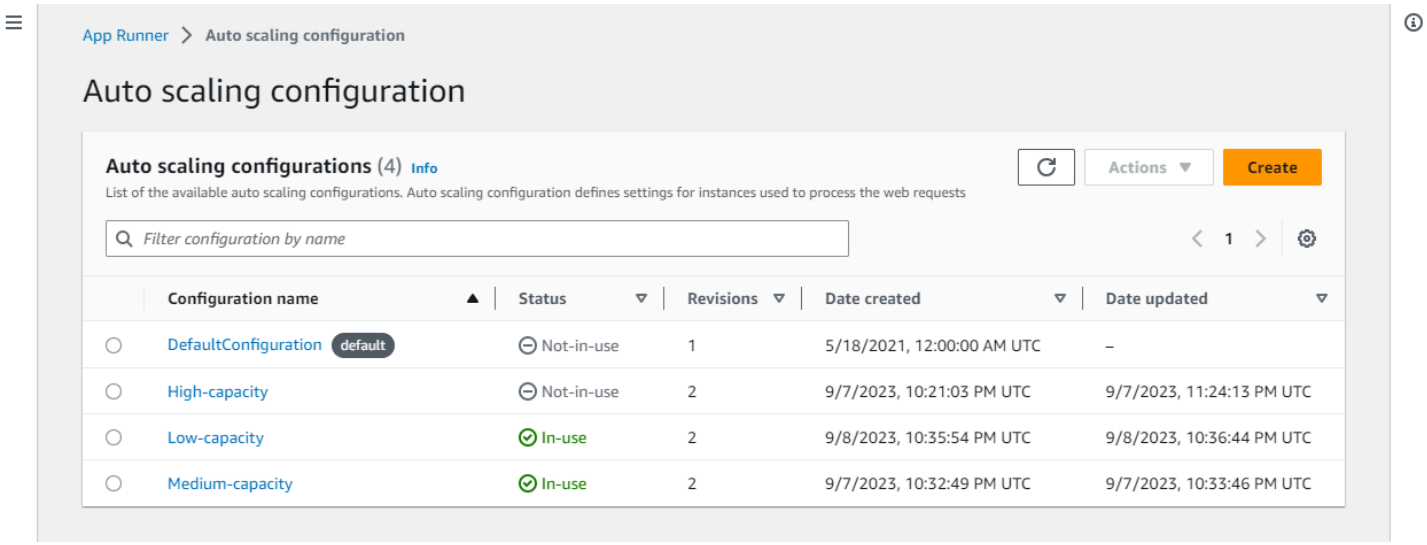
Die Seite mit den Auto Scaling-Konfigurationen

Auf der Seite mit den Auto Scaling-Konfigurationen sind die Auto Scaling-Konfigurationen aufgeführt, die Sie in Ihrem Konto eingerichtet haben. Sie können einige Parameter konfigurieren, um das Auto Scaling-Verhalten anzupassen, und sie in verschiedenen Konfigurationen speichern, die Sie später

einem oder mehreren App Runner-Diensten zuweisen können. Sie können die Liste mithilfe des Filtertextfeldes einschränken. Weitere Informationen zu Auto Scaling-Konfigurationen finden Sie unterAuto Scaling für einen Service verwalten.

Um zur Auto Scaling-Konfigurationsseite zu gelangen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Auto Scaling-Konfiguration aus.



Dinge, die Sie hier tun können:

- Sehen Sie sich die Liste der vorhandenen Auto Scaling-Konfigurationen in Ihrem Konto an.
- Erstellen Sie eine neue Auto Scaling-Konfiguration oder eine Revision für eine bestehende.
- Legen Sie eine Auto Scaling-Konfiguration als Standard für neue Dienste fest, die Sie erstellen.
- Löschen Sie eine Konfiguration.
- Wählen Sie den Namen einer Konfiguration aus, um zum Bereich Auto Scaling-Revisionen zu gelangen, in dem Sie Revisionen verwalten können.

Verwaltung Ihres App Runner-Dienstes

In diesem Kapitel wird beschrieben, wie Sie Ihren AWS App Runner Service verwalten. In diesem Kapitel erfahren Sie, wie Sie den Lebenszyklus Ihres Dienstes verwalten: einen Dienst erstellen, konfigurieren und löschen, neue Anwendungsversionen für Ihren Dienst bereitstellen und die Verfügbarkeit Ihres Webdienstes kontrollieren, indem Sie Ihren Dienst pausieren und wieder aufnehmen. Sie lernen auch, wie Sie andere Aspekte Ihres Dienstes verwalten können, wie Verbindungen und Auto Scaling.

Themen

Einen App Runner-Dienst erstellen

AWS App Runner automatisiert den Übergang von einem Container-Image oder einem Quellcode-Repository zu einem laufenden Webservice, der automatisch skaliert wird. Sie verweisen App Runner auf Ihr Quellbild oder Ihren Quellcode und geben dabei nur eine kleine Anzahl erforderlicher Einstellungen an. App Runner erstellt Ihre Anwendung bei Bedarf, stellt Rechenressourcen bereit und stellt Ihre Anwendung so bereit, dass sie darauf ausgeführt wird.

Wenn Sie einen Dienst erstellen, erstellt App Runner eine Dienstressource. In einigen Fällen müssen Sie möglicherweise eine Verbindungsressource bereitstellen. Wenn Sie die App Runner-Konsole verwenden, erstellt die Konsole implizit die Verbindungsressource. Weitere Informationen zu App Runner-Ressourcentypen finden Sie unterthe section called “App Runner-Ressourcen”. Diese Ressourcentypen haben Kontingente, die jeweils Ihrem Konto zugeordnet sind AWS-Region. Weitere Informationen finden Sie unter the section called “App Runner-Ressourcenkontingente”.

Das Verfahren zur Erstellung eines Dienstes unterscheidet sich je nach Quelltyp und Anbieter geringfügig. In diesem Thema werden verschiedene Verfahren zum Erstellen dieser Quelltypen behandelt, sodass Sie das für Ihre Situation geeignete Verfahren anwenden können. Informationen zum Starten einer grundlegenden Prozedur mit einem Codebeispiel finden Sie unterErste Schritte.

Voraussetzungen

Bevor Sie Ihren App Runner-Dienst erstellen, stellen Sie sicher, dass Sie die folgenden Aktionen ausführen:

- Führen Sie die Einrichtungsschritte unter ausEinrichtung.
- Stellen Sie sicher, dass Ihre Anwendungsquelle bereit ist. Sie können entweder ein Code-Repository in GitHubBitbucket oder ein Container-Image in Amazon Elastic Container Registry (Amazon ECR) verwenden, um einen App Runner-Service zu erstellen.

Einen Service erstellen

In diesem Abschnitt wird der Erstellungsprozess für die beiden App Runner-Servicetypen beschrieben: basierend auf Quellcode und basierend auf einem Container-Image.

> **Note**
>
> Wenn Sie einen VPC-Connector für ausgehenden Verkehr für einen Dienst erstellen, tritt beim darauffolgenden Dienststartvorgang eine einmalige Latenz auf. Sie können diese Konfiguration für einen neuen Dienst bei der Erstellung oder später mit einem Service-Update festlegen. Weitere Informationen finden Sie Einmalige Latenz im Kapitel Networking with App Runner in diesem Handbuch.

Erstellen Sie einen Dienst aus einem Code-Repository

In den folgenden Abschnitten wird gezeigt, wie Sie einen App Runner-Dienst erstellen, wenn Ihre Quelle ein Code-Repository in GitHuboder Bitbucket ist. Wenn du ein Code-Repository verwendest, muss App Runner eine Verbindung zur Anbieterorganisation oder zum Konto herstellen. Daher müssen Sie beim Aufbau dieser Verbindung helfen. Weitere Informationen zu App Runner-Verbindungen finden Sie unterthe section called “Verbindungen”.

Wenn Sie den Dienst erstellen, erstellt App Runner ein Docker-Image, das Ihren Anwendungscode und Ihre Abhängigkeiten enthält. Anschließend wird ein Dienst gestartet, der eine Container-Instance dieses Images ausführt.

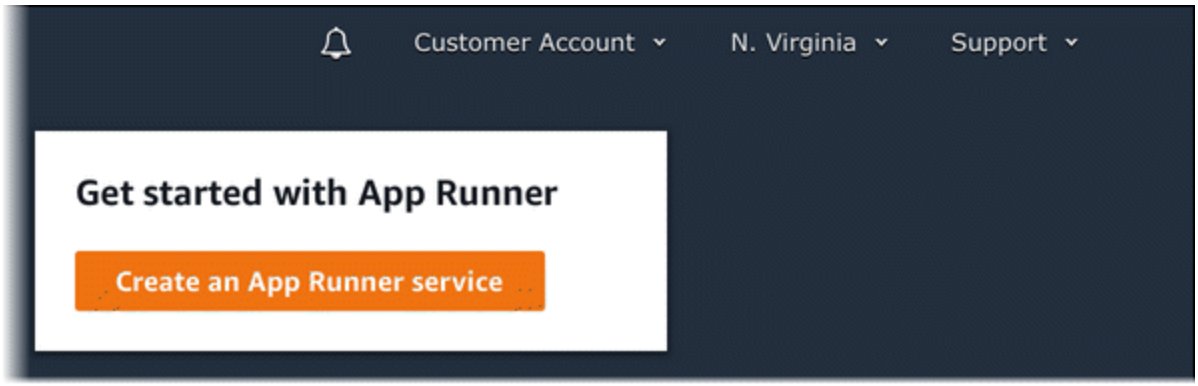
Einen Dienst mithilfe der App Runner-Konsole aus Code erstellen

Um einen App Runner-Dienst mit der Konsole zu erstellen

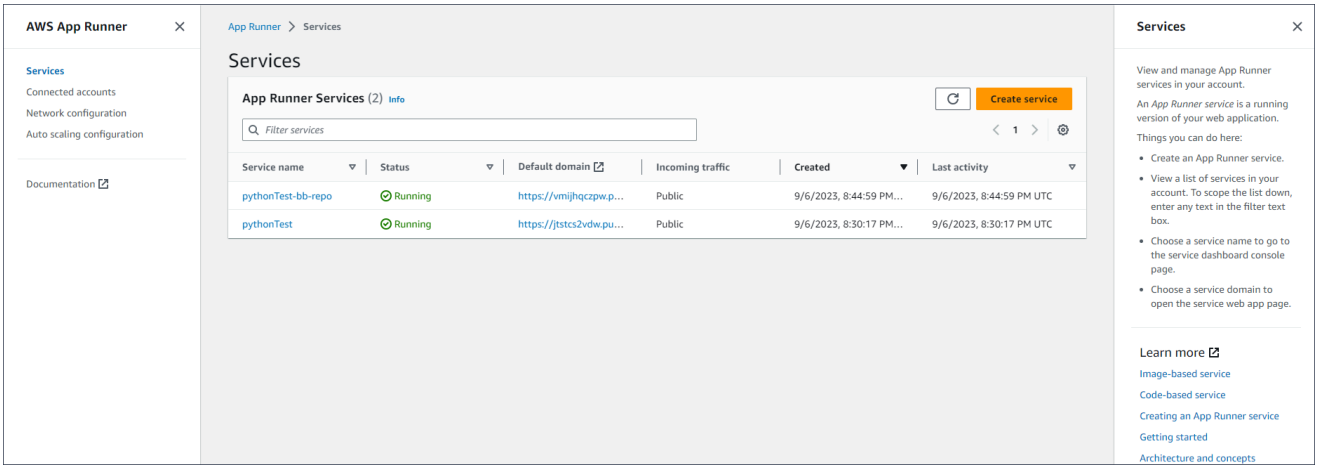
1. Konfigurieren Sie Ihren Quellcode.

   a. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.

   b. Wenn für noch AWS-Konto keine App Runner-Dienste verfügbar sind, wird die Startseite der Konsole angezeigt. Wählen Sie App Runner-Dienst erstellen aus.



   Wenn es AWS-Konto bereits Dienste gibt, wird die Seite Dienste mit einer Liste Ihrer Dienste angezeigt. Wählen Sie Create service.



   c. Wählen Sie auf der Seite Quelle und Bereitstellung im Abschnitt Quelle für Repository-Typ die Option Quellcode-Repository aus.

d. Wählen Sie einen Anbietertyp aus. Wähle entweder GitHuboder Bitbucket.

e. Wähle als Nächstes ein Konto oder eine Organisation für den Anbieter aus, den du zuvor verwendet hast, oder wähle Neu hinzufügen. Gehen Sie dann durch den Prozess, bei dem Sie Ihre Code-Repository-Anmeldeinformationen angeben und ein Konto oder eine Organisation auswählen, mit der Sie sich verbinden möchten.

f. Wählen Sie für Repository das Repository aus, das Ihren Anwendungscode enthält.

g. Wählen Sie für Branch den Branch aus, den Sie bereitstellen möchten.

h. Geben Sie unter Quellverzeichnis das Verzeichnis im Quell-Repository ein, in dem Ihr Anwendungscode und Ihre Konfigurationsdateien gespeichert sind.

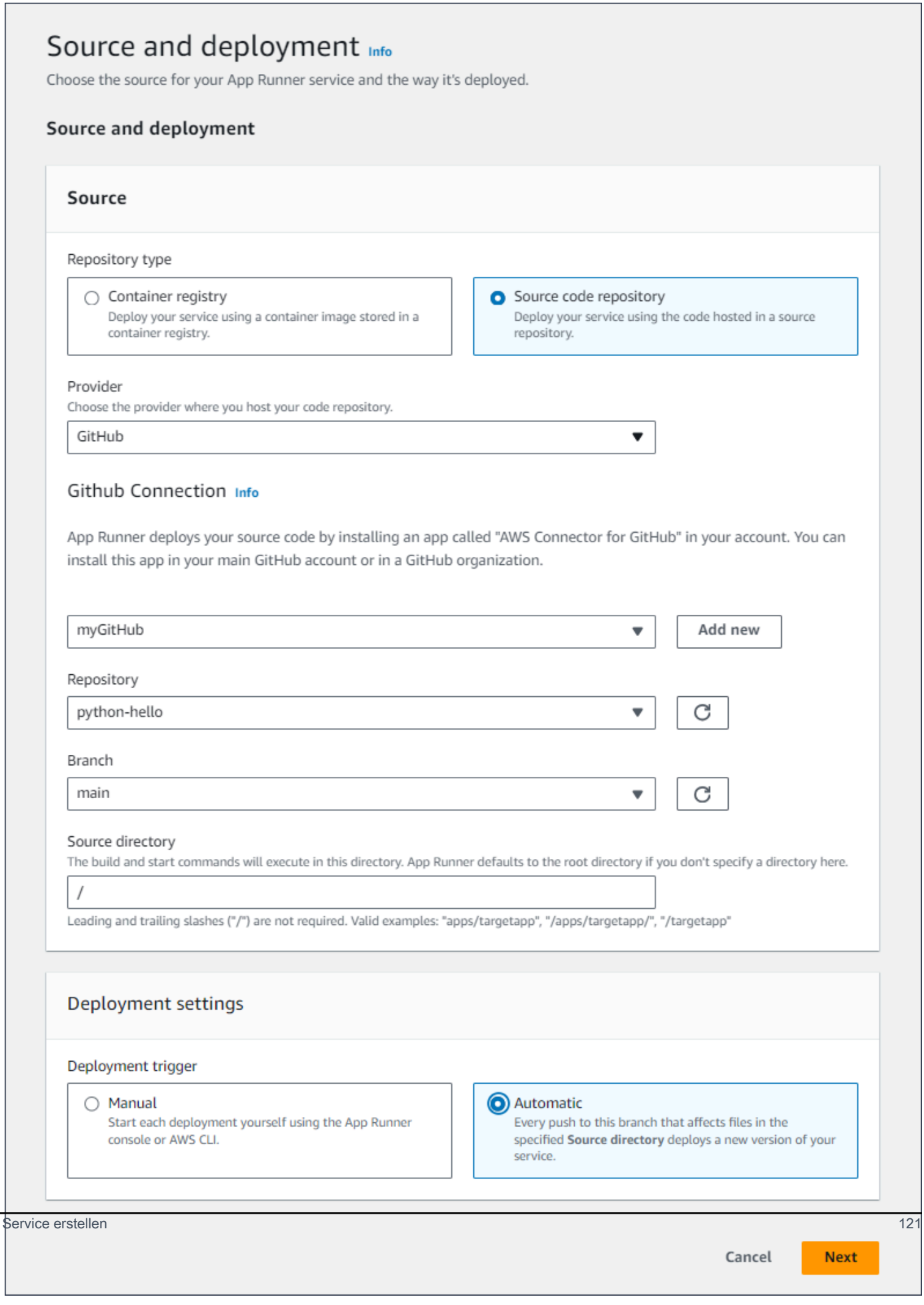
> **Note**
>
> Die Befehle build und start werden in dem von Ihnen angegebenen Quellverzeichnis ausgeführt. App Runner behandelt den Pfad ab Root als absoluten Pfad. Wenn Sie hier keinen Wert angeben, verwendet das Verzeichnis standardmäßig das Stammverzeichnis des Repositorys.

2. Konfigurieren Sie Ihre Bereitstellungen.

a. Wählen Sie im Abschnitt Bereitstellungseinstellungen die Option Manuell oder Automatisch aus.

Weitere Informationen zu Bereitstellungsmethoden finden Sie unterthe section called “Bereitstellungsmethoden”.

b. Wählen Sie Weiter aus.


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. Konfigurieren Sie den Anwendungsbuild.

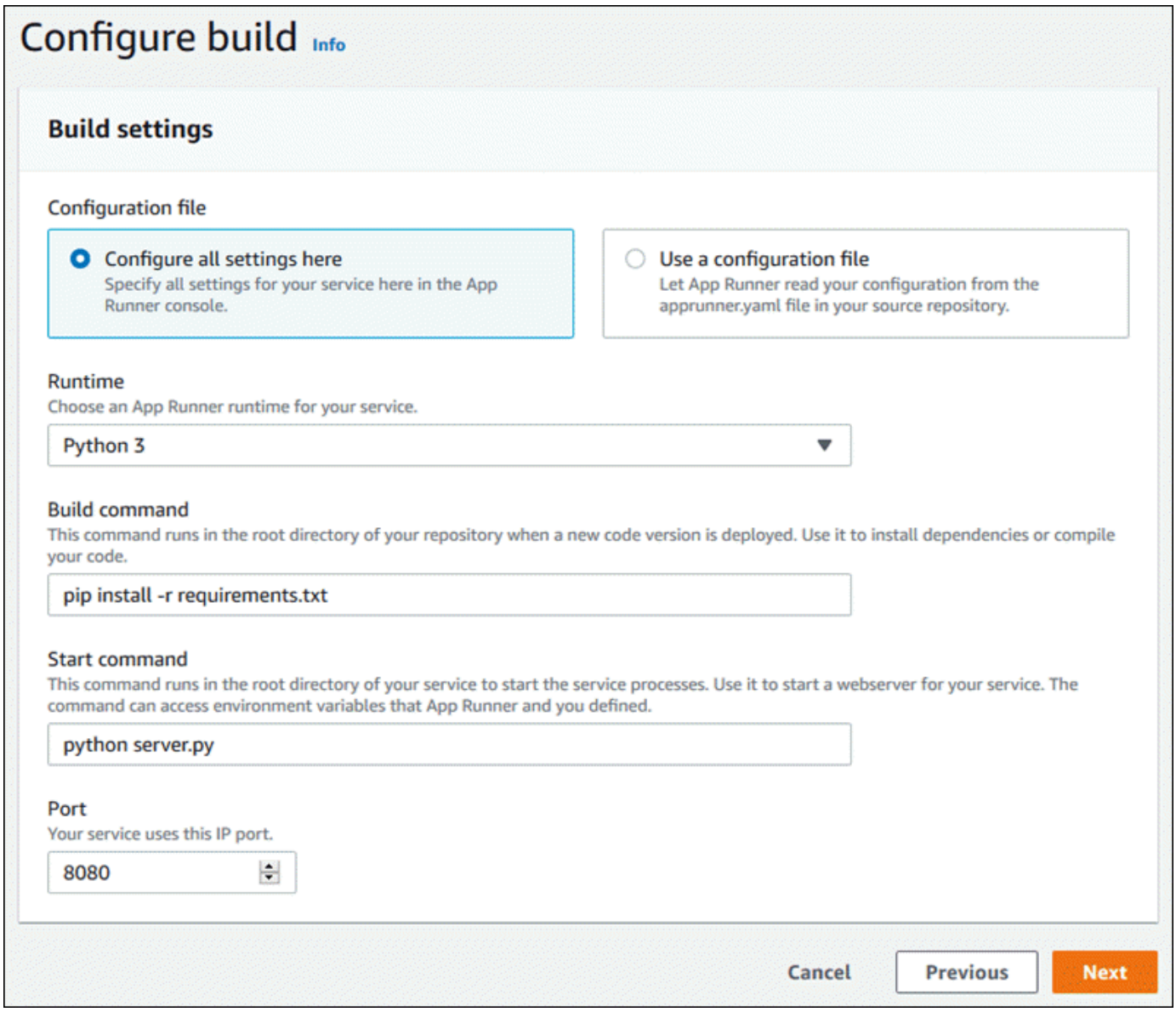
   a. Wählen Sie auf der Seite „Build konfigurieren“ unter Konfigurationsdatei die Option Alle Einstellungen hier konfigurieren aus, wenn Ihr Repository keine App Runner-Konfigurationsdatei enthält, oder wählen Sie in diesem Fall eine Konfigurationsdatei verwenden aus.

   > **Note**
   >
   > Eine App Runner-Konfigurationsdatei ist eine Möglichkeit, Ihre Build-Konfiguration als Teil Ihrer Anwendungsquelle beizubehalten. Wenn Sie eine angeben, liest App Runner einige Werte aus der Datei und lässt Sie sie nicht in der Konsole festlegen.

   b. Geben Sie die folgenden Build-Einstellungen an:

      - Laufzeit — Wählen Sie eine bestimmte verwaltete Laufzeit für Ihre Anwendung aus.
      - Build-Befehl — Geben Sie einen Befehl ein, der Ihre Anwendung aus dem Quellcode erstellt. Dabei kann es sich um ein sprachspezifisches Tool oder ein mit Ihrem Code bereitgestelltes Skript handeln.
      - Startbefehl — Geben Sie den Befehl ein, mit dem Ihr Webservice gestartet wird.
      - Port — Geben Sie den IP-Port ein, den Ihr Webservice abhört.

   c. Wählen Sie Weiter aus.



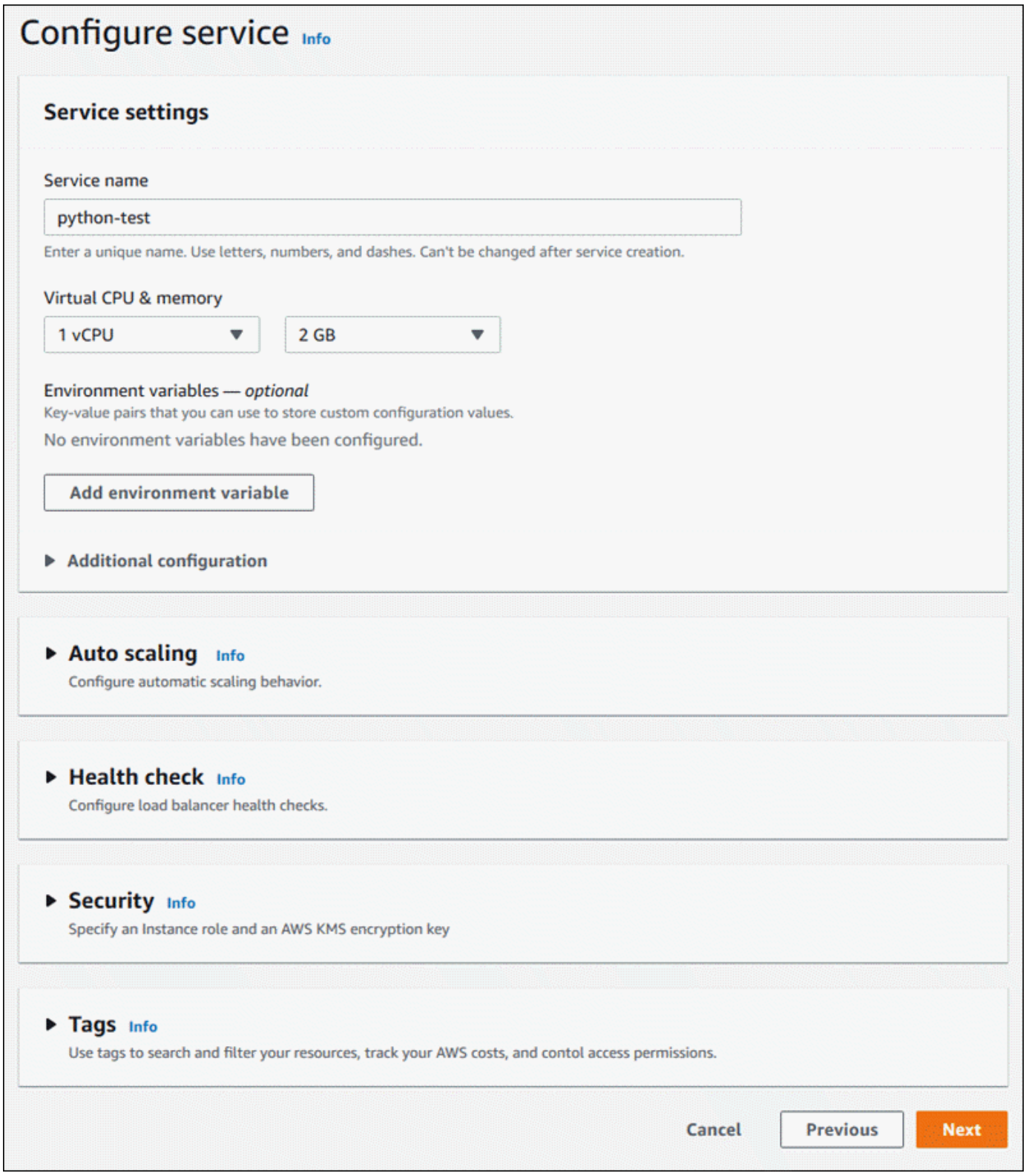
4. Konfigurieren Sie Ihren Dienst.

   a. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Diensteinstellungen einen Dienstnamen ein.

> **Note**
>
> Alle anderen Diensteinstellungen sind entweder optional oder haben von der Konsole bereitgestellte Standardwerte.

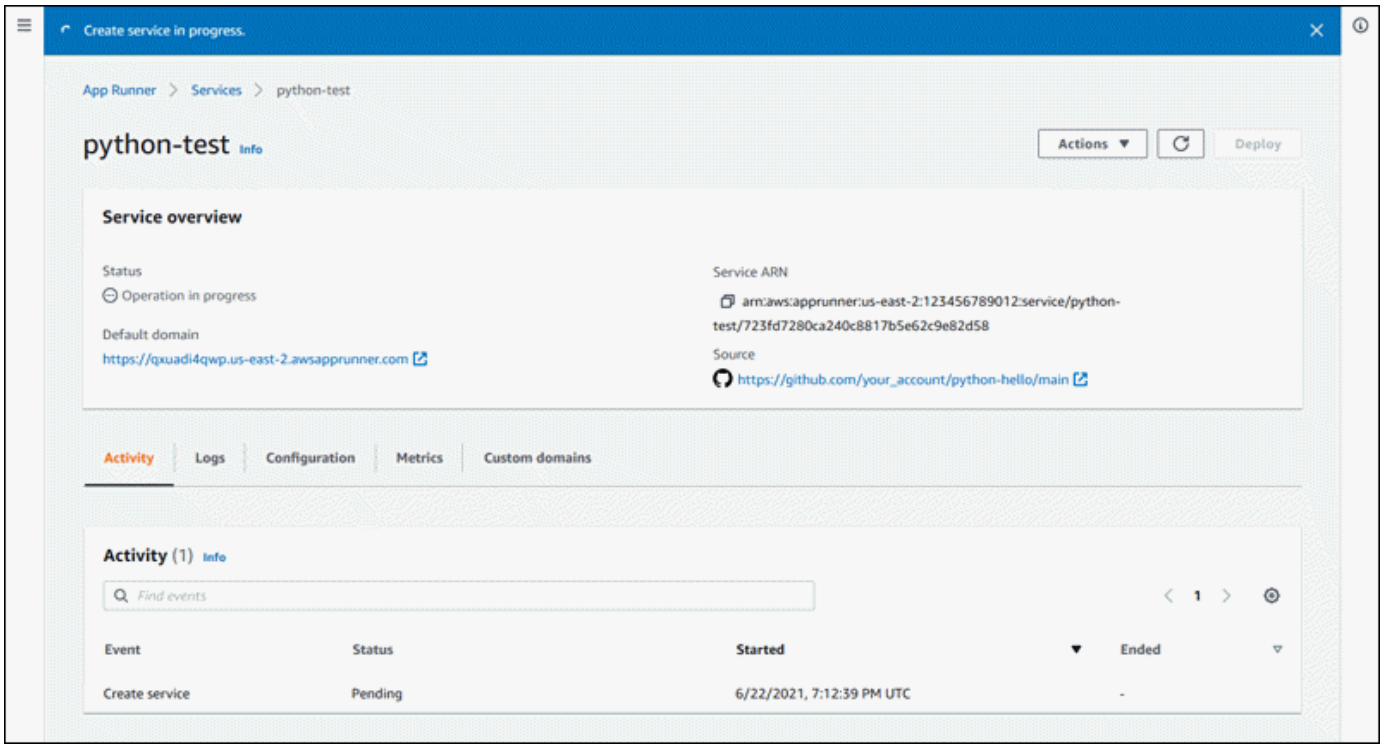
b. Sie können optional weitere Einstellungen ändern oder hinzufügen, um Ihre Anwendungsanforderungen zu erfüllen.

c. Wählen Sie Weiter aus.



5. Überprüfen Sie auf der Seite Überprüfen und erstellen alle von Ihnen eingegebenen Details und wählen Sie dann Erstellen und bereitstellen aus.

Ergebnis: Wenn der Service erfolgreich erstellt wurde, zeigt die Konsole das Service-Dashboard mit einer Serviceübersicht über den neuen Service an.



6. Stellen Sie sicher, dass Ihr Dienst ausgeführt wird.

   a. Warten Sie auf der Service-Dashboard-Seite, bis der Dienststatus Running lautet.

   b. Wählen Sie den Standarddomänenwert. Es ist die URL zur Website Ihres Dienstes.

   c. Verwenden Sie Ihre Website und stellen Sie sicher, dass sie ordnungsgemäß funktioniert.

Einen Dienst aus Code mithilfe der App Runner API erstellen oder AWS CLI

Um einen Dienst mit der App Runner API oder zu erstellen AWS CLI, rufen Sie die `CreateService` API-Aktion auf. Weitere Informationen sowie ein Beispiel finden Sie unter CreateService. Wenn du zum ersten Mal einen Service mit einer bestimmten Organisation oder einem Konto für ein Quellcode-Repository (GitHub oder Bitbucket) erstellst, rufe CreateConnectionzunächst an. Dadurch wird eine Verbindung zwischen App Runner und der Organisation oder dem Konto des Repository-Anbieters hergestellt. Weitere Informationen zu App Runner-Verbindungen finden Sie unterthe section called “Verbindungen”.

Wenn der Aufruf eine erfolgreiche Antwort zurückgibt und ein Service-Objekt angezeigt wird`"Status": "CREATING"`, beginnt Ihr Service mit der Erstellung.

Ein Beispiel für einen Aufruf finden Sie unter Erstellen eines Quellcode-Repository-Service in der AWS App Runner API-Referenz

Einen Service aus einem Amazon ECR-Image erstellen

In den folgenden Abschnitten wird gezeigt, wie Sie einen App Runner-Service erstellen, wenn Ihre Quelle ein in Amazon ECR gespeichertes Container-Image ist. Amazon ECR ist ein AWS-Service. Um einen Service auf der Grundlage eines Amazon ECR-Images zu erstellen, stellen Sie App Runner daher eine Zugriffsrolle mit den erforderlichen Amazon ECR-Aktionsberechtigungen zur Verfügung.

> **Note**
>
> In Amazon ECR Public gespeicherte Bilder sind öffentlich verfügbar. Wenn Ihr Bild also in Amazon ECR Public gespeichert ist, ist keine Zugriffsrolle erforderlich.

Wenn Ihr Service erstellt wird, startet App Runner einen Service, der eine Container-Instance des von Ihnen bereitgestellten Images ausführt. In diesem Fall gibt es keine Erstellungsphase.

Weitere Informationen finden Sie unter Bildbasierter Dienst.

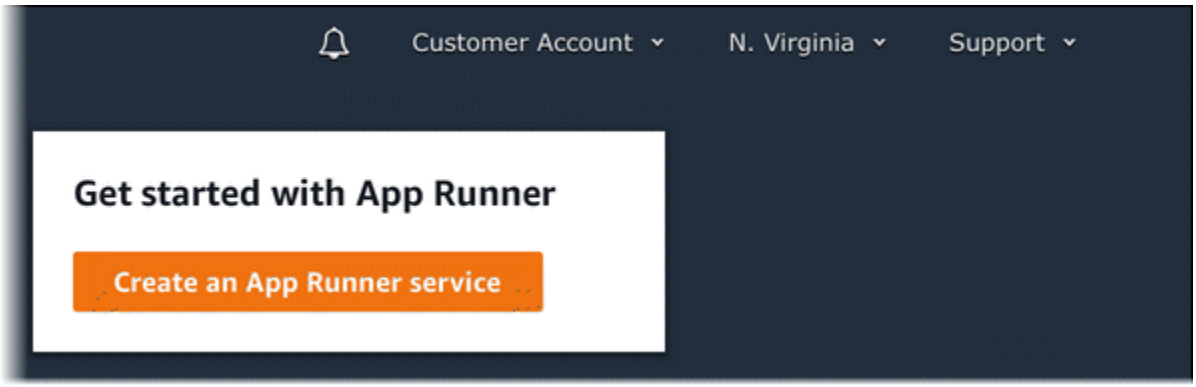
Mit der App Runner-Konsole einen Dienst aus einem Image erstellen

Um einen App Runner-Dienst mit der Konsole zu erstellen

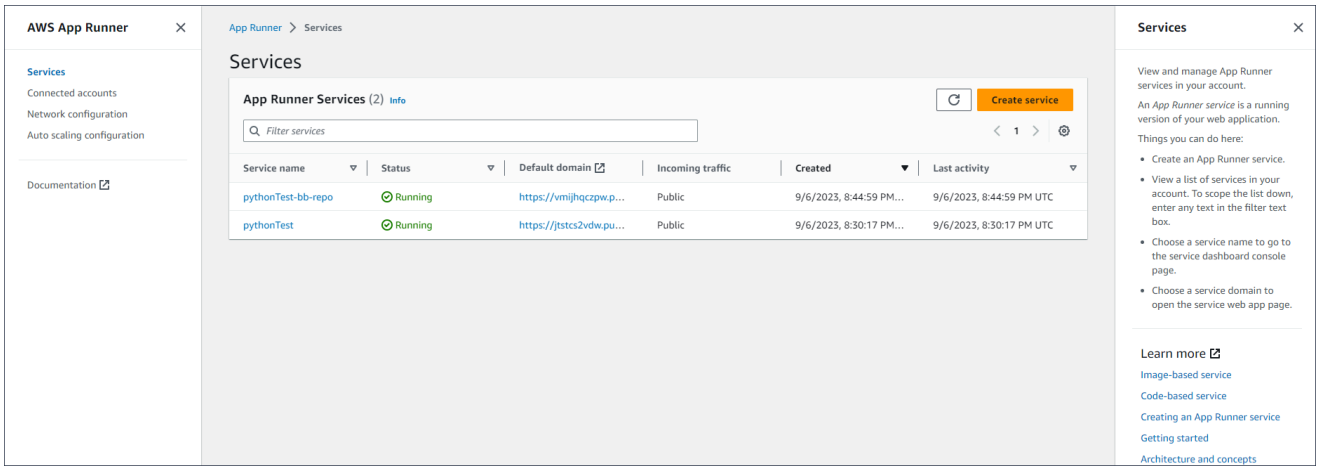
1. Konfigurieren Sie Ihren Quellcode.

   a. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.

   b. Wenn für noch AWS-Konto keine App Runner-Dienste verfügbar sind, wird die Startseite der Konsole angezeigt. Wählen Sie App Runner-Dienst erstellen aus.



Wenn es AWS-Konto bereits Dienste gibt, wird die Seite Dienste mit einer Liste Ihrer Dienste angezeigt. Wählen Sie Create service.



c. Wählen Sie auf der Seite Quelle und Bereitstellung im Abschnitt Quelle für Repository-Typ die Option Container-Registrierung aus.

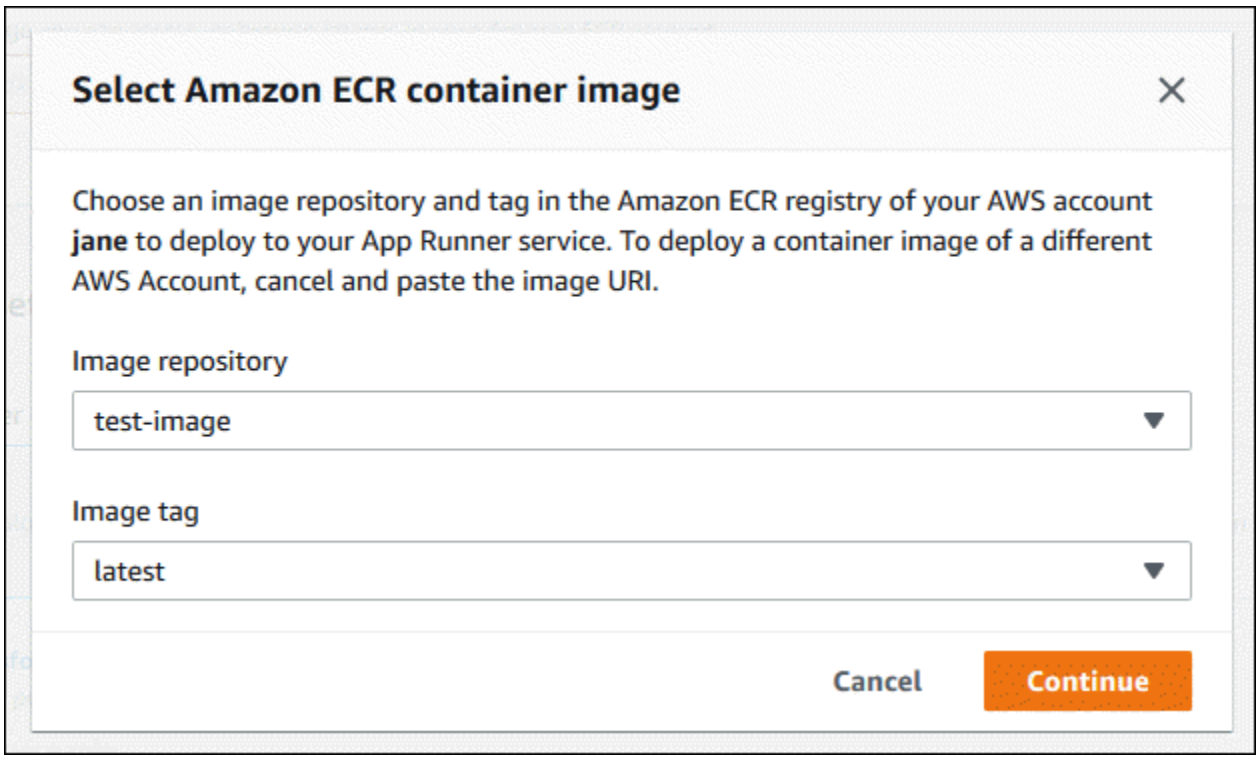
d. Wählen Sie unter Anbieter den Anbieter aus, bei dem Ihr Image gespeichert ist:

- Amazon ECR — Ein privates Bild, das in Amazon ECR gespeichert ist.
- Amazon ECR Public — Ein öffentlich lesbares Bild, das in Amazon ECR Public gespeichert ist.

e. Wählen Sie für Container-Image-URI die Option Browse aus.

f. Wählen Sie im Dialogfeld Amazon ECR-Container-Image auswählen für Image-Repository das Repository aus, das Ihr Image enthält.

g. Wählen Sie unter Image-Tag das spezifische Image-Tag aus, das Sie bereitstellen möchten (z. B. das neueste), und wählen Sie dann Weiter.



2. Konfigurieren Sie Ihre Bereitstellungen.

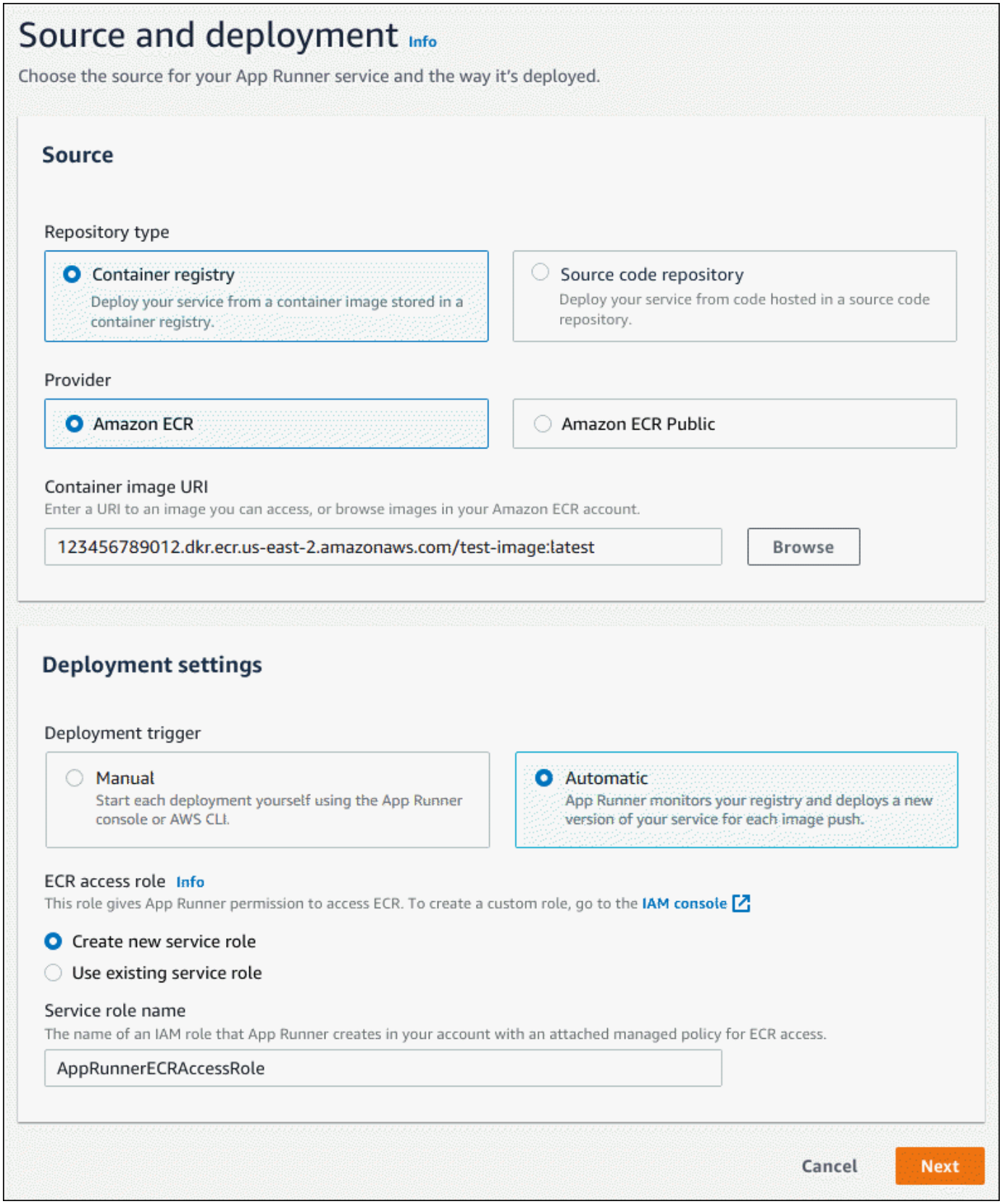
   a. Wählen Sie im Abschnitt Bereitstellungseinstellungen die Option Manuell oder Automatisch aus.

   > **Note**
   >
   > App Runner unterstützt keine automatische Bereitstellung für öffentliche Amazon ECR-Images und für Bilder in einem Amazon ECR-Repository, das zu einem anderen AWS Konto gehört als dem, in dem sich Ihr Service befindet.

   Weitere Informationen zu Bereitstellungsmethoden finden Sie unter. the section called “Bereitstellungsmethoden”

   b. [Amazon ECR-Anbieter] Wählen Sie für die ECR-Zugriffsrolle eine bestehende Servicerolle in Ihrem Konto aus oder erstellen Sie eine neue Rolle. Wenn Sie die manuelle Bereitstellung verwenden, können Sie sich auch dafür entscheiden, die IAM-Benutzerrolle zum Zeitpunkt der Bereitstellung zu verwenden.

   c. Wählen Sie Weiter aus.


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

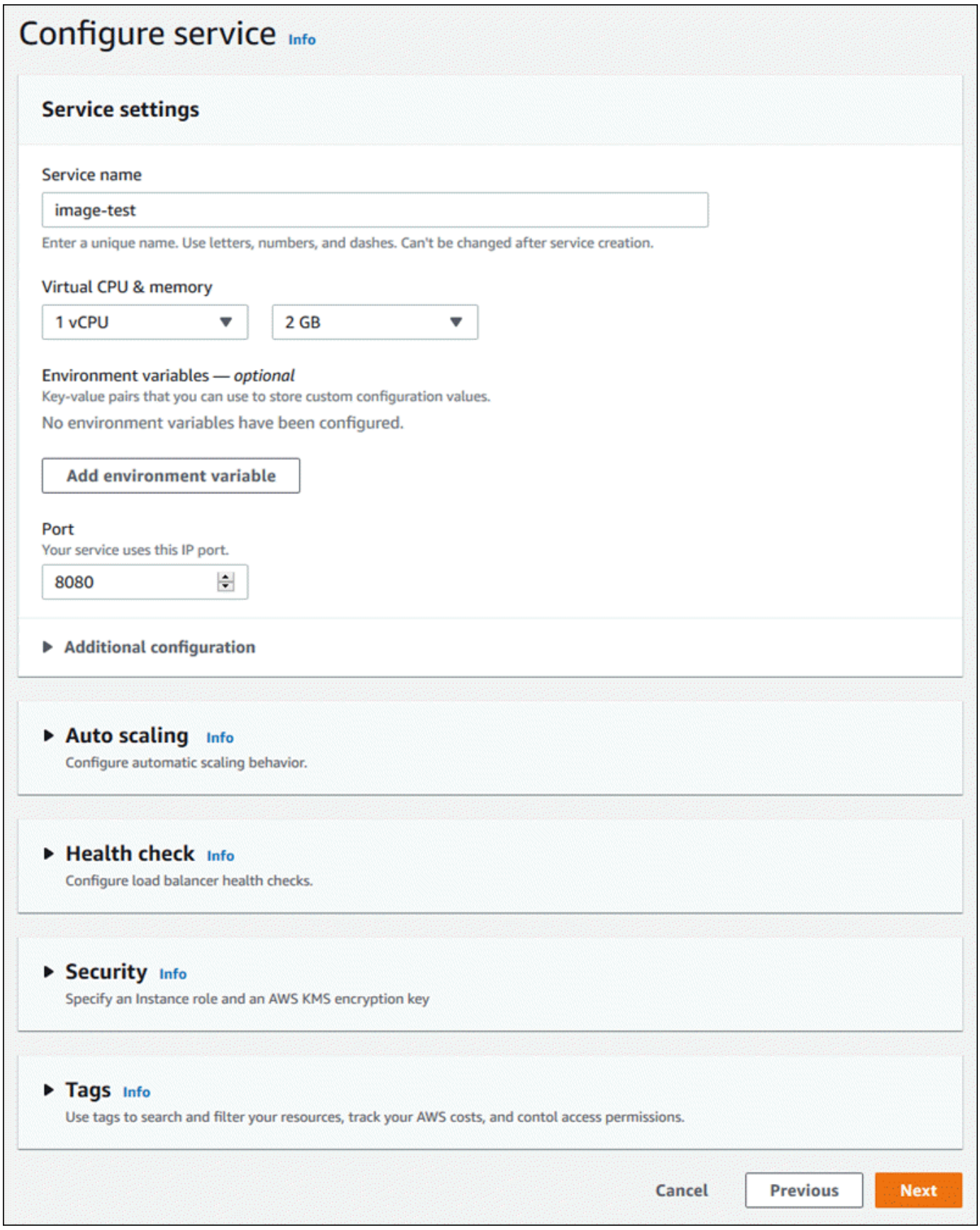
3. Konfigurieren Sie Ihren Dienst.

   a. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Diensteinstellungen einen Dienstnamen und den IP-Port ein, den Ihre Service-Website abhört.

   > **Note**
   >
   > Alle anderen Diensteinstellungen sind entweder optional oder haben von der Konsole bereitgestellte Standardeinstellungen.

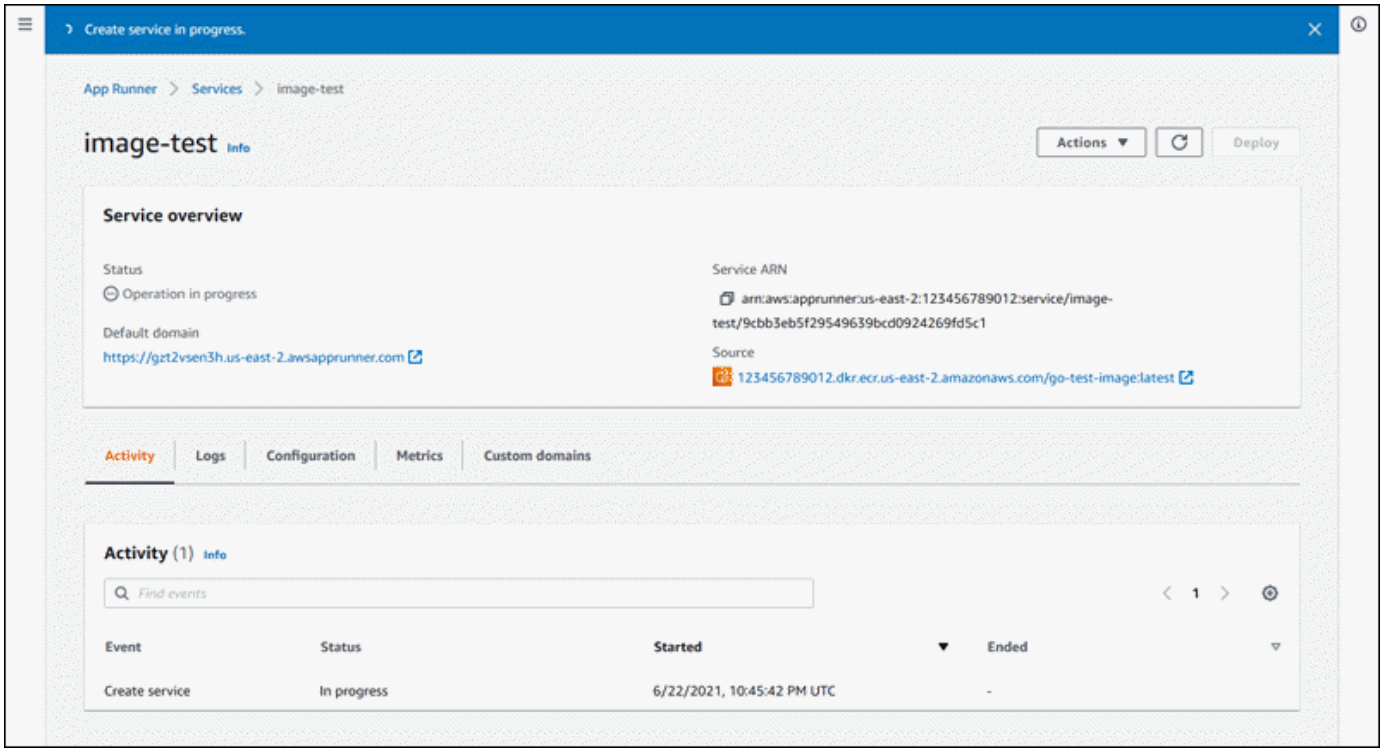
   b. (Optional) Ändern oder fügen Sie weitere Einstellungen hinzu, um sie an die Anforderungen Ihrer Anwendung anzupassen.

   c. Wählen Sie Weiter aus.


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. Überprüfen Sie auf der Seite Überprüfen und erstellen alle von Ihnen eingegebenen Details und wählen Sie dann Erstellen und bereitstellen aus.

   Ergebnis: Wenn der Service erfolgreich erstellt wurde, zeigt die Konsole das Service-Dashboard mit einer Serviceübersicht über den neuen Service an.



5. Stellen Sie sicher, dass Ihr Dienst ausgeführt wird.

   a. Warten Sie auf der Service-Dashboard-Seite, bis der Dienststatus Running lautet.

   b. Wählen Sie den Standarddomänenwert. Es ist die URL zur Website Ihres Dienstes.

   c. Verwenden Sie Ihre Website und stellen Sie sicher, dass sie ordnungsgemäß funktioniert.

Mit der App Runner API einen Dienst aus einem Image erstellen oder AWS CLI

Um einen Dienst mit der App Runner API oder zu erstellen AWS CLI, rufen Sie die CreateServiceAPI-Aktion auf.

Ihre Diensterstellung beginnt, wenn der Aufruf eine erfolgreiche Antwort zurückgibt und ein Service-Objekt angezeigt wird`"Status": "CREATING"`.

Ein Beispiel für einen Aufruf finden Sie unter Create a source image repository service in der AWS App Runner API-Referenz

Einen fehlgeschlagenen App Runner-Dienst neu aufbauen

Wenn Sie beim Erstellen eines App Runner-Dienstes die Fehlermeldung Fehler beim Erstellen erhalten, können Sie einen der folgenden Schritte ausführen.

- Folgen Sie den Schritten unterthe section called “Der Dienst konnte nicht erstellt werden”, um die Ursache des Fehlers zu ermitteln.
- Wenn Sie einen Fehler in der Quelle oder Konfiguration finden, nehmen Sie die erforderlichen Änderungen vor und erstellen Sie dann Ihren Service neu.
- Wenn Ihr Dienst aufgrund eines vorübergehenden Problems mit App Runner ausgefallen ist, erstellen Sie Ihren ausgefallenen Dienst neu, ohne Änderungen an der Quelle oder Konfiguration vorzunehmen.

Sie können Ihren ausgefallenen Dienst entweder über die App Runner-Konsole oder die App Runner-API neu aufbauen oder AWS CLI.

Einen ausgefallenen App Runner-Dienst mithilfe der App Runner-Konsole neu aufbauen

Rebuild with updates

Das Erstellen eines Dienstes kann aus verschiedenen Gründen fehlschlagen. In diesem Fall ist es wichtig, die Ursache des Problems zu identifizieren und zu beheben, bevor Sie Ihren Service neu erstellen. Weitere Informationen finden Sie unter the section called “Der Dienst konnte nicht erstellt werden”.

Um einen ausgefallenen Dienst mit Updates neu aufzubauen

1. Gehen Sie auf Ihrer Serviceseite zur Registerkarte Konfigurationen und wählen Sie Bearbeiten.

   Die Seite öffnet ein Übersichtsfenster, in dem eine Liste all Ihrer Updates angezeigt wird.
2. Nehmen Sie die erforderlichen Änderungen vor und überprüfen Sie sie im Übersichtsfenster.
3. Wählen Sie Speichern und neu erstellen.

   Sie können den Fortschritt auf der Registerkarte Protokolle auf Ihrer Serviceseite überwachen.

Rebuild without updates

Wenn ein vorübergehendes Problem dazu führt, dass Ihre Diensterstellung fehlschlägt, können Sie Ihren Service neu erstellen, ohne seine Quell- oder Konfigurationseinstellungen zu ändern.

Um einen fehlgeschlagenen Dienst ohne Updates neu aufzubauen

- Wählen Sie in der oberen rechten Ecke Ihrer Serviceseite die Option Rebuild aus.

  Sie können den Fortschritt auf der Registerkarte Protokolle auf Ihrer Serviceseite überwachen.
- Wenn Ihr Service nicht erneut erstellt werden kann, folgen Sie den Anweisungen zur Fehlerbehebung unterthe section called “Der Dienst konnte nicht erstellt werden”. Nehmen Sie die erforderlichen Änderungen vor und erstellen Sie dann Ihren Service neu.

Fehlgeschlagener App Runner-Dienst mithilfe der App Runner-API neu aufbauen oder AWS CLI

Rebuild with updates

Um einen ausgefallenen Dienst neu aufzubauen:

1. Folgen Sie den Anweisungen unterthe section called “Der Dienst konnte nicht erstellt werden”, um die Ursache des Fehlers zu ermitteln.
2. Nehmen Sie die erforderlichen Änderungen an dem Branch oder dem Image des Quell-Repositorys oder an der Konfiguration vor, die den Fehler verursacht hat.
3. Erstellen Sie den Vorgang neu, indem Sie die UpdateServiceAPI-Aktion mit den neuen Parametern für das Quellcode-Repository oder das Quell-Image-Repository aufrufen. App Runner ruft den neuesten Commit aus dem Quellcode-Repository ab.

Example Neuaufbau mit Updates

Im folgenden Beispiel wird die Quellkonfiguration eines imagebasierten Dienstes aktualisiert. Der Wert von `Port` wird in geändert. `80`

Die `input.json` Datei für den imagebasierten App Runner-Dienst wird aktualisiert

```
{
```

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

Aufruf der `UpdateService` API-Aktion.

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

Rebuild without updates

Um Ihren fehlgeschlagenen Dienst mithilfe der App Runner API neu aufzubauen AWS CLI, oder rufen Sie die UpdateServiceAPI-Aktion auf, ohne Änderungen an der Quelle oder Konfiguration Ihres Dienstes vorzunehmen. Entscheiden Sie sich nur dann für eine Neuerstellung ohne Aktualisierungen, wenn Ihre Diensterstellung aufgrund eines vorübergehenden Problems mit App Runner fehlgeschlagen ist.

Bereitstellen einer neuen Anwendungsversion in App Runner

Wenn Sie einen Service in erstellen AWS App Runner, konfigurieren Sie eine Anwendungsquelle — ein Container-Image oder ein Quell-Repository. App Runner stellt Ressourcen für die Ausführung Ihres Dienstes bereit und stellt Ihre Anwendung für sie bereit.

In diesem Thema werden Möglichkeiten beschrieben, wie Sie Ihre Anwendungsquelle erneut für Ihren App Runner-Dienst bereitstellen können, wenn eine neue Version verfügbar ist. Dies kann eine neue Image-Version im Image-Repository oder ein neuer Commit im Code-Repository sein. App Runner bietet zwei Methoden für die Bereitstellung in einem Dienst: automatisch und manuell.

Bereitstellungsmethoden

App Runner bietet die folgenden Methoden, mit denen Sie steuern können, wie Anwendungsbereitstellungen initiiert werden.

Automatische Bereitstellung

Verwenden Sie die automatische Bereitstellung, wenn Sie ein kontinuierliches Integrations- und Bereitstellungsverhalten (CI/CD) für Ihren Service wünschen. App Runner überwacht Ihr Image- oder Code-Repository auf Änderungen.

Image-Repository — Immer wenn Sie eine neue Image-Version in Ihr Image-Repository oder einen neuen Commit in Ihr Code-Repository übertragen, stellt App Runner diese automatisch in Ihrem Dienst bereit, ohne dass Sie weitere Maßnahmen ergreifen müssen.

Code-Repository — Immer wenn Sie einen neuen Commit in Ihr Code-Repository übertragen, der Änderungen im Quellverzeichnis vornimmt, stellt App Runner Ihr gesamtes Repository bereit. Da nur Änderungen im Quellverzeichnis eine automatische Bereitstellung auslösen, ist es wichtig zu verstehen, wie sich der Speicherort des Quellverzeichnisses auf den Umfang einer automatisierten Bereitstellung auswirkt.

- Verzeichnis der obersten Ebene (Repository-Stammverzeichnis) — Dies ist der Standardwert, der für das Quellverzeichnis festgelegt wird, wenn Sie einen Service erstellen. Wenn Ihr Quellverzeichnis auf diesen Wert gesetzt ist, bedeutet dies, dass sich das gesamte Repository im Quellverzeichnis befindet. In diesem Fall lösen also alle Commits, die Sie in das Quell-Repository übertragen, ein Deployment aus.
- Jeder Verzeichnispfad, der nicht das Repository-Stammverzeichnis ist (kein Standard) — Da nur Änderungen, die innerhalb des Quellverzeichnisses übertragen werden, ein automatisches Deployment auslösen, lösen alle Änderungen, die an Ihr Repository übertragen werden und sich nicht im Quellverzeichnis befinden, kein automatisches Deployment aus. Daher müssen Sie eine manuelle Bereitstellung verwenden, um Änderungen bereitzustellen, die Sie außerhalb des Quellverzeichnisses übertragen.

> **Note**
>
> App Runner unterstützt keine automatische Bereitstellung für öffentliche Amazon ECR-Images und für Bilder in einem Amazon ECR-Repository, das zu einem anderen AWS Konto gehört als dem, in dem sich Ihr Service befindet.

Manuelle Bereitstellung

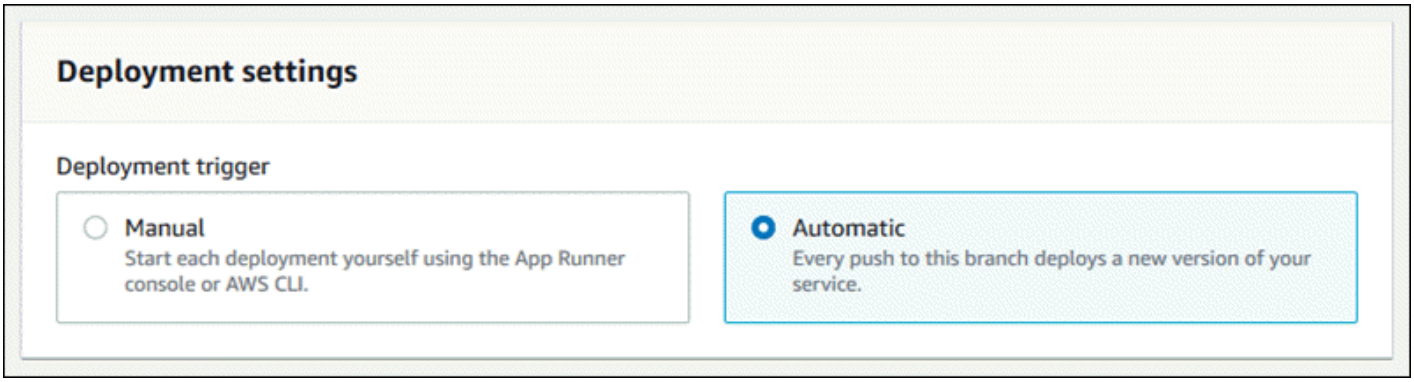
Verwenden Sie die manuelle Bereitstellung, wenn Sie jede Bereitstellung für Ihren Service explizit initiieren möchten. Sie initiieren eine Bereitstellung, wenn das Repository, das Sie für Ihren

Service konfiguriert haben, über eine neue Version verfügt, die Sie bereitstellen möchten. Weitere Informationen finden Sie unter the section called “Manuelles Deployment”.

> **Note**
>
> Wenn Sie eine manuelle Bereitstellung ausführen, stellt App Runner die Quelle aus dem vollständigen Repository bereit.

Sie können die Bereitstellungsmethode für Ihren Service auf folgende Weise konfigurieren:

- Konsole — Wählen Sie für einen neuen Dienst, den Sie erstellen, oder für einen vorhandenen Dienst auf der Konfigurationsseite für Quelle und Bereitstellung im Abschnitt Bereitstellungseinstellungen die Option Manuell oder Automatisch aus.



- API oder AWS CLI — Bei einem Aufruf der UpdateServiceAktion CreateServiceoder legen Sie das `AutoDeploymentsEnabled` Element des SourceConfigurationParameters auf `False` für die manuelle oder `True` für die automatische Bereitstellung fest.

> **Vergleich von automatischen und manuellen Bereitstellungen**
>
> Sowohl automatische als auch manuelle Bereitstellungen führen zu demselben Ergebnis: Beide Methoden stellen das gesamte Repository bereit.
> Der Unterschied zwischen den beiden Methoden ist der Auslösemechanismus:
>
> - Manuelle Bereitstellungen werden durch eine Bereitstellung über die Konsole, einen Aufruf der AWS CLI oder einen Aufruf der App Runner API ausgelöst. Im folgenden Manuelles Deployment Abschnitt werden die Verfahren für diese Verfahren beschrieben.

- Automatische Bereitstellungen werden durch eine Änderung im Inhalt des Quellverzeichnisses ausgelöst.

Manuelles Deployment

Bei der manuellen Bereitstellung müssen Sie jede Bereitstellung für Ihren Service explizit initiieren. Wenn Sie eine neue Version Ihres Anwendungs-Images oder -Codes zur Bereitstellung bereit haben, können Sie in den folgenden Abschnitten nachlesen, wie Sie eine Bereitstellung mithilfe der Konsole und der API durchführen.

> **Note**
>
> Wenn Sie eine manuelle Bereitstellung ausführen, stellt App Runner den Quellcode aus dem vollständigen Repository bereit.

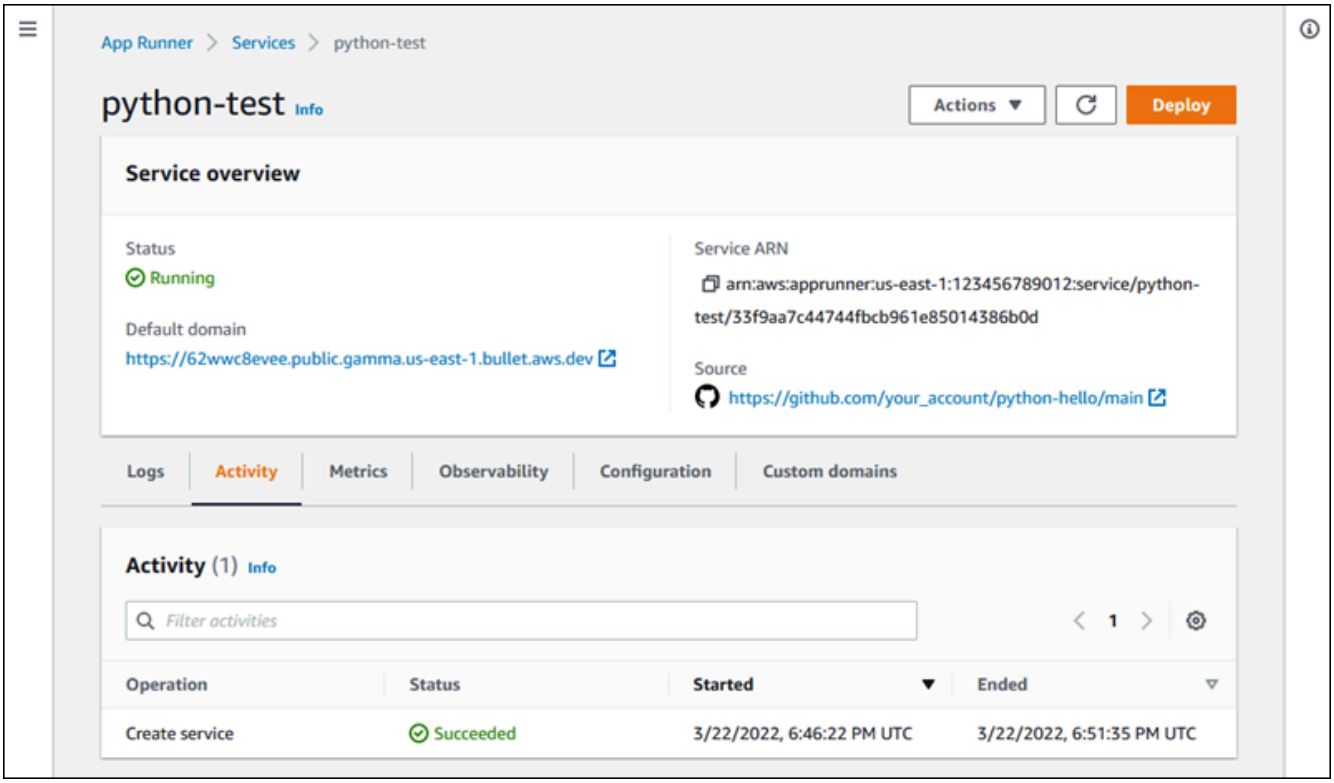
Stellen Sie eine Version Ihrer Anwendung mit einer der folgenden Methoden bereit:

App Runner console

Zur Bereitstellung mit der App Runner-Konsole

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie Bereitstellen.

   Ergebnis: Die Bereitstellung der neuen Version beginnt. Auf der Service-Dashboard-Seite ändert sich der Dienststatus in Vorgang in Bearbeitung.

4. Warten Sie, bis die Bereitstellung beendet ist. Auf der Service-Dashboard-Seite sollte sich der Dienststatus wieder in Wird ausgeführt ändern.

5. Um zu überprüfen, ob die Bereitstellung erfolgreich ist, wählen Sie auf der Service-Dashboard-Seite den Wert Standarddomain aus. Dabei handelt es sich um die URL zur Website Ihres Dienstes. Untersuchen Sie Ihre Webanwendung oder interagieren Sie mit ihr und überprüfen Sie Ihre Versionsänderung.

> **Note**
>
> Um die Sicherheit Ihrer App Runner-Anwendungen zu erhöhen, ist die Domain*.awsapprunner.com in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihre App Runner-Anwendungen einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts,

Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

App Runner API or AWS CLI

Rufen Sie zur Bereitstellung mithilfe der App Runner-API oder AWS CLI die StartDeploymentAPI-Aktion auf. Der einzige Parameter, der übergeben werden muss, ist Ihr Service-ARN. Sie haben den Quellpfad Ihrer Anwendung bereits konfiguriert, als Sie den Dienst erstellt haben, und App Runner kann die neue Version finden. Ihre Bereitstellung beginnt, wenn der Anruf eine erfolgreiche Antwort zurückgibt.

Einen App Runner-Dienst konfigurieren

Wenn Sie einen AWS App Runner Dienst erstellen, legen Sie verschiedene Konfigurationswerte fest. Sie können einige dieser Konfigurationseinstellungen ändern, nachdem Sie den Dienst erstellt haben. Andere Einstellungen können nur bei der Erstellung des Dienstes angewendet werden und können danach nicht mehr geändert werden. In diesem Thema wird die Konfiguration Ihres Dienstes mithilfe der App Runner-API, der App Runner-Konsole und einer App Runner-Konfigurationsdatei beschrieben.

Themen

Konfigurieren Sie Ihren Dienst mithilfe der App Runner API oder AWS CLI

Die API definiert, welche Einstellungen nach der Erstellung des Dienstes geändert werden können. In der folgenden Liste werden die relevanten Aktionen, Typen und Einschränkungen beschrieben.

- UpdateServiceAktion — Kann nach der Erstellung aufgerufen werden, um einige Konfigurationseinstellungen zu aktualisieren.

- Kann aktualisiert werden — Sie können die Einstellungen in den `HealthCheckConfiguration` Parametern `SourceConfigurationInstanceConfiguration`, und aktualisieren. In `SourceConfiguration` können Sie Ihren Quelltyp jedoch nicht von Code zu Bild oder umgekehrt ändern. Sie müssen denselben Repository-Parameter angeben, den Sie bei der Erstellung des Dienstes angegeben haben. Es ist entweder oder. `CodeRepository` `ImageRepository`

  Sie können auch die folgenden ARNs separaten Konfigurationsressourcen aktualisieren, die mit dem Dienst verknüpft sind:

  - `AutoScalingConfigurationArn`
  - `VpcConnectorArn`
- Kann nicht aktualisiert werden — Sie können die `EncryptionConfiguration` Parameter `ServiceName` und, die in der CreateServiceAktion verfügbar sind, nicht ändern. Sie können nicht geändert werden, nachdem sie erstellt wurden. Die UpdateServiceAktion beinhaltet diese Parameter nicht.
- API oder Datei — Sie können den `ConfigurationSource` Parameter des CodeConfigurationTyps (der für Quellcode-Repositorys als Teil von verwendet wird`SourceConfiguration`) auf `Repository` setzen. In diesem Fall ignoriert App Runner die Konfigurationseinstellungen in `CodeConfigurationValues` und liest diese Einstellungen aus einer Konfigurationsdatei in Ihrem Repository. Wenn Sie diese Option festlegen `ConfigurationSourceAPI`, ruft App Runner alle Konfigurationseinstellungen aus dem API-Aufruf ab und ignoriert die Konfigurationsdatei, auch wenn eine vorhanden ist.

- TagResourceaction — Kann aufgerufen werden, nachdem Ihr Service erstellt wurde, um Tags zum Service hinzuzufügen oder Werte vorhandener Tags zu aktualisieren.
- UntagResourceaction — Kann aufgerufen werden, nachdem Ihr Service erstellt wurde, um Tags aus dem Service zu entfernen.

> **Note**
>
> Wenn Sie einen VPC-Connector für ausgehenden Verkehr für einen Dienst erstellen, tritt beim darauffolgenden Dienststartvorgang eine einmalige Latenz auf. Sie können diese Konfiguration für einen neuen Dienst bei der Erstellung oder später mit einem Service-Update

festlegen. Weitere Informationen finden Sie Einmalige Latenz im Kapitel Networking with App Runner in diesem Handbuch.

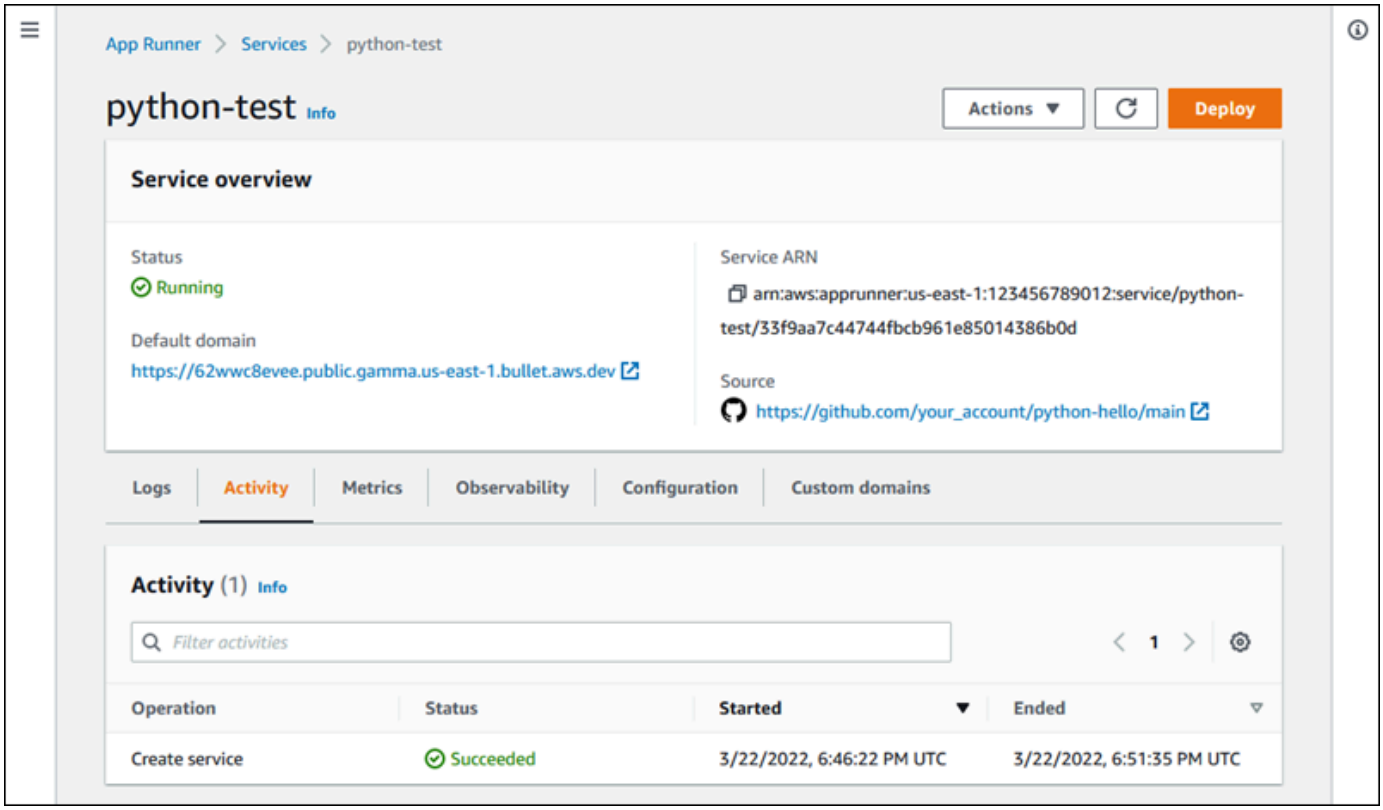
Konfigurieren Sie Ihren Dienst mit der App Runner-Konsole

Die Konsole verwendet die App Runner-API, um Konfigurationsupdates anzuwenden. Die Aktualisierungsregeln, die die API auferlegt, wie im vorherigen Abschnitt definiert, bestimmen, was Sie mit der Konsole konfigurieren können. Einige Einstellungen, die bei der Erstellung des Dienstes verfügbar waren, können später nicht geändert werden. Wenn Sie sich für die Verwendung einer Konfigurationsdatei entscheiden, werden zusätzliche Einstellungen außerdem in der Konsole ausgeblendet und App Runner liest sie aus der Datei.

Um Ihren Dienst zu konfigurieren

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite die Registerkarte Konfiguration aus.

   Ergebnis: Die Konsole zeigt die aktuellen Konfigurationseinstellungen Ihres Dienstes in mehreren Abschnitten an: Quelle und Bereitstellung, Build konfigurieren und Dienst konfigurieren.
4. Um die Einstellungen in einer beliebigen Kategorie zu aktualisieren, wählen Sie Bearbeiten.
5. Nehmen Sie auf der Seite zur Bearbeitung der Konfiguration die gewünschten Änderungen vor und wählen Sie dann Änderungen speichern.

> **Note**
>
> Wenn Sie einen VPC-Connector für ausgehenden Verkehr für einen Dienst erstellen, tritt beim darauffolgenden Dienststartvorgang eine einmalige Latenz auf. Sie können diese Konfiguration für einen neuen Dienst bei der Erstellung oder später mit einem Service-Update festlegen. Weitere Informationen finden Sie Einmalige Latenz im Kapitel Networking with App Runner in diesem Handbuch.

Konfigurieren Sie Ihren Dienst mithilfe einer App Runner-Konfigurationsdatei

Wenn Sie einen App Runner-Dienst erstellen oder aktualisieren, können Sie App Runner anweisen, einige Konfigurationseinstellungen aus einer Konfigurationsdatei zu lesen, die Sie als Teil Ihres Quell-Repositorys bereitstellen. Auf diese Weise können Sie die Einstellungen, die sich auf Ihren Quellcode beziehen, zusammen mit dem Code selbst unter der Quellcodeverwaltung verwalten. Die Konfigurationsdatei enthält auch bestimmte erweiterte Einstellungen, die Sie nicht über die Konsole oder die API festlegen können. Weitere Informationen finden Sie unter App Runner-Konfigurationsdatei.

> **Note**
>
> Wenn Sie einen VPC-Connector für ausgehenden Verkehr für einen Dienst erstellen, tritt beim darauffolgenden Dienststartvorgang eine einmalige Latenz auf. Sie können diese Konfiguration für einen neuen Dienst bei der Erstellung oder später mit einem Service-Update festlegen. Weitere Informationen finden Sie Einmalige Latenz im Kapitel Networking with App Runner in diesem Handbuch.

Observability für Ihren Service konfigurieren

AWS App Runner lässt sich in mehrere AWS Dienste integrieren, um Ihnen eine umfangreiche Suite von Observability-Tools für Ihren App Runner-Service zur Verfügung zu stellen. Weitere Informationen finden Sie unter Beobachtbarkeit.

App Runner unterstützt die Aktivierung einiger Observability-Funktionen und die Konfiguration ihres Verhaltens mithilfe einer gemeinsam nutzbaren Ressource namens. ObservabilityConfiguration Sie können eine Ressource zur Konfiguration von Observability bereitstellen, wenn Sie einen Dienst erstellen oder aktualisieren. Die App Runner-Konsole erstellt eine für Sie, wenn Sie einen neuen App Runner-Dienst erstellen. Die Bereitstellung einer Observability-Konfiguration ist optional. Wenn Sie keine angeben, bietet App Runner eine Standardkonfiguration für Observability.

Sie können eine einzige Observability-Konfiguration für mehrere App Runner-Dienste gemeinsam nutzen, um sicherzustellen, dass sie dasselbe Observability-Verhalten aufweisen. Weitere Informationen finden Sie unter the section called “Ressourcen für die Konfiguration”.

Sie können die folgenden Observability-Funktionen mithilfe von Observability-Konfigurationen konfigurieren:

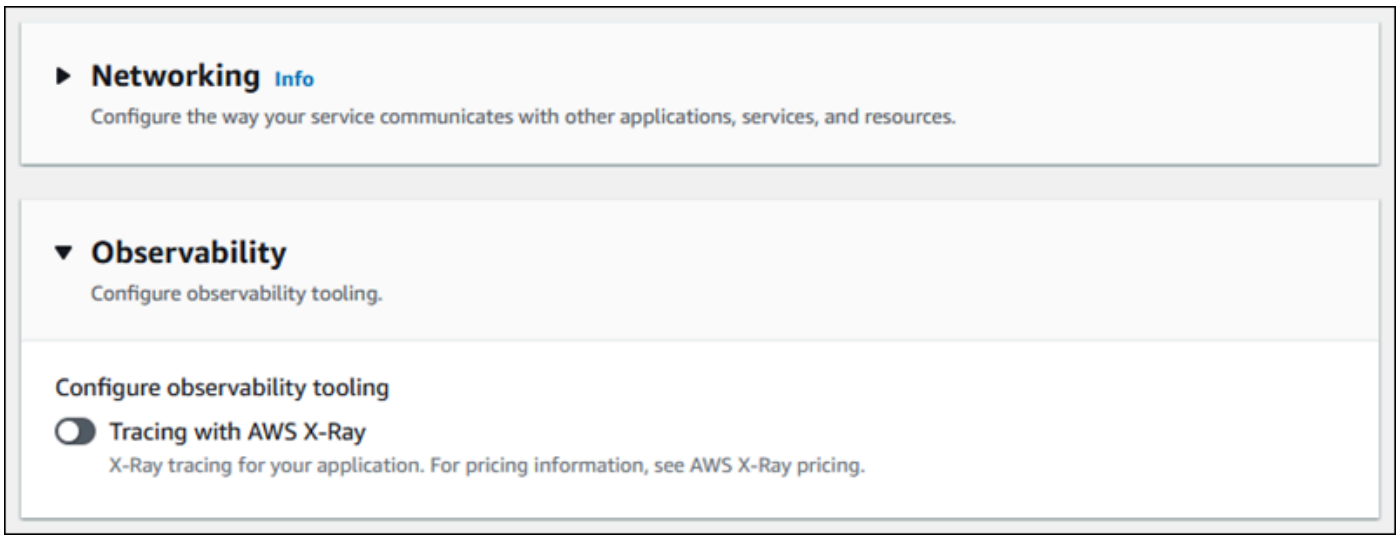
- Trace-Konfiguration — Einstellungen für die Ablaufverfolgung von Anfragen, die Ihre Anwendung bedient, und für die von ihr ausgeführten Downstream-Aufrufe. Weitere Informationen zum Tracing finden Sie unter the section called “Verfolgung (X-Ray)”.

Beobachtbarkeit verwalten

Verwalten Sie die Beobachtbarkeit für Ihre App Runner-Dienste mit einer der folgenden Methoden:

App Runner console

Wenn Sie einen Dienst mit der App Runner-Konsole erstellen oder seine Konfiguration später aktualisieren, können Sie Observability-Funktionen für Ihren Service konfigurieren. Suchen Sie auf der Konsolenseite nach dem Abschnitt zur Observability-Konfiguration.



App Runner API or AWS CLI

Wenn Sie die API-Aktionen CreateServiceoder UpdateServiceApp Runner aufrufen, können Sie das `ObservabilityConfiguration` Parameter-Objekt verwenden, um Observability-Funktionen zu aktivieren und eine Observability-Konfigurationsressource für Ihren Service anzugeben.

Verwenden Sie die folgenden App Runner-API-Aktionen, um Ihre Observability-Konfigurationsressourcen zu verwalten.

- CreateObservabilityConfiguration— Erstellt eine neue Observability-Konfiguration oder eine Revision einer bestehenden.
- ListObservabilityConfigurations— Gibt eine Liste der Observability-Konfigurationen zurück, die mit Ihrer verknüpft sind AWS-Konto, mit zusammenfassenden Informationen.
- DescribeObservabilityConfiguration— Gibt eine vollständige Beschreibung einer Observability-Konfiguration zurück.
- DeleteObservabilityConfiguration— Löscht eine Observability-Konfiguration. Sie können eine bestimmte Revision oder die letzte aktive Revision löschen. Möglicherweise müssen Sie nicht benötigte Observability-Konfigurationen löschen, wenn Sie das Kontingent für die Observability-Konfiguration für Ihre erreichen. AWS-Konto

Konfiguration von Diensteinstellungen mithilfe gemeinsam nutzbarer Ressourcen

Bei einigen Funktionen ist es sinnvoll, die Konfiguration für alle AWS App Runner Dienste gemeinsam zu nutzen. Beispielsweise möchten Sie möglicherweise, dass eine Reihe von Diensten dasselbe Auto Scaling-Verhalten aufweisen. Oder Sie möchten vielleicht identische Observability-Einstellungen für alle Ihre Dienste. Mit App Runner können Sie Einstellungen gemeinsam nutzen, indem Sie separate gemeinsam nutzbare Ressourcen verwenden. Sie erstellen eine Ressource, die eine Reihe von Konfigurationseinstellungen für eine Funktion definiert, und geben dann den Amazon-Ressourcennamen (ARN) dieser Konfigurationsressource an einen oder mehrere App Runner-Dienste weiter.

App Runner implementiert gemeinsam nutzbare Konfigurationsressourcen für die folgenden Funktionen:

- Auto-Scaling
- Beobachtbarkeit
- VPC-Zugriff

Die Dokumentseite für jede dieser Funktionen enthält Informationen zu den verfügbaren Einstellungen und den Verwaltungsverfahren.

Funktionen, die separate Konfigurationsressourcen verwenden, haben einige Konstruktionsmerkmale und Überlegungen gemeinsam.

- Revisionen — Bei einigen Konfigurationsressourcen können Änderungen vorgenommen werden. Auto Scaling und Observability sind Beispiele für zwei Konfigurationsressourcen, die Revisionen verwenden. In diesen Fällen hat jede Konfiguration einen Namen und eine numerische Revision. Mehrere Versionen einer Konfiguration haben denselben Namen und unterschiedliche Revisionsnummern. Sie können unterschiedliche Konfigurationsnamen für verschiedene Szenarien verwenden. Für jeden Namen können Sie mehrere Versionen hinzufügen, um die Einstellungen für ein bestimmtes Szenario zu optimieren.

  Die erste Konfiguration, die Sie mit einem Namen erstellen, erhält die Revisionsnummer 1. Nachfolgende Konfigurationen mit demselben Namen erhalten fortlaufende Revisionsnummern (beginnend mit 2). Sie können Ihren App Runner-Dienst einer bestimmten Konfigurationsrevision oder der neuesten Version der Konfiguration zuordnen.

- Gemeinsam genutzt — Sie können eine einzelne Konfigurationsressource für mehrere App Runner-Dienste gemeinsam nutzen. Dies ist nützlich, wenn Sie identische Konfigurationen für diese Dienste beibehalten möchten. Insbesondere, wenn Ihre Ressourcen Revisionen unterstützen, können Sie mehrere Dienste so konfigurieren, dass sie die neueste Version einer Konfiguration verwenden. Sie können dies tun, indem Sie nur den Namen der Konfiguration angeben, aber keine Revision. Jeder der Dienste, die Sie auf diese Weise konfiguriert haben, erhält Konfigurationsupdates, wenn Sie den Dienst aktualisieren. Weitere Informationen zu Konfigurationsänderungen finden Sie unterthe section called “Konfiguration”.
- Ressourcenverwaltung — Sie können App Runner verwenden, um Konfigurationen zu erstellen und zu löschen. Sie können eine Konfiguration nicht direkt aktualisieren. Stattdessen können Sie für Ressourcen, die Revisionen unterstützen, eine neue Version eines vorhandenen Konfigurationsnamens erstellen, um die Konfiguration effektiv zu aktualisieren.

  > **Note**
  >
  > Für die auto Skalierung können Sie Konfigurationen und mehrere Revisionen sowohl mit der App Runner-Konsole als auch mit der App Runner-API erstellen. Sowohl die App Runner-Konsole als auch die App Runner-API können auch Konfigurationen und Revisionen löschen. Weitere Details finden Sie unter Verwaltung der automatischen Skalierung von App Runner.
  >
  > Für andere Konfigurationstypen, wie Observability-Konfigurationen, können Sie mit der App Runner-Konsole nur eine Konfiguration mit einer einzigen Revision erstellen. Um weitere Revisionen zu erstellen und Konfigurationen zu löschen, müssen Sie die App Runner-API verwenden.

- Ressourcenkontingent — Es gibt festgelegte Kontingente für die Anzahl der eindeutigen Konfigurationsnamen und Revisionen, die Sie jeweils für Ihre Konfigurationsressourcen haben können. AWS-Region Wenn Sie diese Kontingente erreichen, müssen Sie entweder einen Konfigurationsnamen oder zumindest einige seiner Versionen löschen, bevor Sie weitere erstellen können. Für Revisionen von Auto Scaling-Konfigurationen können Sie die App Runner-Konsole oder die App Runner-API verwenden, um sie zu löschen. Weitere Details finden Sie unter Verwaltung der automatischen Skalierung von App Runner. Sie müssen die App Runner-API verwenden, um andere Ressourcen zu löschen. Weitere Informationen zu Kontingenten finden Sie unter the section called “App Runner-Ressourcenkontingente”.
- Keine Ressourcenkosten — Es fallen keine zusätzlichen Kosten für die Erstellung einer Konfigurationsressource an. Möglicherweise fallen Kosten für die Funktion selbst an (wenn Sie X-Ray Tracing aktivieren, werden Ihnen beispielsweise die normalen AWS X-Ray Kosten in

Rechnung gestellt), aber nicht für die App Runner-Konfigurationsressource, die die Funktion für Ihren App Runner-Dienst konfiguriert.

Konfiguration von Zustandsprüfungen für Ihren Dienst

AWS App Runner überwacht den Zustand Ihres Dienstes, indem es Integritätsprüfungen durchführt. Das Standardprotokoll für die Integritätsprüfung ist TCP. App Runner pingt die Ihrem Dienst zugewiesene Domain an. Sie können das Health Check-Protokoll alternativ auf HTTP setzen. App Runner sendet HTTP-Anfragen zur Integritätsprüfung an Ihre Webanwendung.

Sie können einige Einstellungen für Integritätsprüfungen konfigurieren. In der folgenden Tabelle werden die Einstellungen für die Integritätsprüfung und ihre Standardwerte beschrieben.

Einstellung	Beschreibung	Standard
Protokoll	Das IP-Protokoll, das App Runner zur Durchführung von Zustandsprüfungen für Ihren Service verwendet. Wenn Sie das Protokoll auf einstellen`TCP`, pingt App Runner die Ihrem Dienst zugewiesene Standarddomäne an den Port, den Ihre Anwendung abhört. Wenn Sie das Protokoll auf einstellen`HTTP`, sendet App Runner Integritätsprüfungsanfragen an den konfigurierten Pfad.	`TCP`
Pfad	Die URL, an die App Runner HTTP-Integritätsprüfungsanfragen sendet. Gilt nur für HTTP-Prüfungen.	`/`
Intervall	Das Intervall in Sekunden zwischen den Zustandsprüfungen.	5
Zeitüberschreitung	Die Zeit in Sekunden, die auf eine Antwort auf eine Zustandsprüfung gewartet wird, bevor diese als fehlgeschlagen gewertet wird.	2
Gesunder Schwellenwert	Die Anzahl der aufeinanderfolgenden Prüfungen, die erfolgreich sein müssen, bevor App Runner entscheidet, dass der Service einwandfrei ist.	1

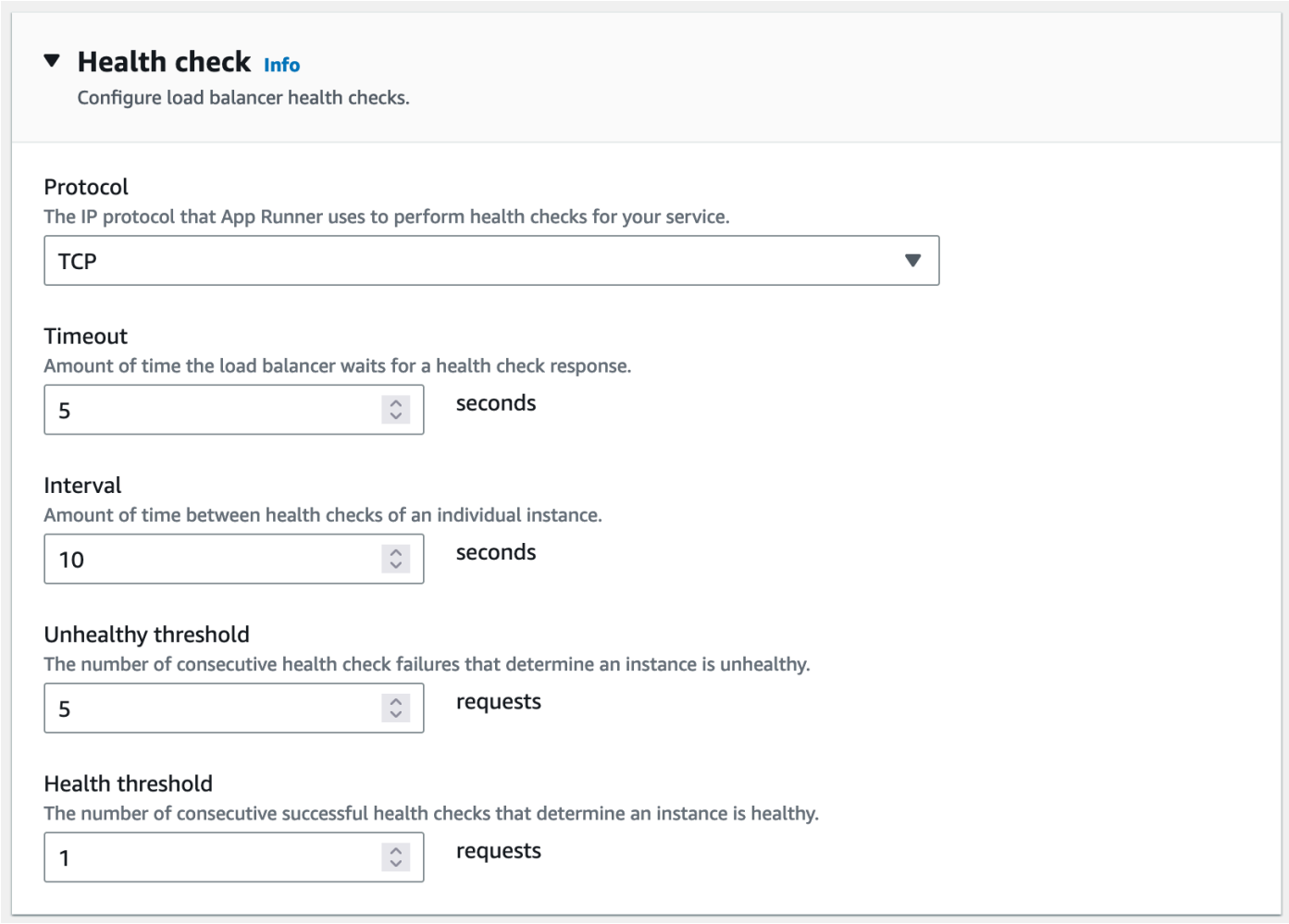
Einstellung	Beschreibung	Standard
Ungesunder Schwellenwert	Die Anzahl der aufeinanderfolgenden Prüfungen, die fehlschlagen müssen, bevor App Runner entscheidet, dass der Service-Zustand nicht ordnungsgemäß ist.	5

Konfigurieren von Zustandsprüfungen

Konfigurieren Sie Integritätsprüfungen für Ihren App Runner-Dienst mit einer der folgenden Methoden:

App Runner console

Wenn Sie Ihren App Runner-Dienst mit der App Runner-Konsole erstellen oder wenn Sie seine Konfiguration später aktualisieren, können Sie die Einstellungen für die Integritätsprüfung konfigurieren. Vollständige Konsolenprozeduren finden Sie unter the section called “Erstellung” undthe section called “Konfiguration”. Suchen Sie in beiden Fällen auf der Konsolenseite nach dem Abschnitt Health Check-Konfiguration.



App Runner API or AWS CLI

Wenn Sie die CreateServiceoder UpdateServiceAPI-Aktionen aufrufen, können Sie den `HealthCheckConfiguration` Parameter verwenden, um Einstellungen für die Integritätsprüfung festzulegen.

Informationen zur Struktur des Parameters finden Sie HealthCheckConfigurationin der AWS App Runner API-Referenz.

App Runner-Verbindungen verwalten

Wenn Sie einen Service in erstellen AWS App Runner, konfigurieren Sie eine Anwendungsquelle — ein Container-Image oder ein Quell-Repository, das bei einem Anbieter gespeichert ist. App Runner muss eine authentifizierte und autorisierte Verbindung mit dem Anbieter herstellen. Dann kann App Runner Ihr Repository lesen und es für Ihren Dienst bereitstellen. App Runner erfordert keinen

Verbindungsaufbau, wenn Sie einen Dienst erstellen, der auf den in Ihrem AWS-Konto gespeicherten Code zugreift.

App Runner verwaltet Verbindungsinformationen in einer Ressource, die als Verbindung bezeichnet wird. In der App Runner-Konsole und in diesem Handbuch werden Verbindungen auch als verbundene Konten bezeichnet. App Runner benötigt eine Verbindungsressource, wenn Sie einen Dienst erstellen, der Verbindungsinformationen von Drittanbietern benötigt. Im Folgenden finden Sie einige wichtige Informationen zu Verbindungen:

- Anbieter — App Runner benötigt derzeit Verbindungsressourcen mit GitHuboder Bitbucket.
- Geteilt — Du kannst eine Verbindungsressource verwenden, um mehrere App Runner-Dienste zu erstellen, die dasselbe Repository-Anbieter-Konto verwenden.
- Ressourcenverwaltung — In App Runner können Sie Verbindungen erstellen und löschen. Sie können eine bestehende Verbindung jedoch nicht ändern.
- Ressourcenkontingent — Verbindungsressourcen haben ein festgelegtes Kontingent, das AWS-Konto jeweils Ihrem entspricht AWS-Region. Wenn Sie dieses Kontingent erreichen, müssen Sie möglicherweise eine Verbindung löschen, bevor Sie eine Verbindung zu einem neuen Anbieterkonto herstellen können. Sie können eine Verbindung mithilfe der App Runner-Konsole oder der API löschen, wie im folgenden Abschnitt beschriebenthe section called “Verbindungen verwalten”. Weitere Informationen finden Sie unter the section called “App Runner-Ressourcenkontingente”.

Verbindungen verwalten

Verwalte deine App Runner-Verbindungen mit einer der folgenden Methoden:

App Runner console

Wenn Sie die App Runner-Konsole verwenden, um einen Dienst zu erstellen, geben Sie Verbindungsdetails an. Sie müssen nicht explizit eine Verbindungsressource erstellen. In der Konsole kannst du wählen, ob du dich mit einem GitHub oder einem Bitbucket-Konto, mit dem du zuvor verbunden warst, oder mit einem neuen Konto verbinden möchtest. Bei Bedarf erstellt App Runner eine Verbindungsressource für dich. Für eine neue Verbindung verlangen einige Anbieter, dass Sie einen Authentifizierungs-Handshake durchführen, bevor Sie die Verbindung verwenden können. Die Konsole führt Sie durch diesen Vorgang.

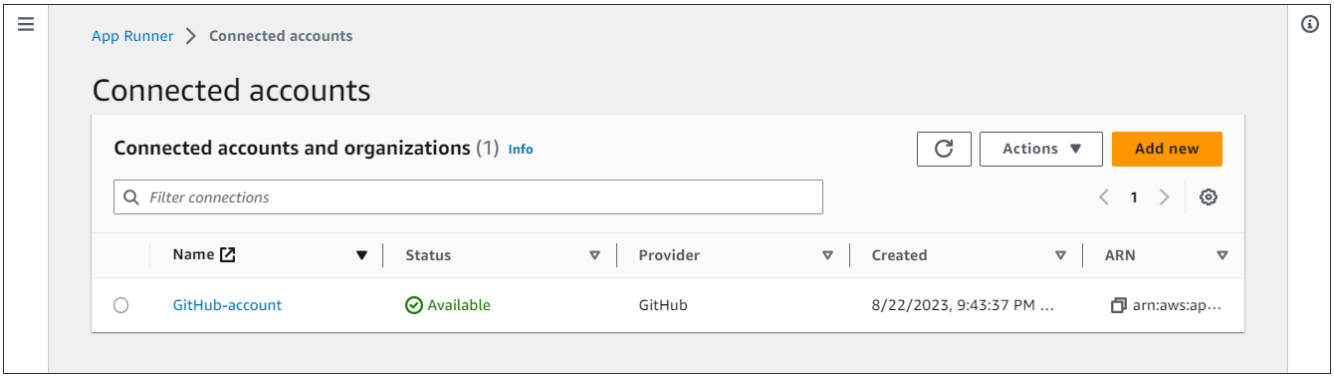
Die Konsole verfügt auch über eine Seite zur Verwaltung Ihrer bestehenden Verbindungen. Sie können den Authentifizierungs-Handshake für eine Verbindung abschließen, wenn Sie ihn bei

der Erstellung Ihres Dienstes nicht getan haben. Sie können auch Verbindungen löschen, die Sie nicht mehr verwenden. Das folgende Verfahren zeigt, wie Sie Verbindungen mit Repository-Anbietern verwalten können.

Um Verbindungen in Ihrem Konto zu verwalten

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Verbundene Konten aus.

   In der Konsole wird dann eine Liste der Repository-Provider-Verbindungen in Ihrem Konto angezeigt.



3. Sie können jetzt mit jeder Verbindung in der Liste eine der folgenden Aktionen ausführen:

   - GitHub/Bitbucket Konto oder Organisation eröffnen — Wählen Sie den Namen der Verbindung.
   - Vollständiger Authentifizierungs-Handshake — Wählen Sie die Verbindung aus und wählen Sie dann im Menü Aktionen die Option Vollständiger Handshake aus. Die Konsole führt Sie durch den Authentifizierungs-Handshake-Prozess.
   - Verbindung löschen — Wählen Sie die Verbindung aus und wählen Sie dann im Menü Aktionen die Option Löschen aus. Folgen Sie den Anweisungen auf der Löschaufforderung.

App Runner API or AWS CLI

Sie können die folgenden App Runner API-Aktionen verwenden, um Ihre Verbindungen zu verwalten.

- CreateConnection— Stellt eine Verbindung zu einem Repository-Anbieter-Konto her. Nachdem die Verbindung hergestellt wurde, müssen Sie den Authentifizierungs-Handshake manuell

mithilfe der App Runner-Konsole abschließen. Dieser Vorgang wurde im vorherigen Abschnitt erklärt.

- ListConnections— Gibt eine Liste der App Runner-Verbindungen zurück, die mit Ihrem verknüpft sind AWS-Konto.
- DeleteConnection— Löscht eine Verbindung. Möglicherweise müssen Sie nicht benötigte Verbindungen löschen, wenn Sie das Verbindungskontingent für Ihre AWS-Konto erreichen.

Verwaltung der automatischen Skalierung von App Runner

AWS App Runner skaliert Rechenressourcen, insbesondere Instanzen, für Ihre App Runner-Anwendung automatisch nach oben oder unten. Die automatische Skalierung sorgt für eine angemessene Bearbeitung von Anfragen bei hohem Datenverkehr und reduziert Ihre Kosten, wenn sich der Verkehr verlangsamt.

Konfiguration für automatische Skalierung

Sie können einige Parameter konfigurieren, um das Auto-Scaling-Verhalten für Ihren Service anzupassen. App Runner verwaltet Auto-Scaling-Einstellungen in einer gemeinsam nutzbaren Ressource, die aufgerufen wird AutoScalingConfiguration. Sie können eigenständige Auto Scaling-Konfigurationen erstellen und verwalten, bevor Sie sie Diensten zuweisen. Nachdem sie einem Dienst zugeordnet wurden, können Sie die Konfigurationen weiterhin verwalten. Sie können sich auch dafür entscheiden, eine neue Auto Scaling-Konfiguration zu erstellen, während Sie gerade dabei sind, einen neuen Service zu erstellen oder einen vorhandenen zu konfigurieren. Sobald die neue Auto Scaling-Konfiguration erstellt wurde, können Sie sie dem Service zuordnen und mit dem Erstellen oder Konfigurieren Ihres Dienstes fortfahren.

Benennung und Überarbeitungen

Eine Auto Scaling-Konfiguration hat einen Namen und eine numerische Revision. Mehrere Revisionen einer Konfiguration haben denselben Namen und unterschiedliche Revisionsnummern. Sie können unterschiedliche Konfigurationsnamen für verschiedene Auto Scaling-Szenarien verwenden, z. B. hohe Verfügbarkeit oder niedrige Kosten. Für jeden Namen können Sie mehrere Versionen hinzufügen, um die Einstellungen für ein bestimmtes Szenario zu optimieren. Sie können bis zu 10 eindeutige Auto Scaling-Konfigurationsnamen und bis zu 5 Revisionen für jede Konfiguration verwenden. Wenn Sie das Limit erreichen und weitere erstellen müssen, können Sie eine löschen und dann eine weitere erstellen. App Runner erlaubt es Ihnen nicht, eine Konfiguration zu löschen, die als Standard festgelegt ist oder von einem aktiven Dienst verwendet

wird. Weitere Informationen zu Kontingenten finden Sie unter the section called “App Runner-Ressourcenkontingente”.

Eine Standardkonfiguration festlegen

Wenn Sie einen App Runner-Dienst erstellen oder aktualisieren, können Sie eine Auto Scaling-Konfigurationsressource bereitstellen. Die Bereitstellung einer Auto Scaling-Konfiguration ist optional. Wenn Sie keine angeben, bietet App Runner eine standardmäßige Auto-Scaling-Konfiguration mit empfohlenen Werten. Die Auto Scaling-Konfigurationsfunktion bietet Ihnen die Möglichkeit, Ihre eigene Standardkonfiguration für Auto Scaling festzulegen, anstatt die von App Runner bereitgestellte Standardkonfiguration zu verwenden. Sobald Sie eine andere Auto Scaling-Konfiguration als Standard angeben, wird diese Konfiguration den neuen Diensten, die Sie in future erstellen, automatisch als Standard zugewiesen. Die neue Standardbezeichnung wirkt sich nicht auf die Zuordnungen aus, die zuvor für bestehende Dienste festgelegt wurden.

Konfiguration von Diensten mit Auto Scaling

Sie können eine einzige Auto Scaling-Konfiguration für mehrere App Runner-Dienste gemeinsam nutzen, um sicherzustellen, dass die Dienste dasselbe Auto Scaling-Verhalten aufweisen. Weitere Informationen zur Konfiguration von Auto Scaling-Konfigurationen mit der App Runner-Konsole oder der App Runner-API finden Sie in den folgenden Abschnitten dieses Themas. Weitere allgemeine Informationen zu gemeinsam nutzbaren Ressourcen finden Sie unterthe section called “Ressourcen für die Konfiguration”.

Konfigurierbare Einstellungen

Sie können die folgenden Auto Scaling-Einstellungen konfigurieren:

- Max. Parallelität — Die maximale Anzahl gleichzeitiger Anfragen, die eine Instance verarbeitet. Wenn die Anzahl gleichzeitiger Anfragen dieses Kontingent überschreitet, skaliert App Runner den Dienst.
- Maximale Größe — Die maximale Anzahl von Instanzen, auf die Ihr Service skaliert werden kann. Dies ist die höchste Anzahl von Instanzen, die den Datenverkehr Ihres Dienstes gleichzeitig verarbeiten können.
- Mindestgröße — Die Mindestanzahl von Instanzen, die App Runner für Ihren Dienst bereitstellen kann. Der Dienst hat immer mindestens diese Anzahl von bereitgestellten Instanzen. Einige dieser Instanzen verarbeiten aktiv den Datenverkehr. Die übrigen sind Teil der kostengünstigen Rechenkapazitätsreserve, die schnell aktiviert werden kann. Sie zahlen für die Speichernutzung aller bereitgestellten Instanzen. Sie zahlen nur für die CPU-Auslastung der aktiven Teilmenge.

> **Note**
>
> Die Anzahl der vCPU-Ressourcen bestimmt die Anzahl der Instanzen, die App Runner für Ihren Dienst bereitstellen kann. Dies ist ein einstellbarer Kontingentwert für die Fargate On-Demand-vCPU-Ressourcenanzahl, die sich im Service befindet. AWS Fargate Um die vCPU-Kontingenteinstellungen für Ihr Konto einzusehen oder eine Erhöhung des Kontingents zu beantragen, verwenden Sie die Konsole Service Quotas in der AWS Management Console. Weitere Informationen finden Sie unter AWS Fargate Service-Kontingente im Amazon Elastic Container Service Developer Guide.

Auto Scaling für einen Service verwalten

Verwalten Sie die auto Skalierung für Ihre App Runner-Dienste mit einer der folgenden Methoden:

App Runner console

Wenn Sie mit der App Runner-Konsole einen Dienst erstellen oder eine Dienstkonfiguration aktualisieren, können Sie eine Auto Scaling-Konfiguration angeben.

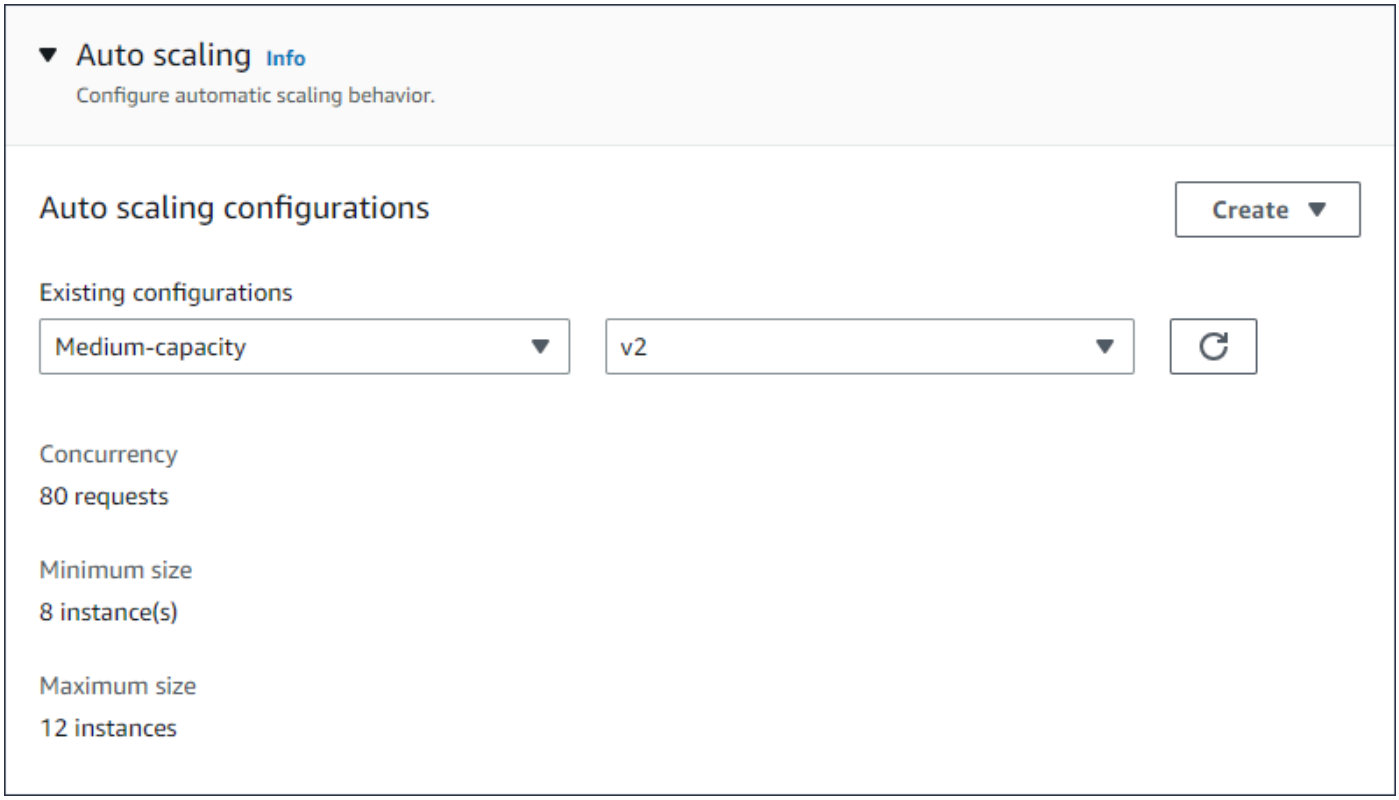
> **Note**
>
> Wenn Sie die Auto Scaling-Konfiguration oder Revision ändern, die einem Service zugeordnet ist, wird Ihr Service erneut bereitgestellt.

Die Auto Scaling-Konfigurationsseite bietet mehrere Optionen zur Konfiguration von Auto Scaling für Ihren Service.

- Um eine bestehende Konfiguration und Revision zuzuweisen — Wählen Sie einen Wert aus der Dropdownliste Bestehende Konfigurationen aus. Die neueste Revisionsversion wird standardmäßig in der nebenstehenden Drop-down-Liste angezeigt. Wenn es eine andere Version gibt, die Sie lieber auswählen möchten, wählen Sie diese im Drop-down-Menü für die Revision aus. Die Konfigurationswerte für die Revisionsversion werden angezeigt.
- Um eine neue Auto Scaling-Konfiguration zu erstellen und zuzuweisen, wählen Sie im Menü Erstellen die Option Neues ASC erstellen aus. Dadurch wird die Seite Benutzerdefinierte Auto Scaling-Konfiguration hinzufügen geöffnet. Geben Sie einen Konfigurationsnamen und Werte für die Auto Scaling-Parameter ein. Wählen Sie dann Hinzufügen aus. App Runner erstellt die

neue Auto Scaling-Konfigurationsressource für Sie und bringt Sie zurück zum Bereich Auto Scaling, wobei die neue Konfiguration ausgewählt und angezeigt wird.

- Um eine neue Revision zu erstellen und zuzuweisen — Wählen Sie zunächst den Konfigurationsnamen aus der Drop-down-Liste „Vorhandene Konfigurationen“ aus. Wählen Sie dann im Menü Erstellen die Option ASC-Revision erstellen aus. Dadurch wird die Seite Benutzerdefinierte Auto Scaling-Konfiguration hinzufügen geöffnet. Geben Sie Werte für die Auto Scaling-Parameter ein. Wählen Sie dann Hinzufügen aus. App Runner erstellt eine neue Version der Auto Scaling-Konfiguration für Sie und kehrt zum Auto Scaling-Bereich zurück, wobei die neue Version ausgewählt und angezeigt wird.



App Runner API or AWS CLI

Wenn Sie die API-Aktionen CreateServiceoder UpdateServiceApp Runner aufrufen, können Sie den `AutoScalingConfigurationArn` Parameter verwenden, um eine Auto Scaling-Konfigurationsressource für Ihren Service anzugeben.

Der nächste Abschnitt enthält Anleitungen zur Verwaltung Ihrer Auto Scaling-Konfigurationsressourcen.

Ressourcen für Auto Scaling-Konfigurationen verwalten

Verwalte die Auto Scaling-Konfigurationen und Revisionen von App Runner für dein Konto mit einer der folgenden Methoden:

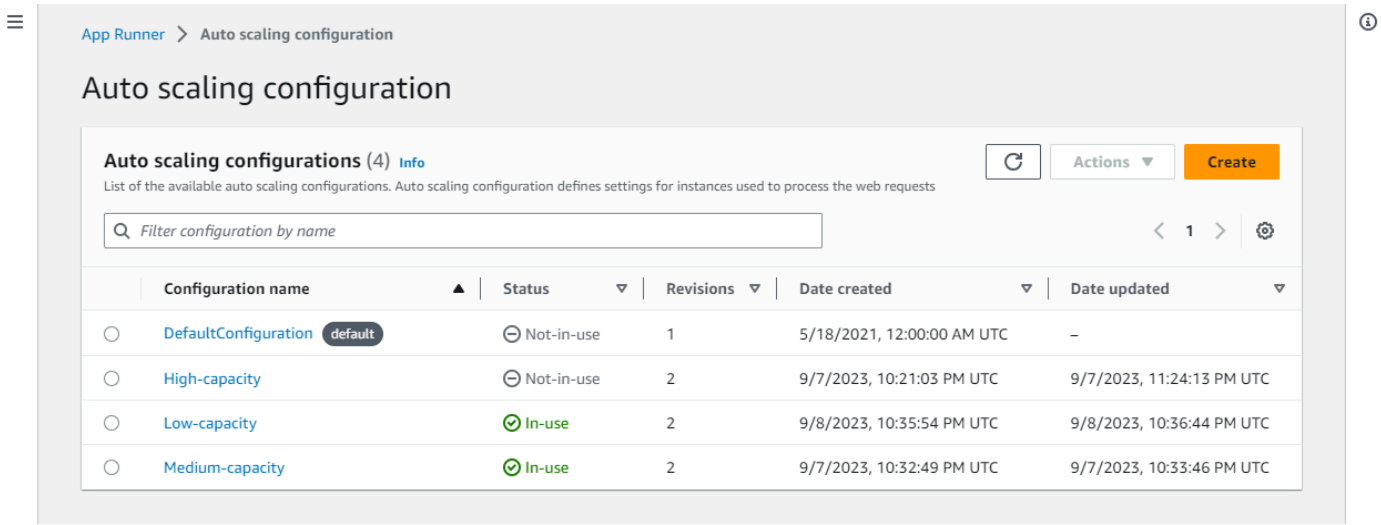
App Runner console

Auto Scaling-Konfigurationen verwalten

Auf der Seite mit den Auto Scaling-Konfigurationen sind die Auto Scaling-Konfigurationen aufgeführt, die Sie in Ihrem Konto eingerichtet haben. Sie können Ihre Auto Scaling-Konfigurationen auf dieser Seite erstellen und verwalten und sie später einem oder mehreren App Runner-Diensten zuweisen.

Auf dieser Seite können Sie jeden der folgenden Schritte ausführen:

- Erstellen Sie eine neue Auto Scaling-Konfiguration.
- Erstellen Sie eine neue Revision für eine bestehende Auto Scaling-Konfiguration.
- Löschen Sie eine Auto Scaling-Konfiguration.
- Stellen Sie eine Auto Scaling-Konfiguration als Standard ein.



Um Auto Scaling-Konfigurationen in Ihrem Konto zu verwalten

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.

2. Wählen Sie im Navigationsbereich Auto Scaling-Konfigurationen aus. Die Konsole zeigt eine Liste der Auto Scaling-Konfigurationen in Ihrem Konto an.

   Sie können jetzt eine der folgenden Aktionen ausführen.

   - Gehen Sie folgendermaßen vor, um eine neue Auto Scaling-Konfiguration zu erstellen.

     a. Wählen Sie auf der Seite mit den Auto Scaling-Konfigurationen die Option Erstellen aus.

        Die Seite Auto Scaling-Konfiguration erstellen wird angezeigt.

     b. Geben Sie Werte für Konfigurationsname, Parallelität, Mindestgröße und Maximalgröße ein.

     c. (Optional) Wenn Sie Tags hinzufügen möchten, wählen Sie Automatisch neues Tag aus. Geben Sie dann in den angezeigten Feldern einen Namen und einen Wert ein (optional).

     d. Wählen Sie Erstellen aus.

   - Gehen Sie folgendermaßen vor, um eine neue Version für eine bestehende Auto Scaling-Konfiguration zu erstellen.

     a. Wählen Sie auf der Seite mit den Auto Scaling-Konfigurationen das Optionsfeld neben der Konfiguration aus, für die die neue Version erforderlich ist. Wählen Sie dann im Menü Aktionen die Option Revision erstellen aus.

        Die Seite „Revision erstellen“ wird angezeigt.

     b. Geben Sie bei Aktivierung Werte für Parallelität, Mindestgröße und Maximalgröße ein.

     c. (Optional) Wenn Sie Tags hinzufügen möchten, wählen Sie Automatisch neues Tag aus. Geben Sie dann in den angezeigten Feldern einen Namen und einen Wert ein (optional).

     d. Wählen Sie Erstellen aus.

   - Gehen Sie folgendermaßen vor, um eine Auto Scaling-Konfiguration zu löschen.

     a. Wählen Sie auf der Seite mit den Auto Scaling-Konfigurationen das Optionsfeld neben der Konfiguration aus, die Sie löschen möchten.

     b. Wählen Sie im Menü Aktionen die Option Löschen aus.

c. Um mit dem Löschen fortzufahren, wählen Sie im Bestätigungsdialogfeld die Option Löschen aus. Wählen Sie andernfalls Abbrechen aus.

> **Note**
>
> App Runner überprüft, ob Ihre Löschoption nicht als Standard festgelegt ist oder derzeit von aktiven Diensten verwendet wird.

- Gehen Sie wie folgt vor, um eine Auto Scaling-Konfiguration als Standard festzulegen.

a. Wählen Sie auf der Seite mit den Auto Scaling-Konfigurationen das Optionsfeld neben der Konfiguration aus, die Sie als Standard festlegen müssen.

b. Wählen Sie im Menü Aktionen die Option Als Standard festlegen aus.

c. Es wird ein Dialogfeld angezeigt, in dem Sie darüber informiert werden, dass App Runner die neueste Version als Standardkonfiguration für alle neuen Dienste verwendet, die Sie erstellen. Wählen Sie Bestätigen, um fortzufahren. Wählen Sie andernfalls Abbrechen aus.

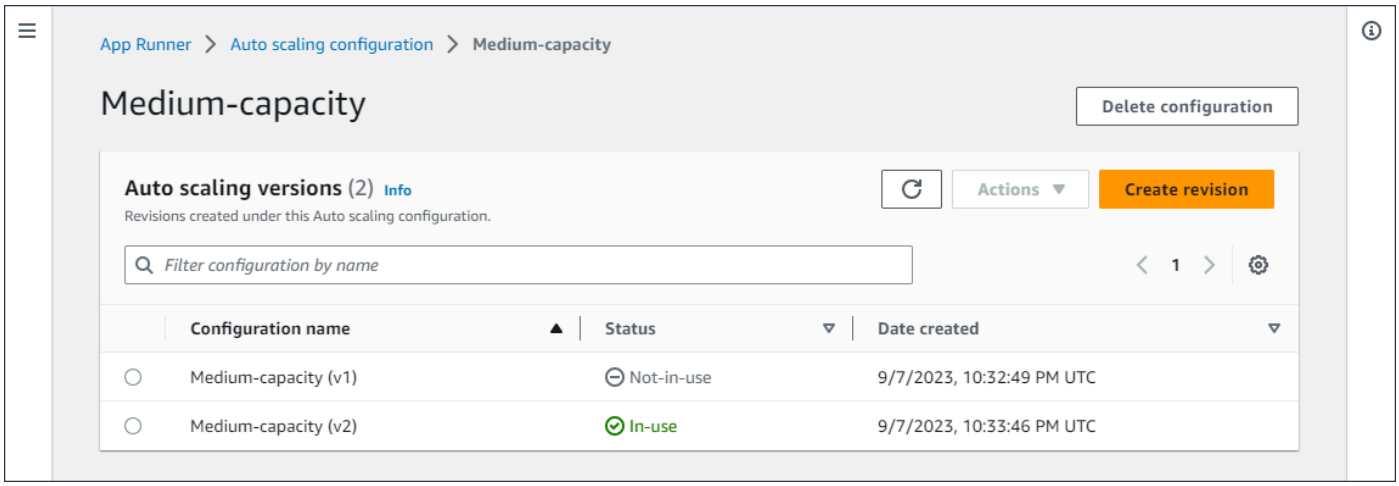
> **Note**
>
> - Wenn Sie eine Auto Scaling-Konfiguration als Standard festlegen, wird sie den neuen Diensten, die Sie in future erstellen, automatisch als Standardkonfiguration zugewiesen.
> - Die neue Standardbezeichnung wirkt sich nicht auf die Zuordnungen aus, die zuvor für bestehende Dienste festgelegt wurden.
> - Wenn die angegebene Standardkonfiguration für Auto Scaling Revisionen enthält, weist App Runner die neueste Version als Standard zu.

Revisionen verwalten

Die Konsole verfügt auch über eine Seite zum Erstellen und Verwalten Ihrer vorhandenen Auto Scaling-Revisionen, die als Auto Scaling-Revisionen bezeichnet werden. Rufen Sie diese Seite auf, indem Sie auf der Seite mit den Auto Scaling-Konfigurationen den Namen einer Konfiguration auswählen.

Auf der Seite mit den Auto Scaling-Revisionen können Sie eine der folgenden Aktionen ausführen:

- Erstellen Sie eine neue Auto Scaling-Revision.
- Legen Sie eine Revision der Auto Scaling-Konfiguration als Standard fest.
- Löscht eine Revision.
- Löschen Sie die gesamte Auto Scaling-Konfiguration, einschließlich aller zugehörigen Revisionen.
- Sehen Sie sich die Konfigurationsdetails für eine Revision an.
- Eine Liste der mit einer Revision verknüpften Dienste anzeigen.
- Ändern Sie die Revision für einen aufgelisteten Dienst.



Um Auto Scaling-Revisionen in Ihrem Konto zu verwalten

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Wählen Sie im Navigationsbereich Auto Scaling-Konfigurationen aus. Die Konsole zeigt eine Liste der Auto Scaling-Konfigurationen in Ihrem Konto an. Die vorherigen Verfahren in diesem Auto Scaling-Konfigurationen verwalten Abschnitt enthalten ein Bildschirmbild dieser Seite.
3. Jetzt können Sie eine bestimmte Auto Scaling-Konfiguration detailliert untersuchen, um alle Änderungen anzuzeigen und zu verwalten. Wählen Sie im Bereich Auto Scaling-Konfigurationen in der Spalte Konfigurationsname einen Namen für die Auto Scaling-Konfiguration aus. Wählen Sie den tatsächlichen Namen und nicht das Optionsfeld aus. Dadurch gelangen Sie zu einer Liste aller Revisionen für diese Konfiguration auf der Seite mit den Auto Scaling-Revisionen.
4. Sie können jetzt eine der folgenden Aktionen ausführen.

- Gehen Sie folgendermaßen vor, um eine neue Version für eine bestehende Auto Scaling-Konfiguration zu erstellen.

  a. Wählen Sie auf der Seite mit Auto Scaling-Revisionen die Option Revision erstellen aus.

     Die Seite Revision erstellen wird angezeigt.

  b. Geben Sie Werte für Parallelität, Mindestgröße und Maximalgröße ein.

  c. (Optional) Wenn Sie Tags hinzufügen möchten, wählen Sie Automatisch neues Tag aus. Geben Sie dann in den angezeigten Feldern einen Namen und einen Wert ein (optional).

  d. Wählen Sie Erstellen aus.

- Gehen Sie wie folgt vor, um die gesamte Auto Scaling-Konfiguration einschließlich aller zugehörigen Revisionen zu löschen.

  a. Wählen Sie oben rechts auf der Seite Konfiguration löschen aus.

  b. Um mit dem Löschen fortzufahren, wählen Sie im Bestätigungsdialogfeld Löschen aus. Wählen Sie andernfalls Abbrechen aus.

  > **Note**
  >
  > App Runner überprüft, ob Ihre Löschoption nicht als Standard festgelegt ist oder derzeit von aktiven Diensten verwendet wird.

- Gehen Sie wie folgt vor, um eine Auto Scaling-Revision als Standard festzulegen.

  a. Wählen Sie das Optionsfeld neben der Revision aus, die Sie als Standard festlegen müssen.

  b. Wählen Sie im Menü Aktionen die Option Als Standard festlegen aus.

  > **Note**
  >
  > - Wenn Sie eine Auto Scaling-Konfiguration als Standard festlegen, wird sie den neuen Diensten, die Sie in future erstellen, automatisch als Standardkonfiguration zugewiesen.

- Die neue Standardbezeichnung wirkt sich nicht auf die Zuordnungen aus, die zuvor für bestehende Dienste festgelegt wurden.

- Gehen Sie wie folgt vor, um die Konfigurationsdetails für eine Version anzuzeigen.
  - Wählen Sie das Optionsfeld neben der Revision aus.

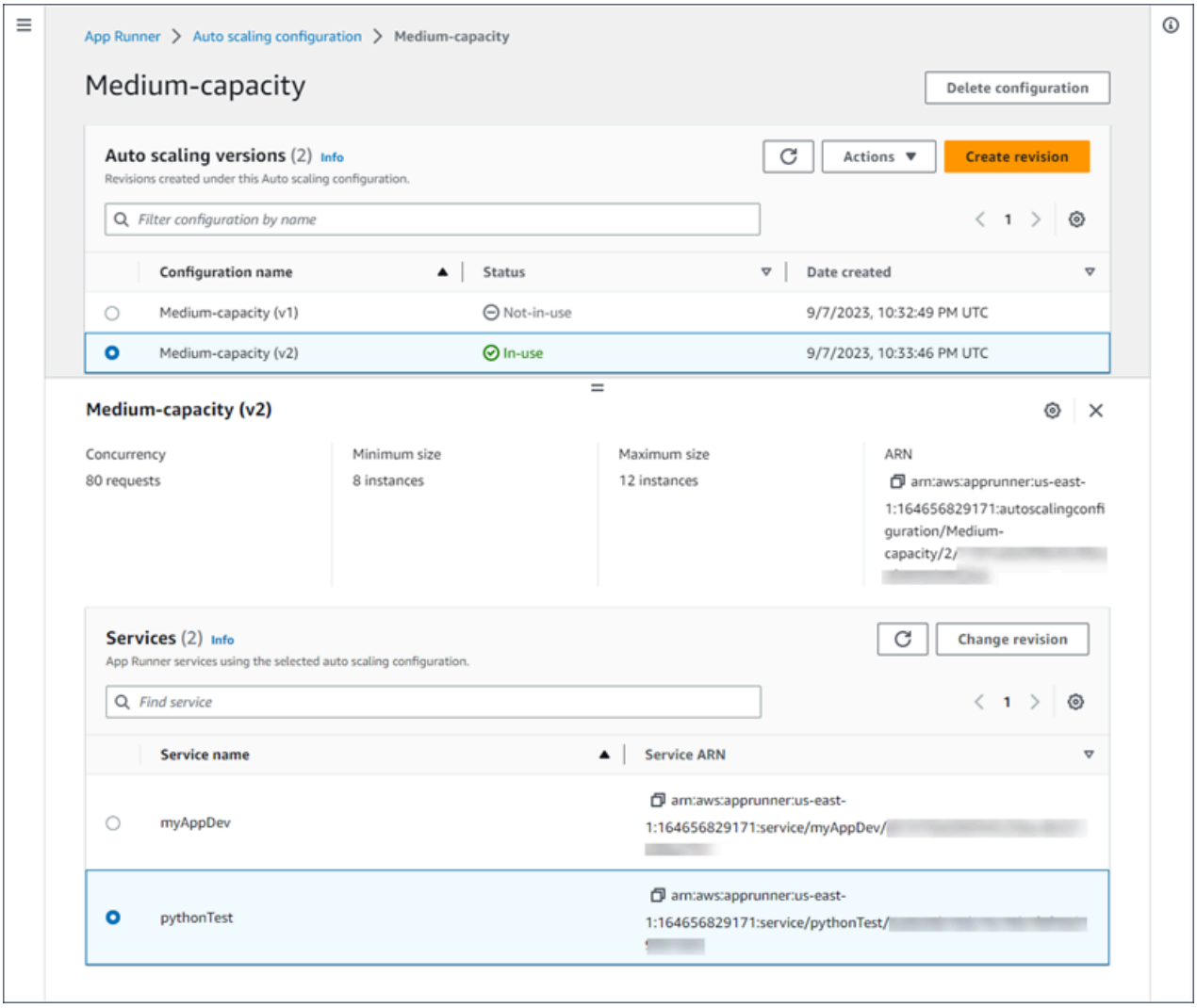
    Die Konfigurationsdetails für die Revision, einschließlich des ARN, werden im unteren geteilten Bereich angezeigt. Sehen Sie sich das Bildschirmbild am Ende dieses Vorgangs an.
- Gehen Sie wie folgt vor, um eine Liste der mit einer Revision verknüpften Dienste anzuzeigen.
  - Wählen Sie das Optionsfeld neben der Revision aus.

    Der Bereich Dienste wird im unteren geteilten Bereich unter den Konfigurationsdetails der Revision angezeigt. Das Panel listet alle Dienste auf, die diese Version der Auto Scaling-Konfiguration verwenden. Sehen Sie sich das Bildschirmbild am Ende dieses Vorgangs an.
- Gehen Sie wie folgt vor, um die Version für einen aufgelisteten Dienst zu ändern.
  a. Wählen Sie das Optionsfeld neben der Revision aus, falls Sie dies noch nicht getan haben.

     Der Bereich Dienste wird im unteren geteilten Bereich unter den Konfigurationsdetails der Revision angezeigt. Das Panel listet alle Dienste auf, die diese Version der Auto Scaling-Konfiguration verwenden. Sehen Sie sich das Bildschirmbild am Ende dieses Vorgangs an.
  b. Wählen Sie im Bereich Dienste das Optionsfeld neben dem Dienst aus, den Sie ändern möchten. Wählen Sie dann Revisionen ändern aus.
  c. Das Fenster ASC-Version ändern wird angezeigt. Wählen Sie im Drop-down-Menü eine der verfügbaren Versionen aus. Nur die Versionen der Auto Scaling-Konfiguration, die Sie zuvor ausgewählt haben, sind verfügbar. Wenn Sie zu einer anderen Auto Scaling-Konfiguration wechseln müssen, gehen Sie wie im vorherigen Abschnitt beschrieben vorthe section called “Auto Scaling für einen Service verwalten”.

Wählen Sie Aktualisieren aus, um mit der Änderung fortzufahren. Wählen Sie andernfalls Abbrechen aus.

> **Note**
>
> Wenn Sie eine Revision ändern, die einem Service zugeordnet ist, wird Ihr Service erneut bereitgestellt.
> Sie müssen in diesem Bereich die Option Aktualisieren auswählen, um die aktualisierten Verknüpfungen zu sehen.
> Um die laufenden Aktivitäten und den Status der erneuten Bereitstellung des Dienstes zu sehen, navigieren Sie in den Breadcrumbs des Panels zu App Runner > Dienste, wählen Sie den Dienst aus und rufen Sie dann im Bereich Dienstübersicht die Registerkarte Protokolle auf.



App Runner API or AWS CLI

Verwenden Sie die folgenden App Runner API-Aktionen, um Ihre Auto Scaling-Konfigurationsressourcen zu verwalten.

- CreateAutoScalingConfiguration— Erstellt eine neue Auto Scaling-Konfiguration oder eine Revision einer vorhandenen.
- UpdateDefaultAutoScalingConfiguration— Legt eine Auto Scaling-Konfiguration als Standard fest. Die bestehende Standardkonfiguration für auto Skalieren wird automatisch auf „Nicht standardmäßig“ gesetzt.
- ListAutoScalingConfigurations— Gibt eine Liste der Auto Scaling-Konfigurationen zurück, die mit Ihrer verknüpft sind AWS-Konto, mit zusammenfassenden Informationen.

- ListServicesForAutoScalingConfiguration— Gibt eine Liste der zugehörigen App Runner-Dienste zurück, die eine Auto Scaling-Konfiguration verwenden.
- DescribeAutoScalingConfiguration— Gibt eine vollständige Beschreibung einer Auto Scaling-Konfiguration zurück.
- DeleteAutoScalingConfiguration— Löscht eine Auto Scaling-Konfiguration. Sie können eine Auto Scaling-Konfiguration der obersten Ebene, eine bestimmte Revision einer oder alle Revisionen löschen, die der Konfiguration der obersten Ebene zugeordnet sind. Verwenden Sie den optionalen `DeleteAllRevisions` Parameter, um alle Revisionen zu löschen. Wenn Sie das für Sie erforderliche Ressourcenkontingent für die Auto Scaling-Konfiguration erreichen AWS-Konto, müssen Sie möglicherweise nicht benötigte Auto Scaling-Konfigurationen löschen.

Verwaltung benutzerdefinierter Domainnamen für einen App Runner-Dienst

Wenn Sie einen AWS App Runner Dienst erstellen, weist App Runner ihm einen Domainnamen zu. Dies ist eine Subdomain in der `awsapprunner.com` Domain, die App Runner gehört. Sie können den Domainnamen verwenden, um auf die Webanwendung zuzugreifen, die in Ihrem Dienst ausgeführt wird.

> **Note**
>
> Um die Sicherheit Ihrer App Runner-Anwendungen zu erhöhen, ist die Domain*.awsapprunner.com in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihre App Runner-Anwendungen einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Wenn Sie einen Domainnamen besitzen, können Sie ihn mit Ihrem App Runner-Dienst verknüpfen. Nachdem App Runner Ihre neue Domain validiert hat, können Sie Ihre Domain zusätzlich zur App Runner-Domain für den Zugriff auf Ihre Anwendung verwenden. Sie können bis zu fünf benutzerdefinierte Domains verknüpfen.

> **Note**
>
> Sie können optional die `www` Subdomain Ihrer Domain einbeziehen. Dies wird derzeit jedoch nur von der API unterstützt. Die App Runner-Konsole unterstützt das Einbeziehen einer `www` Subdomain Ihrer Domain nicht.

> **Note**
>
> AWS App Runner unterstützt nicht die Verwendung von privaten, gehosteten Zonen auf Route 53. Private gehostete Zonen passen die Auflösung von Domainnamen für den Amazon VPC-Verkehr an. Weitere Informationen zu privat gehosteten Zonen finden Sie unter Arbeiten mit privat gehosteten Zonen in der Route 53-Dokumentation.

Ordnen Sie Ihrem Service eine benutzerdefinierte Domain zu (verknüpfen)

Wenn Sie Ihrem Dienst eine benutzerdefinierte Domain zuordnen, müssen Sie die CNAME-Einträge und DNS-Zieleinträge zu Ihrem DNS-Server hinzufügen. Die folgenden Abschnitte enthalten Informationen zu CNAME-Einträgen und DNS-Zieldatensätzen und deren Verwendung.

> **Note**
>
> Wenn Sie Amazon Route 53 als Ihren DNS-Anbieter verwenden, konfiguriert App Runner Ihre benutzerdefinierte Domain automatisch mit der erforderlichen Zertifikatsvalidierung und den DNS-Einträgen, um eine Verbindung zu Ihrer App Runner-Webanwendung herzustellen. Dies passiert, wenn Sie die App Runner-Konsole verwenden, um Ihre benutzerdefinierte Domain mit Ihrem Service zu verknüpfen. Das folgende Verwalte benutzerdefinierte Domains Thema enthält weitere Informationen.

CNAME-Datensätze

Wenn Sie Ihrem Dienst eine benutzerdefinierte Domain zuordnen, stellt Ihnen App Runner eine Reihe von Zertifikatsvalidierungsdatensätzen für die Zertifikatsvalidierung zur Verfügung. Sie müssen diese Datensätze zur Zertifikatsvalidierung zu Ihrem DNS-Server (Domain Name System) hinzufügen. Fügen Sie die von App Runner bereitgestellten Datensätze zur Zertifikatsvalidierung Ihrem DNS-

Server hinzu. Auf diese Weise kann App Runner überprüfen, ob Sie die Domain besitzen oder kontrollieren.

> **Note**
>
> Um Ihre benutzerdefinierten Domainzertifikate automatisch zu verlängern, stellen Sie sicher, dass Sie die Einträge zur Zertifikatsvalidierung nicht von Ihrem DNS-Server löschen. Informationen zur Lösung von Problemen im Zusammenhang mit der Verlängerung des Zertifikats finden Sie unterthe section called “Verlängerung des benutzerdefinierten Domainzertifikats”.

App Runner verwendet ACM, um die Domain zu verifizieren. Wenn Sie CAA-Einträge in Ihren DNS-Einträgen verwenden, stellen Sie sicher, dass mindestens ein CAA-Eintrag darauf verweist. `amazon.com` Andernfalls kann ACM die Domain nicht verifizieren und Ihre Domain nicht erfolgreich erstellen.

Wenn Sie Fehler im Zusammenhang mit CAA erhalten, finden Sie unter den folgenden Links Informationen zur Behebung dieser Fehler:

- Probleme mit der Autorisierung durch die Zertifizierungsstelle (CAA)
- Wie behebe ich CAA-Fehler bei der Ausstellung oder Verlängerung eines ACM-Zertifikats?
- Benutzerdefinierte Domainnamen

> **Note**
>
> Wenn Sie Amazon Route 53 als Ihren DNS-Anbieter verwenden, konfiguriert App Runner Ihre benutzerdefinierte Domain automatisch mit der erforderlichen Zertifikatsvalidierung und den DNS-Einträgen, um eine Verbindung zu Ihrer App Runner-Webanwendung herzustellen. Dies passiert, wenn Sie die App Runner-Konsole verwenden, um Ihre benutzerdefinierte Domain mit Ihrem Service zu verknüpfen. Das folgende Verwalte benutzerdefinierte Domains Thema enthält weitere Informationen.

DNS-Zieldatensätze

Fügen Sie die DNS-Zieldatensätze zu Ihrem DNS-Server hinzu, um die App Runner-Domain als Ziel zu verwenden. Fügen Sie einen Datensatz für die benutzerdefinierte Domain und einen weiteren für

die www Subdomain hinzu, wenn Sie diese Option gewählt haben. Warten Sie dann, bis der Status der benutzerdefinierten Domain in der App Runner-Konsole auf Aktiv gesetzt ist. Dies dauert in der Regel mehrere Minuten, kann aber auch bis zu 24—48 Stunden (1—2 Tage) dauern. Wenn Ihre benutzerdefinierte Domain validiert ist, leitet App Runner den Datenverkehr von dieser Domain an Ihre Webanwendung weiter.

> **Note**
>
> Für eine bessere Kompatibilität mit App Runner-Diensten empfehlen wir, Amazon Route 53 als DNS-Anbieter zu verwenden. Wenn Sie Amazon Route 53 nicht zur Verwaltung Ihrer öffentlichen DNS-Einträge verwenden, wenden Sie sich an Ihren DNS-Anbieter, um zu erfahren, wie Sie Einträge hinzufügen können.
> Wenn Sie Amazon Route 53 als Ihren DNS-Anbieter verwenden, können Sie entweder einen CNAME- oder einen Aliaseintrag für die Subdomain hinzufügen. Stellen Sie für die Root-Domain sicher, dass Sie den Alias-Datensatz verwenden.

Sie können einen Domainnamen von Amazon Route 53 oder einem anderen Anbieter erwerben. Informationen zum Kauf eines Domainnamens bei Amazon Route 53 finden Sie unter Registrierung einer neuen Domain im Amazon Route 53-Entwicklerhandbuch.

Anweisungen zur Konfiguration eines DNS-Ziels in Route 53 finden Sie unter Weiterleiten von Datenverkehr zu Ihren Ressourcen im Amazon Route 53-Entwicklerhandbuch.

Anweisungen zur Konfiguration eines DNS-Ziels bei anderen Registraren wie Shopify GoDaddy, Hover usw. finden Sie in der jeweiligen Dokumentation zum Hinzufügen von DNS-Zieldatensätzen.

Geben Sie eine Domain an, die mit Ihrem App Runner-Dienst verknüpft werden soll

Sie können auf folgende Weise eine Domain angeben, die mit Ihrem App Runner-Dienst verknüpft werden soll:

- Eine Root-Domain — DNS hat einige inhärente Einschränkungen, die Sie möglicherweise daran hindern, CNAME-Einträge für den Root-Domainnamen zu erstellen. Wenn Ihr Domainname beispielsweise lautet`example.com`, können Sie einen CNAME-Eintrag erstellen, der den Datenverkehr `acme.example.com` an Ihren App Runner-Dienst weiterleitet. Sie können jedoch keinen CNAME-Eintrag erstellen, der den Traffic `example.com` an Ihren App Runner-Dienst weiterleitet. Um eine Root-Domain zu erstellen, stellen Sie sicher, dass Sie einen Alias-Datensatz hinzufügen.

Ein Aliaseintrag ist spezifisch für Route 53 und bietet gegenüber CNAME-Einträgen die folgenden Vorteile:

- Route 53 bietet Ihnen mehr Flexibilität, da Aliaseinträge für die Stammdomain oder Subdomain erstellt werden können. Wenn Ihr Domainname beispielsweise lautete`example.com`, können Sie einen Datensatz erstellen, der Anfragen für `example.com` oder `acme.example.com` an Ihren App Runner-Dienst weiterleitet.
- Das ist kostengünstiger. Dies liegt daran, dass Route 53 keine Gebühren für Anfragen erhebt, die einen Aliaseintrag zum Weiterleiten des Datenverkehrs verwenden.

- Eine Subdomain — zum Beispiel, `login.example.com` oder`admin.login.example.com`. Sie können die `www` Subdomain optional auch als Teil desselben Vorgangs zuordnen. Sie können entweder einen CNAME-Datensatz oder einen Alias-Datensatz für die Subdomain hinzufügen.
- Ein Platzhalter — Zum Beispiel. `*.example.com` In diesem Fall können Sie die `www` Option nicht verwenden. Sie können einen Platzhalter nur als unmittelbare Subdomain einer Stammdomain und nur als eigenständige Domain angeben. Dies sind keine gültigen Spezifikationen:`login*.example.com`,. `*.login.example.com` Diese Platzhalterspezifikation ordnet alle unmittelbaren Subdomänen zu und ordnet nicht die Stammdomäne selbst zu. Die Stammdomäne muss in einem separaten Vorgang zugeordnet werden.

Eine spezifischere Domänenzuweisung hat Vorrang vor einer weniger spezifischen. Zum Beispiel `login.example.com` `*.example.com` Überschreibungen. Das Zertifikat und der CNAME der spezifischeren Assoziation werden verwendet.

Das folgende Beispiel zeigt, wie Sie mehrere benutzerdefinierte Domänenzuordnungen verwenden können:

1. Stellen Sie eine Verknüpfung `example.com` mit der Startseite Ihres Dienstes her. Aktivieren Sie `www` die Verknüpfung`www.example.com`.
2. Stellen Sie eine `login.example.com` Verbindung zur Anmeldeseite Ihres Dienstes her.
3. `*.example.com`Mit einer benutzerdefinierten Seite „Nicht gefunden“ verknüpfen.

Eine benutzerdefinierte Domain trennen (Verknüpfung aufheben)

Sie können eine benutzerdefinierte Domain von Ihrem App Runner-Dienst trennen (die Verknüpfung aufheben). Wenn Sie die Verknüpfung einer Domain aufheben, beendet App Runner die Weiterleitung des Datenverkehrs von dieser Domain zu Ihrer Webanwendung.

> **Note**
>
> Sie müssen die Einträge für die Domain löschen, die Sie von Ihrem DNS-Server getrennt haben.

App Runner erstellt intern Zertifikate, die die Gültigkeit der Domain nachverfolgen. Diese Zertifikate werden in AWS Certificate Manager (ACM) gespeichert. App Runner löscht diese Zertifikate 7 Tage lang nicht, nachdem eine Domain von Ihrem Dienst getrennt wurde oder nachdem der Dienst gelöscht wurde.

Verwalte benutzerdefinierte Domains

Verwalten Sie benutzerdefinierte Domains für Ihren App Runner-Dienst mit einer der folgenden Methoden:

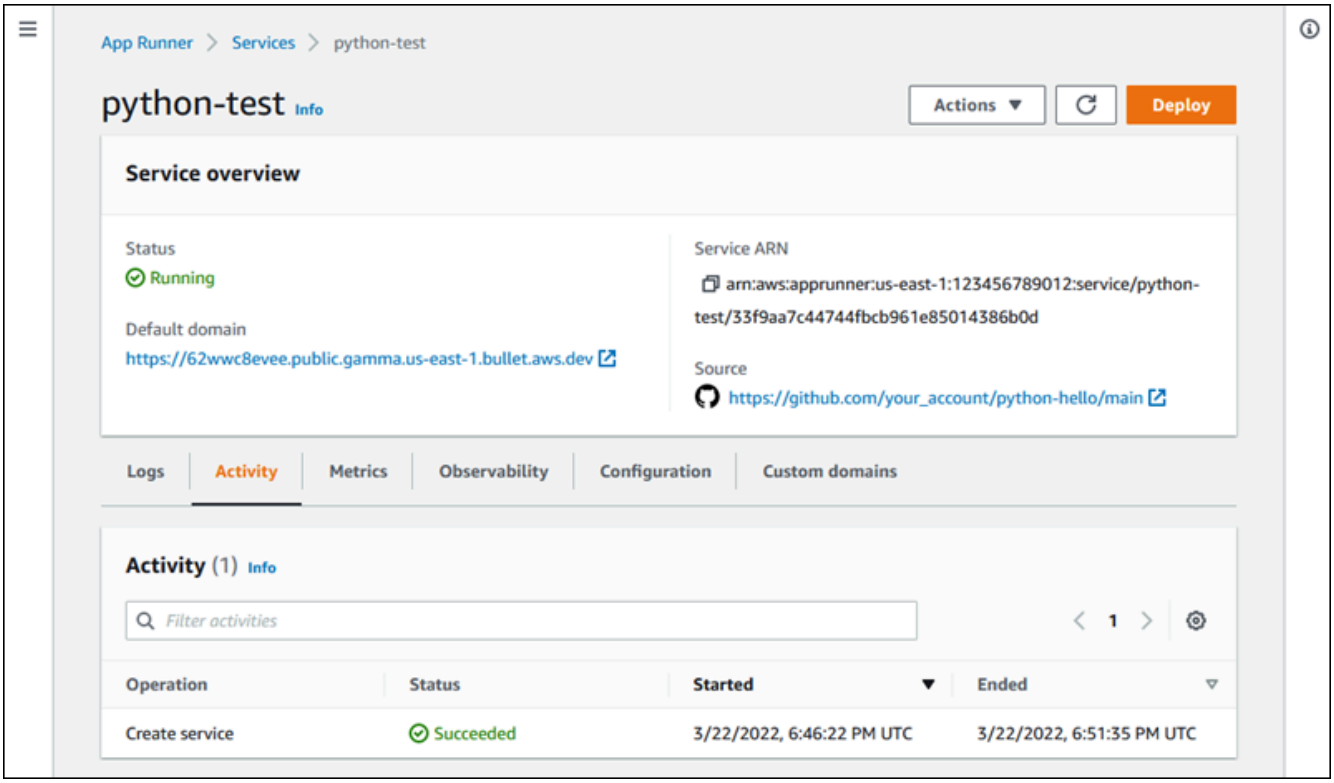
> **Note**
>
> Für eine bessere Kompatibilität mit App Runner-Diensten empfehlen wir, Amazon Route 53 als DNS-Anbieter zu verwenden. Wenn Sie Amazon Route 53 nicht zur Verwaltung Ihrer öffentlichen DNS-Einträge verwenden, wenden Sie sich an Ihren DNS-Anbieter, um zu erfahren, wie Sie Einträge hinzufügen können.
> Wenn Sie Amazon Route 53 als Ihren DNS-Anbieter verwenden, können Sie entweder einen CNAME- oder einen Aliaseintrag für die Subdomain hinzufügen. Stellen Sie für die Root-Domain sicher, dass Sie den Alias-Datensatz verwenden.

App Runner console

Um eine benutzerdefinierte Domain mithilfe der App Runner-Konsole zuzuordnen (zu verknüpfen)

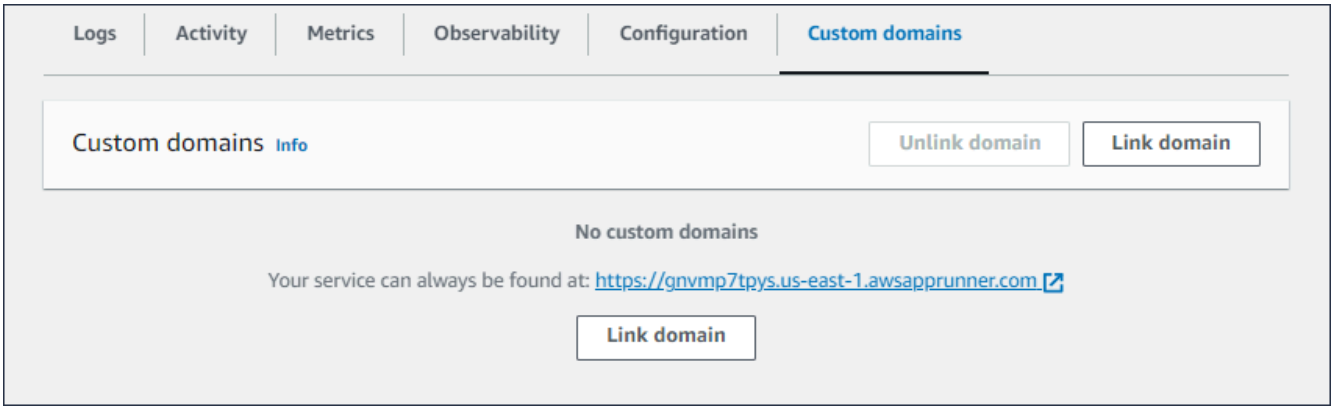
1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

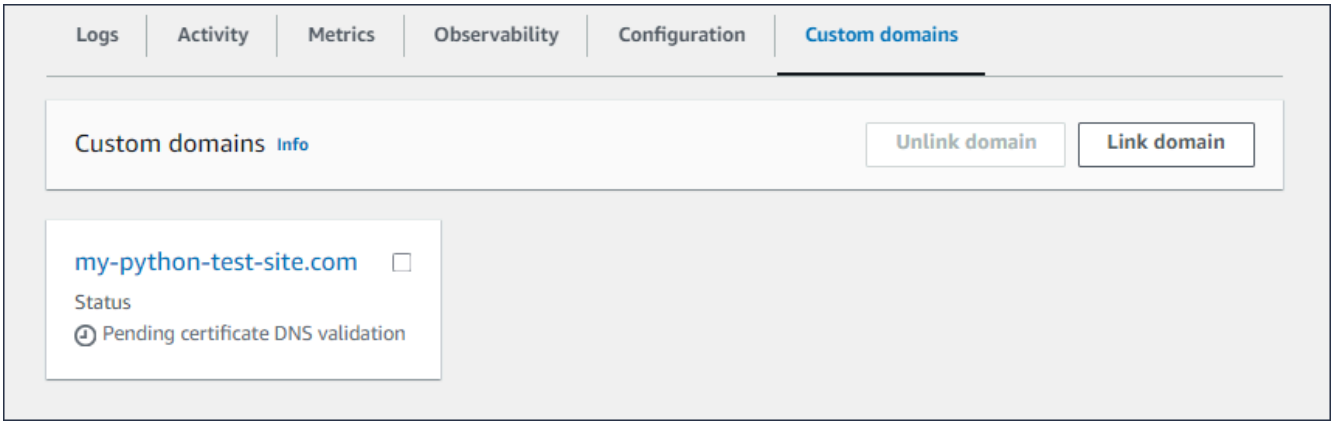
   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite den Tab Benutzerdefinierte Domains aus.

   In der Konsole werden die benutzerdefinierten Domänen angezeigt, die mit Ihrem Dienst verknüpft sind, oder „Keine benutzerdefinierten Domänen“.

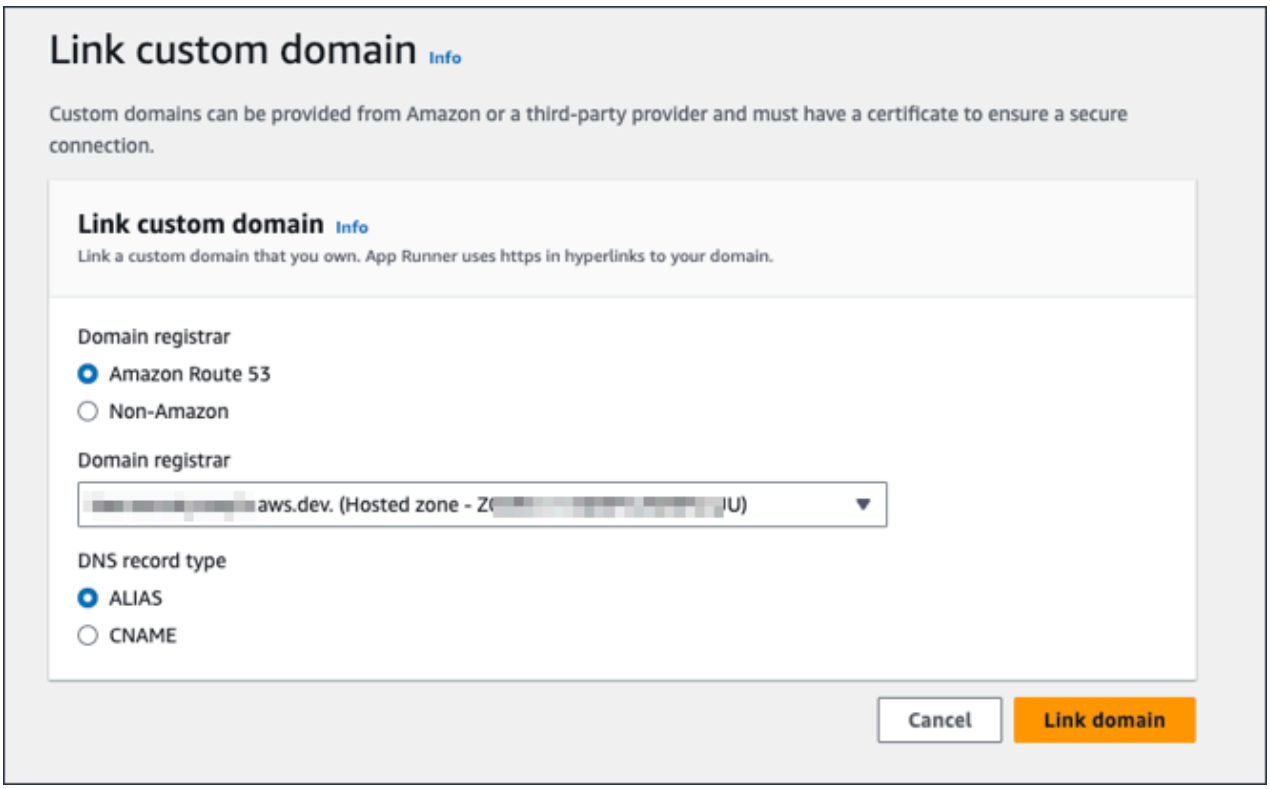




4. Wählen Sie auf der Registerkarte Benutzerdefinierte Domains die Option Domain verknüpfen aus.
5. Die Seite Benutzerdefinierte Domain verknüpfen wird angezeigt.
   - Wenn Ihre benutzerdefinierte Domain bei Amazon Route 53 registriert ist, wählen Sie Amazon Route 53 als Domain-Registrar aus.
     a. Wählen Sie den Domainnamen aus der Drop-down-Liste aus. In dieser Liste werden der Name Ihrer Route 53-Domainnamen und die Hosting-Zonen-ID angezeigt.

        > **Note**
        >
        > Sie müssen zunächst eine Route 53-Domain mithilfe des Amazon Route 53-Service von demselben AWS Konto aus erstellen, das Sie für die Verwaltung Ihrer anderen App Runner-Ressourcen verwenden.

     b. Wählen Sie den DNS-Eintragstyp aus.
     c. Wählen Sie Link-Domain.

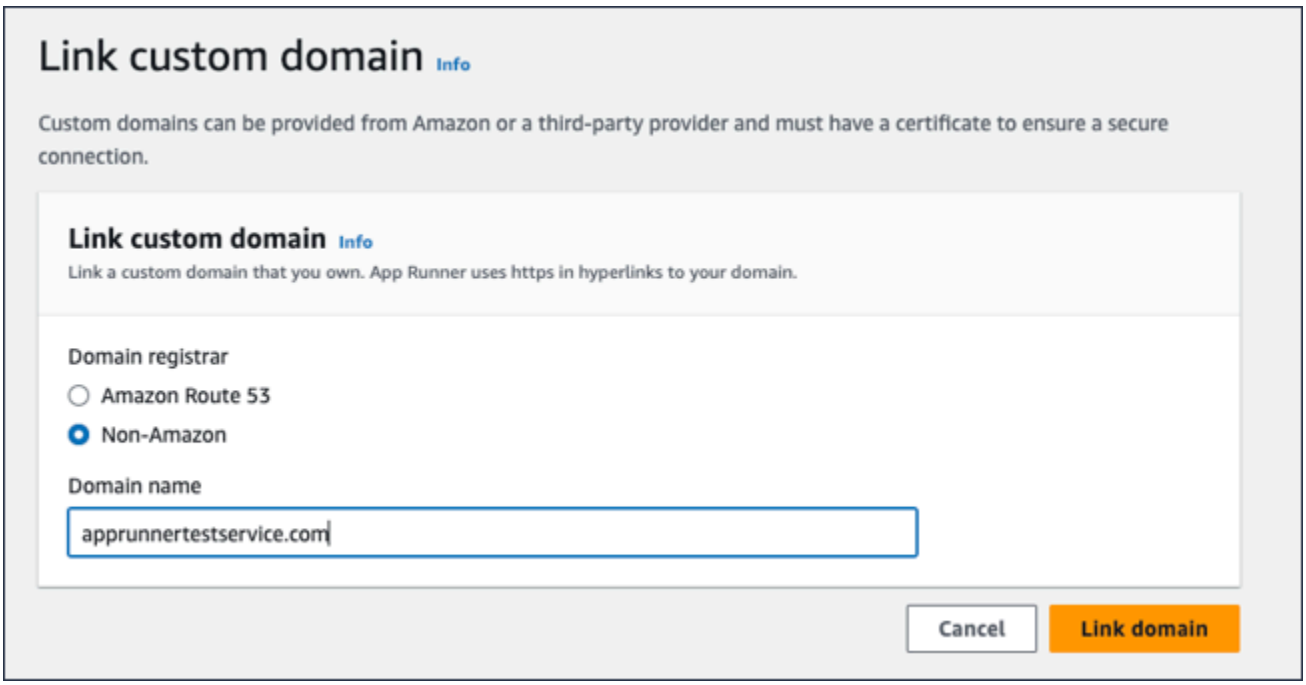


> **Note**
>
> Wenn App Runner eine Fehlermeldung anzeigt, dass der automatische Konfigurationsversuch fehlgeschlagen ist, können Sie fortfahren, indem Sie die DNS-Einträge manuell konfigurieren. Dieses Problem kann auftreten, wenn derselbe Domainname zuvor von einem Dienst getrennt wurde, ohne dass die DNS-Anbieter-Einträge, die auf den Dienst hinweisen, anschließend gelöscht wurden. In diesem Fall wird App Runner daran gehindert, diese Datensätze automatisch zu überschreiben. Um die DNS-Konfiguration abzuschließen, überspringen Sie die restlichen Schritte in diesem Verfahren und folgen Sie dann den Anweisungen unter. Einen Amazon Route 53-Aliaseintrag konfigurieren

- Wenn Ihre benutzerdefinierte Domain bei einem anderen Domain-Registrar registriert ist, wählen Sie Non-Amazon als Domain-Registrar aus.

  a. Geben Sie den Domainnamen ein.

  b. Wählen Sie Link-Domain.



6. Die Seite „DNS konfigurieren“ wird angezeigt.

   - Wenn Amazon Route 53 es Ihr DNS-Anbieter ist, ist dieser Schritt optional.

     Zu diesem Zeitpunkt hat App Runner Ihre Route 53-Domain automatisch mit den erforderlichen Zertifikatsvalidierungen und DNS-Einträgen konfiguriert.

     > **Note**
     >
     > Wenn derselbe Domainname zuvor von einem Dienst getrennt wurde, ohne dass die DNS-Anbietereinträge, die auf den Dienst hinweisen, anschließend gelöscht wurden, wäre die automatische Konfiguration, die App Runner versucht hat, möglicherweise fehlgeschlagen. Um dieses Problem zu umgehen und die DNS-Zuordnung abzuschließen, fahren Sie mit den Schritten (1) und (2) auf der Seite „DNS konfigurieren“ fort, um die aktuellen Ziel- und Zertifikatseinträge auf den DNS-Anbieter zu kopieren.

   - Kopieren Sie die Datensätze zur Zertifikatsvalidierung und die DNS-Zieldatensätze und fügen Sie sie Ihrem DNS-Server hinzu. App Runner kann dann überprüfen, ob Sie die Domain besitzen oder kontrollieren.

> **Note**
>
> Um Ihre benutzerdefinierten Domainzertifikate automatisch zu verlängern, stellen Sie sicher, dass Sie die Einträge zur Zertifikatsvalidierung nicht von Ihrem DNS-Server löschen.

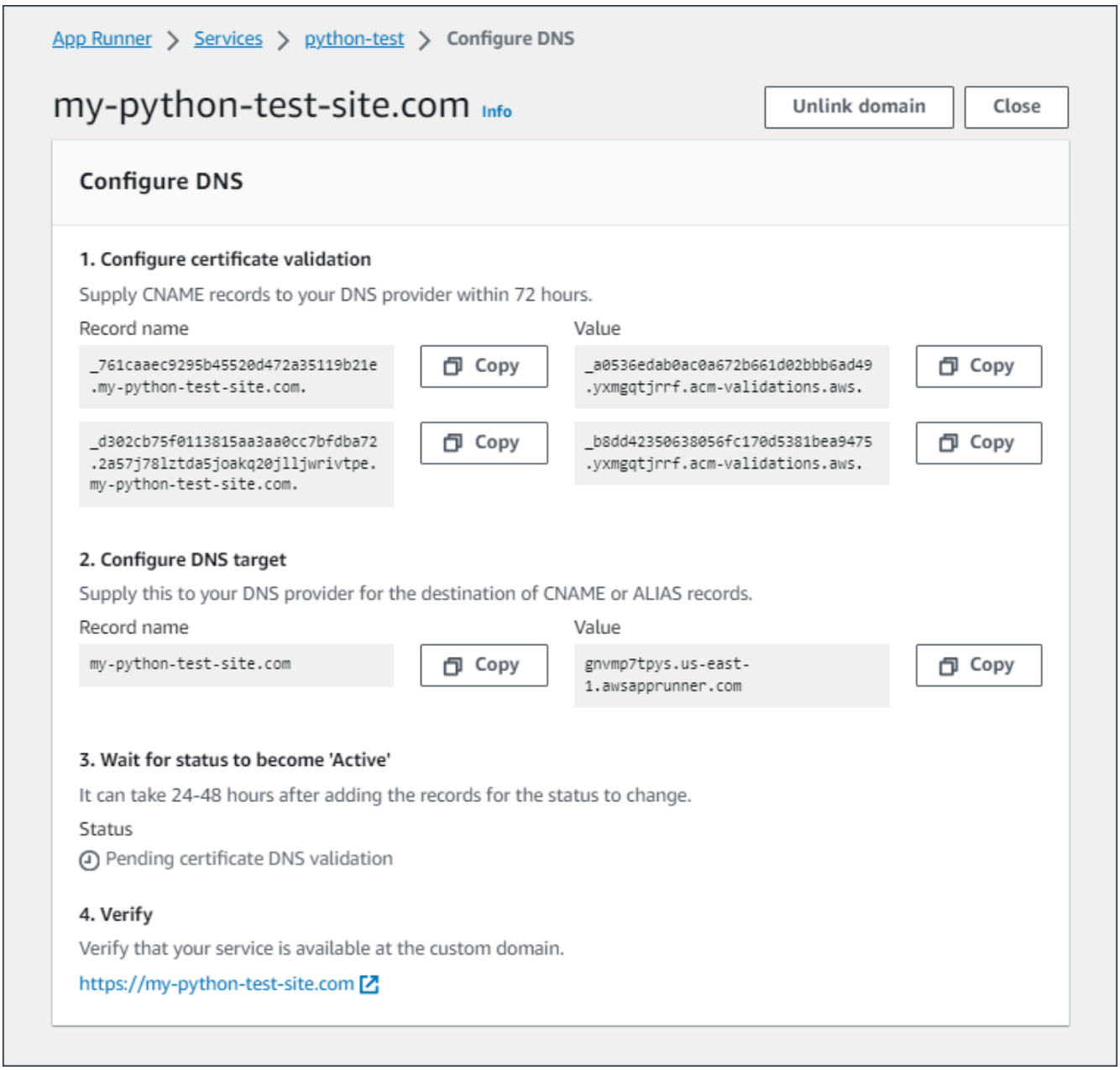
- Weitere Informationen zur Konfiguration der Zertifikatsvalidierung finden Sie unter DNS-Validierung im AWS Certificate Manager Benutzerhandbuch.
- Informationen zur Konfiguration eines DNS-Ziels mit einem Amazon Route 53-Aliaseintrag finden Sie unterthe section called “Einen Amazon Route 53-Aliaseintrag konfigurieren”.

- Wenn Sie einen anderen DNS-Anbieter als Amazon Route 53 verwenden, gehen Sie wie folgt vor.

  - Kopieren Sie die Datensätze zur Zertifikatsvalidierung und die DNS-Zieldatensätze und fügen Sie sie Ihrem DNS-Server hinzu. App Runner kann dann überprüfen, ob Sie die Domain besitzen oder kontrollieren.

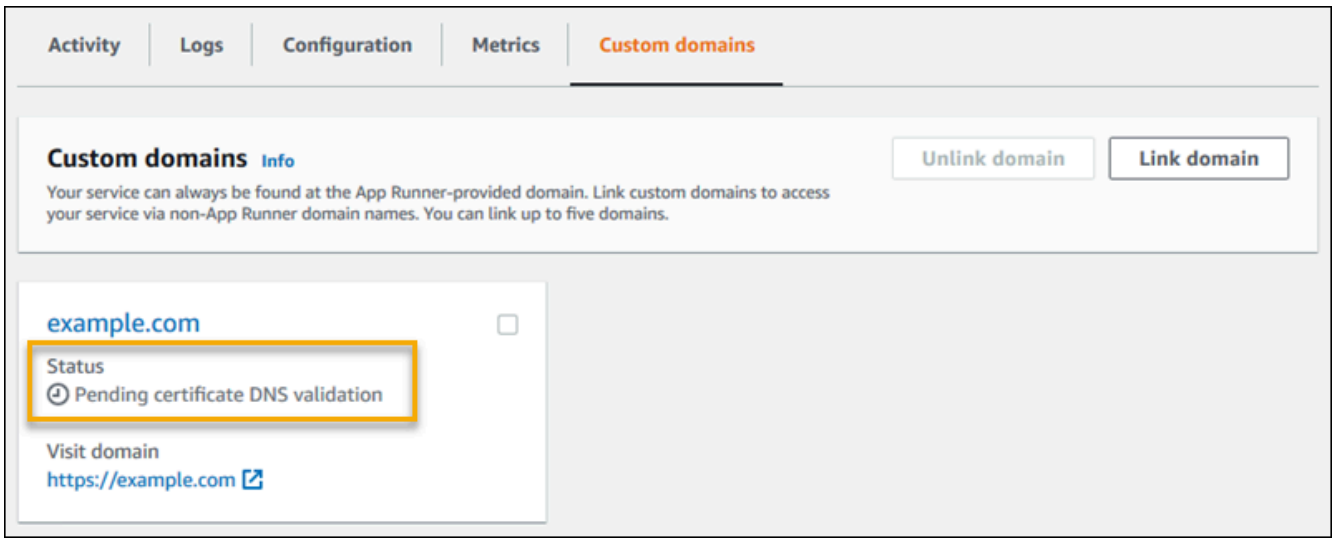
    > **Note**
    >
    > Um Ihre benutzerdefinierten Domainzertifikate automatisch zu verlängern, stellen Sie sicher, dass Sie die Einträge zur Zertifikatsvalidierung nicht von Ihrem DNS-Server löschen.

    - Weitere Informationen zur Konfiguration der Zertifikatsvalidierung finden Sie unter DNS-Validierung im AWS Certificate Manager Benutzerhandbuch.
    - Anweisungen zur Konfiguration eines DNS-Ziels bei anderen Registraren wie Shopify GoDaddy, Hover usw. finden Sie in der jeweiligen Dokumentation zum Hinzufügen von DNS-Zielen.

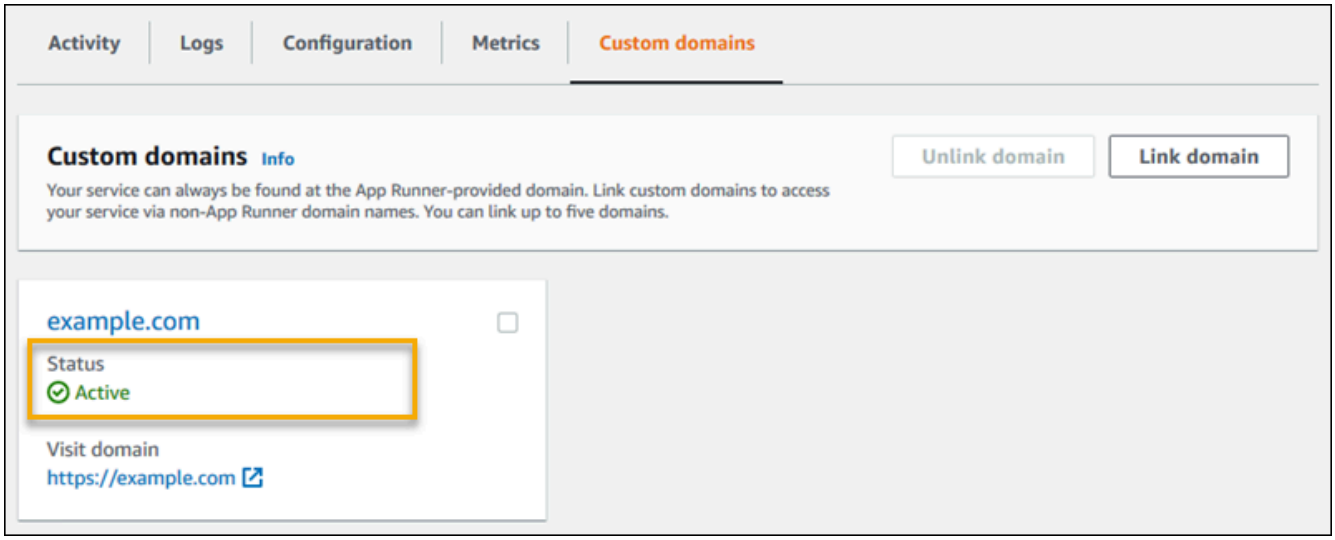


7. Wählen Sie „Schließen“

In der Konsole wird das Dashboard wieder angezeigt. Die Registerkarte Benutzerdefinierte Domänen enthält eine neue Kachel, in der die Domain angezeigt wird, die Sie gerade verknüpft haben, im DNS-Validierungsstatus „Ausstehendes Zertifikat“.



8. Wenn sich der Domainstatus auf Aktiv ändert, überprüfen Sie, ob die Domain für die Weiterleitung von Traffic funktioniert, indem Sie zu ihr navigieren.



> **Note**
>
> Anweisungen zur Behebung von Fehlern im Zusammenhang mit einer benutzerdefinierten Domain finden Sie unterthe section called “Benutzerdefinierte Domainnamen”.

So trennen Sie mit der App Runner-Konsole die Verknüpfung zu einer benutzerdefinierten Domain (heben die Verknüpfung auf)

1. Wählen Sie auf der Registerkarte Benutzerdefinierte Domains die Kachel für die Domain aus, deren Zuordnung Sie aufheben möchten, und wählen Sie dann Domainverknüpfung aufheben aus.
2. Überprüfen Sie im Dialogfeld Domainverknüpfung aufheben die Aktion, indem Sie Domainverknüpfung aufheben wählen.

> **Note**
>
> Sie müssen die Einträge für die Domain löschen, die Sie von Ihrem DNS-Server getrennt haben.

App Runner API or AWS CLI

Um Ihrem Dienst mithilfe der App Runner API oder eine benutzerdefinierte Domain zuzuordnen AWS CLI, rufen Sie die AssociateCustomDomainAPI-Aktion auf. Wenn der Aufruf erfolgreich ist, wird ein CustomDomainObjekt zurückgegeben, das die benutzerdefinierte Domain beschreibt, die Ihrem Service zugeordnet ist. Das Objekt zeigt einen `CREATING` Status an und enthält eine Liste von CertificateValidationRecordObjekten. Der Aufruf gibt auch den Zielalias zurück, mit dem Sie das DNS-Ziel konfigurieren können. Dies sind Einträge, die Sie Ihrem DNS hinzufügen können.

Um eine benutzerdefinierte Domain mithilfe der App Runner API von Ihrem Dienst zu trennen oder AWS CLI rufen Sie die DisassociateCustomDomainAPI-Aktion auf. Wenn der Aufruf erfolgreich ist, wird ein CustomDomainObjekt zurückgegeben, das die benutzerdefinierte Domain beschreibt, die von Ihrem Dienst getrennt wird. Das Objekt zeigt einen `DELETING` Status an.

Themen

- Amazon Route 53-Aliaseintrag für Ihren Ziel-DNS konfigurieren

Amazon Route 53-Aliaseintrag für Ihren Ziel-DNS konfigurieren

> **Note**
>
> Sie müssen dieses Verfahren nicht befolgen, wenn Amazon Route 53 Ihr DNS-Anbieter ist. In diesem Fall konfiguriert App Runner Ihre Route 53-Domain automatisch mit den erforderlichen Zertifikatsvalidierungen und DNS-Einträgen, um eine Verbindung zu Ihrer App Runner-Webanwendung herzustellen.
> Wenn der automatische Konfigurationsversuch von App Runner fehlgeschlagen ist, gehen Sie wie folgt vor, um die DNS-Konfiguration abzuschließen. Wenn derselbe Domainname zuvor von einem Dienst getrennt wurde, ohne dass die DNS-Anbieter-Einträge, die auf den Dienst hinweisen, anschließend gelöscht wurden, wird App Runner daran gehindert, diese Einträge automatisch zu überschreiben. In diesem Verfahren wird erklärt, wie Sie sie manuell in Ihr Route 53-DNS kopieren.

Sie können Amazon Route 53 als Ihren DNS-Anbieter verwenden, um den Datenverkehr an Ihren App Runner-Service weiterzuleiten. Es ist ein hochverfügbarer und skalierbarer Domain Name System (DNS) -Webservice. Der Amazon Route 53-Datensatz enthält die Einstellungen, die steuern, wie der Verkehr an Ihren App Runner-Service weitergeleitet wird. Sie erstellen entweder einen CNAME-Eintrag oder einen ALIAS-Datensatz. Einen Vergleich von CNAME-Datensätzen und Alias-Datensätzen finden Sie unter Choosing between alias and non-alias records im Amazon Route 53 Developer Guide.

> **Note**
>
> Amazon Route 53 unterstützt derzeit Aliasdatensätze für Services, die nach dem 1. August 2022 erstellt wurden.

Amazon Route 53 console

So konfigurieren Sie den Amazon Route 53-Aliaseintrag

1. Melden Sie sich bei der Route 53-Konsole an AWS Management Console und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Hosted Zones (Gehostete Zonen).
3. Wählen Sie den Namen der gehosteten Zone aus, die Sie verwenden möchten, um den Datenverkehr an Ihren App Runner-Dienst weiterzuleiten.

4. Wählen Sie Datensatz erstellen.
5. Geben Sie die folgenden Werte an:

   - Routing-Richtlinie: Wählen Sie die entsprechende Routing-Richtlinie aus. Weitere Informationen finden Sie unter Auswahl einer Routing-Richtlinie.
   - Datensatzname: Geben Sie den Domainnamen ein, den Sie verwenden möchten, um Traffic an Ihren App Runner-Dienst weiterzuleiten. Der Standardwert ist der Name der gehosteten Zone. Wenn der Name der gehosteten Zone beispielsweise lautet `example.com` und Sie den Datenverkehr an Ihre Umgebung weiterleiten möchten, geben Sie ein`acme`. `acme.example.com`
   - Traffic werten/weiterleiten an: Wählen Sie Alias to App Runner Application und wählen Sie dann die Region aus, aus der der Endpunkt stammt. Wählen Sie den Domainnamen der Anwendung aus, zu der Sie den Datenverkehr weiterleiten möchten.
   - Datensatztyp: Akzeptieren Sie die Standardadresse A — IPv4 Adresse.
   - Zustand des Ziels auswerten: Akzeptieren Sie den Standardwert, Ja.
6. Wählen Sie Create records (Datensätze erstellen).

Der Route 53-Aliaseintrag, den Sie erstellt haben, wird innerhalb von 60 Sekunden auf allen Route 53-Servern verbreitet. Wenn die Route 53-Server mit Ihrem Aliaseintrag weitergegeben werden, können Sie den Verkehr an Ihren App Runner-Dienst weiterleiten, indem Sie den Namen des Aliaseintrags verwenden, den Sie erstellt haben.

Informationen zur Problembehandlung, wenn die Weitergabe der DNS-Änderungen zu lange dauert, finden Sie unter Warum dauert es so lange, bis meine DNS-Änderungen in Route 53 und öffentlichen Resolvern propagiert werden? .

Amazon Route 53 API or AWS CLI

Um den Amazon Route 53-Aliaseintrag mithilfe der Amazon Route 53-API zu konfigurieren oder die ChangeResourceRecordSetsAPI-Aktion AWS CLI aufzurufen. Weitere Informationen zur Ziel-Hosting-Zonen-ID von Route 53 finden Sie unter Service-Endpunkte.

Einen App Runner-Dienst anhalten und wieder aufnehmen

Wenn Sie Ihre Webanwendung vorübergehend deaktivieren und die Ausführung des Codes beenden müssen, können Sie Ihren AWS App Runner Dienst unterbrechen. App Runner reduziert die Rechenkapazität für den Service auf Null.

Wenn Sie bereit sind, Ihre Anwendung erneut auszuführen, können Sie Ihren App Runner-Dienst wieder aufnehmen. App Runner stellt neue Rechenkapazitäten bereit, stellt Ihre Anwendung bereit und führt die Anwendung aus. Ihre Anwendungsquelle wird nicht erneut bereitgestellt, und es ist kein Build erforderlich. Stattdessen fährt App Runner mit Ihrer aktuell bereitgestellten Version fort. Ihre Anwendung behält ihre App Runner-Domain.

Important

- Wenn Sie Ihren Dienst pausieren, verliert Ihre Anwendung ihren Status. Beispielsweise geht jeglicher kurzlebige Speicher, den Ihr Code verwendet hat, verloren. Für Ihren Code entspricht das Anhalten und Wiederaufnehmen Ihres Dienstes der Bereitstellung für einen neuen Dienst.
- Wenn Sie einen Dienst aufgrund eines Fehlers in Ihrem Code unterbrechen (z. B. aufgrund eines entdeckten Fehlers oder eines Sicherheitsproblems), können Sie keine neue Version bereitstellen, bevor Sie den Dienst wieder aufnehmen.

  Daher empfehlen wir, dass Sie den Dienst weiter ausführen und stattdessen zu Ihrer letzten stabilen Anwendungsversion zurückkehren.
- Wenn Sie Ihren Dienst wieder aufnehmen, stellt App Runner die letzte Anwendungsversion bereit, die verwendet wurde, bevor Sie den Dienst angehalten haben. Wenn Sie seit der Unterbrechung Ihres Dienstes neue Quellversionen hinzugefügt haben, stellt App Runner diese nicht automatisch bereit, auch wenn die automatische Bereitstellung ausgewählt ist. Nehmen wir zum Beispiel an, Sie haben neue Image-Versionen im Image-Repository oder neue Commits im Code-Repository. Diese Versionen werden nicht automatisch bereitgestellt.

  Um eine neuere Version bereitzustellen, führen Sie eine manuelle Bereitstellung durch oder fügen Sie Ihrem Quell-Repository eine weitere Version hinzu, nachdem Sie Ihren App Runner-Dienst wieder aufgenommen haben.

Pausieren und Löschen verglichen

Unterbrechen Sie Ihren App Runner-Dienst, um ihn vorübergehend zu deaktivieren. Nur Rechenressourcen werden beendet, und Ihre gespeicherten Daten (z. B. das Container-Image mit Ihrer Anwendungsversion) bleiben erhalten. Die Wiederaufnahme Ihres Dienstes ist schnell erledigt

— Ihre Anwendung ist bereit, für neue Rechenressourcen bereitgestellt zu werden. Ihre App Runner-Domain bleibt dieselbe.

Löschen Sie Ihren App Runner-Dienst, um ihn dauerhaft zu entfernen. Ihre gespeicherten Daten werden gelöscht. Wenn Sie den Dienst neu erstellen müssen, muss App Runner Ihre Quelle erneut abrufen und sie auch erstellen, falls es sich um ein Code-Repository handelt. Ihre Webanwendung erhält eine neue App-Runner-Domain.

Wenn Ihr Dienst angehalten ist

Wenn Sie Ihren Dienst pausieren und er sich im Status Angehalten befindet, reagiert er unterschiedlich auf Aktionsanfragen, einschließlich API-Aufrufe oder Konsolenoperationen. Wenn ein Dienst angehalten ist, können Sie immer noch App Runner-Aktionen ausführen, die die Definition oder Konfiguration des Dienstes nicht so ändern, dass sich dies auf seine Laufzeit auswirkt. Mit anderen Worten, wenn eine Aktion das Verhalten, den Umfang oder andere Eigenschaften eines laufenden Dienstes ändert, können Sie diese Aktion nicht für einen angehaltenen Dienst ausführen.

Die folgenden Listen enthalten Informationen zu API-Aktionen, die Sie für einen unterbrochenen Dienst ausführen können und die nicht. Die entsprechenden Konsolenoperationen sind in ähnlicher Weise erlaubt oder verweigert.

Aktionen, die Sie für einen unterbrochenen Dienst ausführen können

- *`List*`*und *`Describe*`* Aktionen — Aktionen, bei denen nur Informationen gelesen werden.
- *`DeleteService`*— Sie können einen Dienst jederzeit löschen.
- *`TagResource`*, *`UntagResource`* — Tags sind mit einem Dienst verknüpft, sind aber nicht Teil seiner Definition und haben keinen Einfluss auf sein Laufzeitverhalten.

Aktionen, die Sie bei einem angehaltenen Dienst nicht ausführen können

- *`StartDeployment`*Aktionen (oder eine manuelle Bereitstellung mithilfe der Konsole)
- *`UpdateService`*(oder eine Konfigurationsänderung mithilfe der Konsole, mit Ausnahme von Tagging-Änderungen)
- *`CreateCustomDomainAssociations`*, *`DeleteCustomDomainAssociations`*
- *`CreateConnection`*, *`DeleteConnection`*

Unterbrechen Sie Ihren Dienst und setzen Sie ihn fort

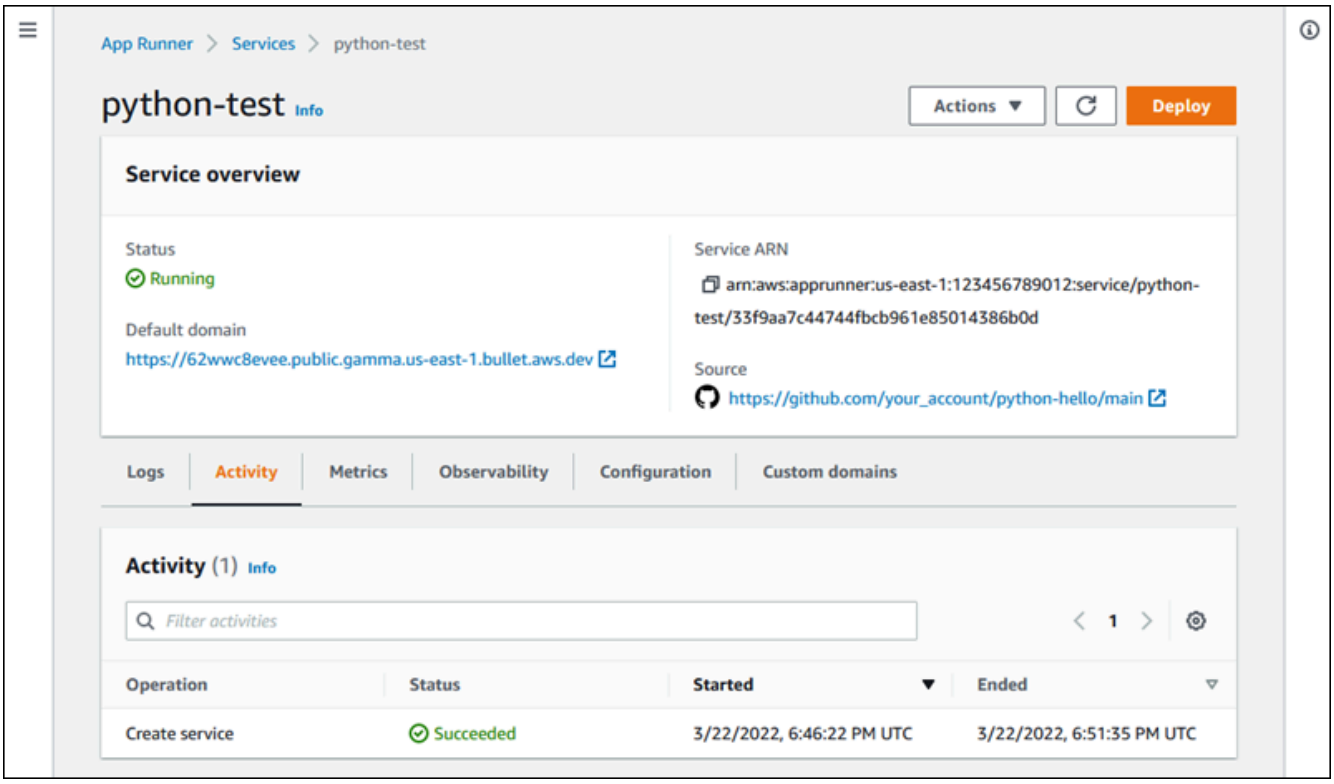
Unterbrechen Sie Ihren App Runner-Dienst und setzen Sie ihn mit einer der folgenden Methoden fort:

App Runner console

So unterbrechen Sie Ihren Dienst mithilfe der App Runner-Konsole

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie „Aktionen“ und anschließend „Pause“.

   Auf der Service-Dashboard-Seite ändert sich der Dienststatus in Betrieb in Bearbeitung und dann in Unterbrochen. Ihr Dienst ist jetzt angehalten.

Um Ihren Dienst mit der App Runner-Konsole wieder aufzunehmen

1. Wählen Sie „Aktionen“ und anschließend „Fortfahren“.

Auf der Service-Dashboard-Seite ändert sich der Servicestatus in Betrieb.

2. Warten Sie, bis der Dienst wieder aufgenommen wird. Auf der Service-Dashboard-Seite ändert sich der Dienststatus wieder in Wird ausgeführt.
3. Um zu überprüfen, ob die Wiederaufnahme des Dienstes erfolgreich ist, wählen Sie auf der Service-Dashboard-Seite den App Runner-Domänenwert aus. Das ist die URL für die Website Ihres Dienstes. Stellen Sie sicher, dass Ihre Webanwendung ordnungsgemäß ausgeführt wird.

App Runner API or AWS CLI

Um Ihren Dienst mithilfe der App Runner API anzuhalten oder AWS CLI rufen Sie die PauseServiceAPI-Aktion auf. Wenn der Aufruf eine erfolgreiche Antwort zurückgibt und ein Service-Objekt angezeigt wird`"Status": "OPERATION_IN_PROGRESS"`, beginnt App Runner, Ihren Dienst zu pausieren.

Rufen Sie die API-Aktion auf, um Ihren Dienst mit der App ResumeServiceRunner-API fortzusetzen oder AWS CLI. Wenn der Aufruf eine erfolgreiche Antwort zurückgibt und ein Service-Objekt angezeigt wird`"Status": "OPERATION_IN_PROGRESS"`, beginnt App Runner, Ihren Dienst wieder aufzunehmen.

Einen App Runner-Dienst löschen

Wenn Sie die Webanwendung beenden möchten, die in Ihrem AWS App Runner Dienst ausgeführt wird, können Sie den Dienst löschen. Durch das Löschen eines Dienstes wird der laufende Webdienst beendet, die zugrunde liegenden Ressourcen entfernt und die zugehörigen Daten werden gelöscht.

Möglicherweise möchten Sie einen App Runner-Dienst aus einem oder mehreren der folgenden Gründe löschen:

- Sie benötigen die Webanwendung nicht mehr — zum Beispiel, weil sie eingestellt wurde oder es sich um eine Entwicklungsversion handelt, die Sie nicht mehr verwenden.
- Sie haben das App Runner-Servicekontingent erreicht — Sie möchten in derselben Version einen neuen Dienst erstellen AWS-Region und haben das mit Ihrem Konto verknüpfte Kontingent erreicht. Weitere Informationen finden Sie unter the section called “App Runner-Ressourcenkontingente”.

- Überlegungen zur Sicherheit oder zum Datenschutz — Sie möchten, dass App Runner die Daten löscht, die es für Ihren Dienst speichert.

Pausieren und Löschen im Vergleich

Unterbrechen Sie Ihren App Runner-Dienst, um ihn vorübergehend zu deaktivieren. Nur Rechenressourcen werden beendet, und Ihre gespeicherten Daten (z. B. das Container-Image mit Ihrer Anwendungsversion) bleiben erhalten. Die Wiederaufnahme Ihres Dienstes ist schnell erledigt — Ihre Anwendung ist bereit, für neue Rechenressourcen bereitgestellt zu werden. Ihre App Runner-Domain bleibt dieselbe.

Löschen Sie Ihren App Runner-Dienst, um ihn dauerhaft zu entfernen. Ihre gespeicherten Daten werden gelöscht. Wenn Sie den Dienst neu erstellen müssen, muss App Runner Ihre Quelle erneut abrufen und sie auch erstellen, falls es sich um ein Code-Repository handelt. Ihre Webanwendung erhält eine neue App-Runner-Domain.

Was löscht App Runner?

Wenn Sie Ihren Dienst löschen, löscht App Runner einige zugehörige Elemente und andere nicht. Die folgenden Listen enthalten die Einzelheiten.

Elemente, die App Runner löscht:

- Container-Image — Eine Kopie des Images, das Sie bereitgestellt haben, oder des Images, das App Runner aus Ihrem Quellcode erstellt hat. Es wird in Amazon Elastic Container Registry (Amazon ECR) mithilfe interner Daten gespeichert AWS-Konten , die App Runner gehören.
- Dienstkonfiguration — Die Konfigurationseinstellungen, die mit Ihrem App Runner-Service verknüpft sind. Sie werden in Amazon DynamoDB mithilfe interner Dateien gespeichert AWS-Konten , die App Runner gehören.

Elemente, die App Runner nicht löscht:

- Verbindung — Möglicherweise besteht eine Verbindung, die mit Ihrem Dienst verknüpft ist. Eine App Runner-Verbindung ist eine separate Ressource, die möglicherweise von mehreren App Runner-Diensten gemeinsam genutzt wird. Wenn Sie die Verbindung nicht mehr benötigen, können Sie sie explizit löschen. Weitere Informationen finden Sie unter the section called “Verbindungen”.
- Benutzerdefinierte Domain-Zertifikate — Wenn Sie benutzerdefinierte Domains mit einem App Runner-Dienst verknüpfen, erstellt App Runner intern Zertifikate, die die Gültigkeit der Domain

nachverfolgen. Sie werden in AWS Certificate Manager (ACM) gespeichert. App Runner löscht das Zertifikat sieben Tage lang nicht, nachdem eine Domain von Ihrem Dienst getrennt wurde oder nachdem der Dienst gelöscht wurde. Weitere Informationen finden Sie unter the section called “Benutzerdefinierte Domainnamen”.

Löschen Sie Ihren Dienst

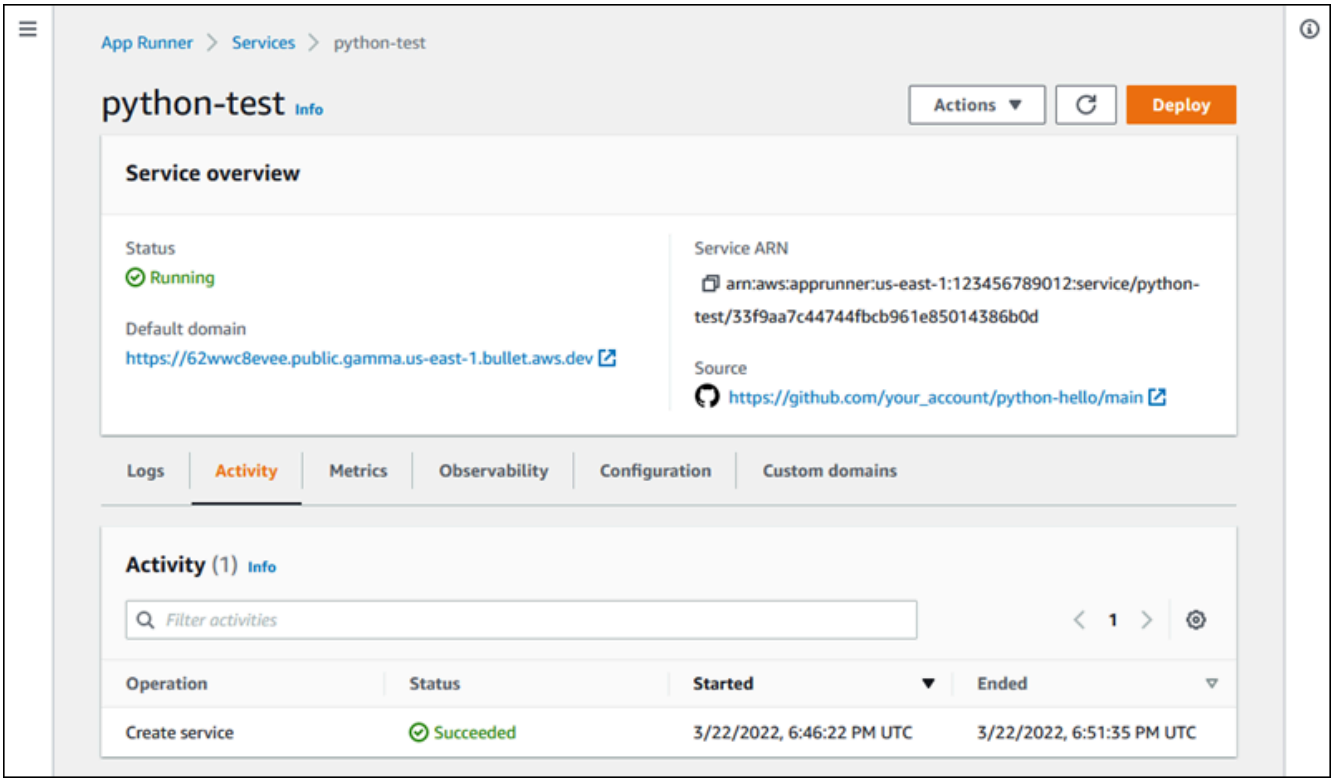
Löschen Sie Ihren App Runner-Dienst mit einer der folgenden Methoden:

App Runner console

Um Ihren Dienst mit der App Runner-Konsole zu löschen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie Aktionen und anschließend Löschen aus.

Über die Konsole gelangen Sie zur Seite Dienste. Der gelöschte Service zeigt den Status Vorgang wird erstellt. Anschließend verschwindet der Service aus der Liste. Ihr Dienst ist jetzt gelöscht.

App Runner API or AWS CLI

Um Ihren Dienst mit der App Runner API oder zu löschen AWS CLI, rufen Sie die DeleteServiceAPI-Aktion auf. Wenn der Aufruf eine erfolgreiche Antwort zurückgibt und ein Service-Objekt angezeigt wird`"Status": "OPERATION_IN_PROGRESS"`, beginnt App Runner mit dem Löschen Ihres Dienstes.

Referenzieren von Umgebungsvariablen

Mit App Runner können Sie Geheimnisse und Konfigurationen als Umgebungsvariablen in Ihrem Service referenzieren, wenn Sie einen Service erstellen oder aktualisieren.

Sie können auf nicht sensible Konfigurationsdaten wie Timeouts und Wiederholungszählungen im Klartext als Schlüssel-Wert-Paare verweisen. Die Konfigurationsdaten, auf die Sie im Klartext verweisen, sind nicht verschlüsselt und in den App Runner-Dienstkonfigurations- und Anwendungsprotokollen für andere sichtbar.

> **Note**
>
> Verweisen Sie aus Sicherheitsgründen in Ihrem App Runner-Dienst nicht auf sensible Daten im Klartext.

Vertrauliche Daten als Umgebungsvariablen referenzieren

App Runner unterstützt die sichere Referenzierung sensibler Daten als Umgebungsvariablen in Ihrem Service. Erwägen Sie, die sensiblen Daten, auf die Sie verweisen möchten, in unserem AWS Secrets ManagerAWS Systems Manager Parameter Store zu speichern. Anschließend können Sie sie in Ihrem Service über die App Runner-Konsole oder durch Aufrufen der API sicher als Umgebungsvariablen referenzieren. Dadurch wird die Verwaltung von Geheimnissen und Parametern effektiv von Ihrem Anwendungscode und Ihrer Dienstkonfiguration getrennt, wodurch die allgemeine Sicherheit Ihrer Anwendungen, die auf App Runner ausgeführt werden, verbessert wird.

> **Note**
>
> App Runner berechnet Ihnen keine Gebühren für die Referenzierung von Secrets Manager und SSM Parameter Store als Umgebungsvariablen. Sie zahlen jedoch den Standardpreis für die Nutzung von Secrets Manager und SSM Parameter Store.
> Weitere Informationen zu Preisen finden Sie unter:
>
> - AWS Secrets Manager — Preisgestaltung
> - AWS Preisgestaltung im SSM Parameter Store

Im Folgenden wird beschrieben, wie sensible Daten als Umgebungsvariablen referenziert werden:

1. Speichern Sie vertrauliche Daten wie API-Schlüssel, Datenbankanmeldedaten, Datenbankverbindungsparameter oder Anwendungsversionen als Geheimnisse oder Parameter in einem AWS Secrets Manager oder im AWS Systems Manager Parameter Store.
2. Aktualisieren Sie die IAM-Richtlinie Ihrer Instanzrolle, sodass App Runner auf die in Secrets Manager und SSM Parameter Store gespeicherten Geheimnisse und Parameter zugreifen kann. Weitere Informationen finden Sie unter -Berechtigungen.
3. Verweisen Sie auf sichere Weise auf die Geheimnisse und Parameter als Umgebungsvariablen, indem Sie ihnen einen Namen zuweisen und ihren Amazon-Ressourcennamen (ARN) angeben. Sie können Umgebungsvariablen hinzufügen, wenn Sie einen Service erstellen oder die Konfiguration eines Dienstes aktualisieren. Sie können eine der folgenden Optionen verwenden, um Umgebungsvariablen hinzuzufügen:
   - App Runner-Konsole
   - App Runner-API
   - `apprunner.yaml` Konfigurationsdatei

   > **Note**
   >
   > Sie können einer Umgebungsvariablen keinen Namen zuweisen`PORT`, wenn Sie Ihren App Runner-Dienst erstellen oder aktualisieren. Es ist eine reservierte Umgebungsvariable für den App Runner-Dienst.

   Weitere Informationen zum Referenzieren von Geheimnissen und Parametern finden Sie unter Umgebungsvariablen verwalten.

> **Note**
>
> Da App Runner nur den Verweis auf Secret und Parameter speichert ARNs, sind die sensiblen Daten in der App Runner-Dienstkonfiguration und in den Anwendungsprotokollen für andere nicht sichtbar.

Überlegungen

- Stellen Sie sicher, dass Sie Ihre Instanzrolle mit den entsprechenden Berechtigungen für den Zugriff auf die Geheimnisse und Parameter im AWS Secrets Manager oder im AWS Systems Manager Parameter Store aktualisieren. Weitere Informationen finden Sie unter -Berechtigungen.
- Stellen Sie sicher, dass sich der AWS Systems Manager Parameter Store im selben AWS-Konto Verzeichnis befindet wie der Dienst, den Sie starten oder aktualisieren möchten. Derzeit können Sie nicht kontenübergreifend auf SSM-Parameter des Parameterspeichers verweisen.
- Wenn die Secrets und Parameterwerte rotiert oder geändert werden, werden sie in Ihrem App Runner-Dienst nicht automatisch aktualisiert. Stellen Sie Ihren App Runner-Dienst erneut bereit, da App Runner während der Bereitstellung nur Geheimnisse und Parameter abruft.
- Sie haben auch die Möglichkeit, direkt über das SDK in Ihrem App Runner-Dienst aufzurufen AWS Secrets Manager und AWS Systems Manager Parameter zu speichern.
- Um Fehler zu vermeiden, stellen Sie Folgendes sicher, wenn Sie sie als Umgebungsvariablen referenzieren:
  - Sie geben den richtigen ARN des Geheimnisses an.
  - Sie geben den richtigen Namen oder ARN des Parameters an.

Berechtigungen

Um die Referenzierung von Geheimnissen und Parametern zu ermöglichen, die im AWS Secrets Manager oder SSM Parameter Store gespeichert sind, fügen Sie der IAM-Richtlinie Ihrer Instanzrolle entsprechende Berechtigungen für den Zugriff auf Secrets Manager und SSM Parameter Store hinzu.

> **Note**
>
> App Runner kann ohne Ihre Zustimmung nicht auf Ressourcen in Ihrem Konto zugreifen. Sie erteilen die Erlaubnis, indem Sie Ihre IAM-Richtlinie aktualisieren.

Sie können die folgenden Richtlinienvorlagen verwenden, um Ihre Instanzrolle in der IAM-Konsole zu aktualisieren. Sie können diese Richtlinienvorlagen an Ihre spezifischen Anforderungen anpassen. Weitere Informationen zum Aktualisieren einer Instanzrolle finden Sie unter Ändern einer Rolle im IAM-Benutzerhandbuch.

> **Note**
>
> Sie können beim Erstellen der Umgebungsvariablen auch die folgenden Vorlagen aus der App Runner-Konsole kopieren.

Kopieren Sie die folgende Vorlage in Ihre Instanzrolle, um Berechtigungen zum Verweisen auf Geheimnisse hinzuzufügen AWS Secrets Manager.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "kms:Decrypt*"
            ],
            "Resource": [

 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
                "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
            ]
        }
    ]
}
```

Kopieren Sie die folgende Vorlage in Ihre Instanzrolle, um Berechtigungen für Referenzparameter aus dem AWS Systems ManagerParameter Store hinzuzufügen.

JSON

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

```
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter_name>"
            ]
        }
    ]
}
```

Verwaltung Ihrer Umgebungsvariablen

Verwalten Sie die Umgebungsvariablen für Ihren App Runner-Dienst mit einer der folgenden Methoden:

- the section called “App Runner-Konsole”
- the section called “App Runner API oder AWS CLI”

App Runner-Konsole

Wenn Sie in der App Runner-Konsole einen Dienst erstellen oder aktualisieren, können Sie Umgebungsvariablen hinzufügen.

Umgebungsvariable hinzufügen

Um eine Umgebungsvariable hinzuzufügen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste Regionen Ihre aus AWS-Region.
2. Je nachdem, ob Sie einen Dienst erstellen oder aktualisieren, führen Sie einen der folgenden Schritte aus:
   - Wenn Sie einen neuen Dienst erstellen, wählen Sie App Runner-Dienst erstellen und gehen Sie zu Dienst konfigurieren.
   - Wenn Sie einen vorhandenen Dienst aktualisieren, wählen Sie den Dienst aus, den Sie aktualisieren möchten, und wechseln Sie zur Registerkarte Konfiguration des Dienstes.
3. Gehen Sie unter Diensteinstellungen zu Umgebungsvariablen — optional.
4. Wählen Sie je nach Anforderung eine der folgenden Optionen aus:

- Wählen Sie Klartext aus der Quelle der Umgebungsvariablen aus und geben Sie die Schlüssel-Wert-Paare unter Name der Umgebungsvariablen bzw. Wert der Umgebungsvariablen ein.

  > **Note**
  >
  > Wählen Sie „Nur Text“, wenn Sie auf nicht sensible Daten verweisen möchten. Diese Daten sind nicht verschlüsselt und für andere in der App Runner-Dienstkonfiguration und in den Anwendungsprotokollen sichtbar.

- Wählen Sie Secrets Manager aus der Umgebungsvariablenquelle aus, um auf das Geheimnis zu verweisen, das AWS Secrets Manager als Umgebungsvariable in Ihrem Service gespeichert ist. Geben Sie den Namen der Umgebungsvariablen und den Amazon-Ressourcennamen (ARN) des Geheimnisses, auf das Sie verweisen, jeweils unter Umgebungsvariablenname und Umgebungsvariablenwert ein.
- Wählen Sie SSM Parameter Store aus der Umgebungsvariablenquelle aus, um auf den im SSM Parameter Store gespeicherten Parameter als Umgebungsvariable in Ihrem Service zu verweisen. Geben Sie den Namen der Umgebungsvariablen und den ARN des Parameters, auf den Sie verweisen, jeweils unter Umgebungsvariablenname und Umgebungsvariablenwert ein.

  > **Note**
  >
  > - Sie können einer Umgebungsvariablen keinen Namen zuweisen`PORT`, wenn Sie Ihren App Runner-Dienst erstellen oder aktualisieren. Es ist eine reservierte Umgebungsvariable für den App Runner-Dienst.
  > - Wenn der SSM Parameter Store-Parameter mit dem Service AWS-Region identisch ist, den Sie starten möchten, können Sie den vollständigen Amazon-Ressourcennamen (ARN) oder den Namen des Parameters angeben. Wenn sich der Parameter in einer anderen Region befindet, müssen Sie den vollständigen ARN angeben.
  > - Stellen Sie sicher, dass sich der Parameter, auf den Sie verweisen, in demselben Konto befindet wie der Dienst, den Sie starten oder aktualisieren. Derzeit können Sie nicht kontenübergreifend auf den SSM Parameter Store-Parameter verweisen.

5. Wählen Sie Umgebungsvariable hinzufügen, um auf eine andere Umgebungsvariable zu verweisen.
6. Erweitern Sie IAM-Richtlinienvorlagen, um die für den SSM-Parameterspeicher bereitgestellten IAM-Richtlinienvorlagen anzuzeigen AWS Secrets Manager und zu kopieren. Sie müssen dies nur tun, wenn Sie die IAM-Richtlinie Ihrer Instanzrolle noch nicht mit den erforderlichen Berechtigungen aktualisiert haben. Weitere Informationen finden Sie unter -Berechtigungen.

Umgebungsvariable wird entfernt

Bevor Sie eine Umgebungsvariable löschen, stellen Sie sicher, dass Ihr Anwendungscode aktualisiert wurde, sodass er dieselbe wiedergibt. Wenn der Anwendungscode nicht aktualisiert wird, schlägt Ihr App Runner-Dienst möglicherweise fehl.

Um Umgebungsvariablen zu entfernen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste Regionen Ihre aus AWS-Region.
2. Gehen Sie zur Registerkarte Konfiguration des Dienstes, den Sie aktualisieren möchten.
3. Gehen Sie unter Diensteinstellungen zu Umgebungsvariablen — optional.
4. Wählen Sie neben der Umgebungsvariablen, die Sie entfernen möchten, die Option Entfernen aus. Sie erhalten eine Nachricht zur Bestätigung des Löschvorgangs.
5. Wählen Sie Löschen aus.

App Runner API oder AWS CLI

Sie können auf sensible Daten verweisen, die in Secrets Manager und SSM Parameter Store gespeichert sind, indem Sie sie als Umgebungsvariablen in Ihrem Service hinzufügen.

> **Note**
>
> Aktualisieren Sie die IAM-Richtlinie Ihrer Instanzrolle, sodass App Runner auf Geheimnisse und Parameter zugreifen kann, die in Secrets Manager und SSM Parameter Store gespeichert sind. Weitere Informationen finden Sie unter -Berechtigungen.

Um Geheimnisse und Konfigurationen als Umgebungsvariablen zu referenzieren

1. Erstellen Sie ein Geheimnis oder eine Konfiguration im Secrets Manager oder SSM Parameter Store.

   Die folgenden Beispiele zeigen, wie Sie mithilfe des SSM-Parameterspeichers ein Geheimnis und einen Parameter erstellen.

   Example Ein Geheimnis erstellen — Anfrage

   Das folgende Beispiel zeigt, wie ein Geheimnis erstellt wird, das die Datenbankanmeldedaten darstellt.

   ```
   aws secretsmanager create-secret \
   –name DevRdsCredentials \
   –description "Rds credentials for development account." \
   –secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
   ```

   Example Ein Geheimnis erstellen — Antwort

   ```
   arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
   ```

   Example Konfiguration erstellen — Anfrage

   Das folgende Beispiel zeigt, wie ein Parameter erstellt wird, der die RDS-Verbindungszeichenfolge darstellt.

   ```
   aws systemsmanager put-parameter \
   –name DevRdsConnectionString \
   –value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
   –type "String" \
   –description "Rds connection string for development account."
   ```

   Example Konfiguration erstellen — Antwort

   ```
   arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
   ```

2. Verweisen Sie auf die Geheimnisse und Konfigurationen, die in Secrets Manager und SSM Parameter Store gespeichert sind, indem Sie sie als Umgebungsvariablen hinzufügen. Sie

können Umgebungsvariablen hinzufügen, wenn Sie Ihren App Runner-Dienst erstellen oder aktualisieren.

Die folgenden Beispiele zeigen, wie Sie Geheimnisse und Konfigurationen als Umgebungsvariablen in einem codebasierten und einem imagebasierten App Runner-Dienst referenzieren.

Example Input.json-Datei für den bildbasierten App Runner-Dienst

```
{
  "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 Imagebasierter App Runner-Dienst — Anfrage

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

Example Bildbasierter App Runner-Dienst — Antwort

```
{
```

```
...
      "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 Input.json-Datei für den codebasierten App Runner-Dienst

```
{
  "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 Codebasierter App Runner-Dienst — Anfrage

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

Example Codebasierter App Runner-Dienst — Antwort

```
{
...
   "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. Das `apprunner.yaml` Modell wurde aktualisiert, um die hinzugefügten Geheimnisse widerzuspiegeln.

   Das Folgende ist ein Beispiel für das aktualisierte `apprunner.yaml` Modell.

   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"
```

Networking mit App Runner

In diesem Kapitel werden Netzwerkkonfigurationen für Ihre AWS App Runner Dienste beschrieben.

In diesem Kapitel erfahren Sie Folgendes:

- So konfigurieren Sie Ihren eingehenden Datenverkehr für private und öffentliche Endgeräte. Weitere Informationen finden Sie unter Netzwerkkonfigurationen für eingehenden Datenverkehr einrichten.
- So konfigurieren Sie Ihren ausgehenden Datenverkehr für den Zugriff auf andere Anwendungen, die in einer Amazon VPC ausgeführt werden. Weitere Informationen finden Sie unter VPC-Zugriff für ausgehenden Datenverkehr aktivieren.

> **Note**
>
> App Runner unterstützt derzeit den Dual-Stack-Adresstyp (IPv4 und IPv6) nur für eingehenden öffentlichen Datenverkehr. Es wird nur IPv4 für ausgehenden Verkehr und privaten eingehenden Verkehr unterstützt.

Themen

Terminologie

Um zu erfahren, wie Sie Ihren Netzwerkverkehr an Ihre Bedürfnisse anpassen können, sollten wir die folgenden Begriffe verstehen, die in diesem Kapitel verwendet werden.

Allgemeine Bedingungen

Um zu wissen, was für die Verknüpfung mit einer Amazon Virtual Private Cloud (VPC) erforderlich ist, sollten wir die folgenden Begriffe verstehen:

- VPC: Eine Amazon VPC ist ein logisch isoliertes virtuelles Netzwerk, mit dem Sie die vollständige Kontrolle über Ihre virtuelle Netzwerkumgebung haben, einschließlich Ressourcenplatzierung, Konnektivität und Sicherheit. Es ist ein virtuelles Netzwerk, das einem herkömmlichen Netzwerk, das Sie in Ihrem eigenen Rechenzentrum betreiben würden, sehr ähnlich ist.
- VPC-Schnittstellenendpunkt: Der VPC-Schnittstellenendpunkt, eine AWS PrivateLink Ressource, verbindet eine VPC mit einem Endpunktdienst. Erstellen Sie einen VPC-Schnittstellenendpunkt, um Datenverkehr an Endpunktdienste zu senden, die einen Network Load Balancer zur Verteilung des Datenverkehrs verwenden. Der für den Endpunkt-Service bestimmte Datenverkehr wird mithilfe von DNS aufgelöst.
- Regionen: Jede Region ist ein separates geografisches Gebiet, in dem Sie einen App Runner-Dienst hosten können.
- Availability Zones: Eine Availability Zone ist ein isolierter Standort innerhalb einer AWS Region. Es handelt sich um ein oder mehrere diskrete Rechenzentren mit redundanter Stromversorgung, Vernetzung und Konnektivität. Availability Zones helfen Ihnen dabei, Produktionsanwendungen hochverfügbar, fehlertolerant und skalierbar zu machen.
- Subnetze: Ein Subnetz ist ein Bereich von IP-Adressen in Ihrer VPC. Ein Subnetz muss sich in einer einzigen Availability Zone befinden. Sie können eine AWS Ressource in einem bestimmten Subnetz starten. Verwenden Sie öffentliche Subnetze für Ressourcen, die mit dem Internet verbunden sein müssen, und private Subnetze für Ressourcen, die nicht mit dem Internet verbunden sein werden.
- Sicherheitsgruppen: Eine Sicherheitsgruppe kontrolliert den Verkehr, der die Ressourcen erreichen und verlassen darf, denen sie zugeordnet ist. Sicherheitsgruppen bieten eine zusätzliche Sicherheitsebene zum Schutz der AWS Ressourcen in jedem Subnetz, sodass Sie mehr Kontrolle über Ihren Netzwerkverkehr haben. Wenn Sie eine VPC erstellen, verfügt diese über eine Standardsicherheitsgruppe. Sie können für jede VPC zusätzliche Sicherheitsgruppen erstellen. Sie können eine Sicherheitsgruppe nur Ressourcen innerhalb der VPC zuordnen, für die sie erstellt wurde.
- Dual-Stack: Ein Dual-Stack ist ein Adresstyp, der Netzwerkverkehr sowohl von Endpunkten als auch von IPv4 IPv6 Endpunkten unterstützt.

Spezifischer Begriff für die Konfiguration von ausgehendem Verkehr

VPC-Anschluss

Ein VPC Connector ist eine App Runner-Ressource, die es dem App Runner-Service ermöglicht, auf Anwendungen zuzugreifen, die in einer privaten Amazon VPC ausgeführt werden.

Spezifische Bedingungen für die Konfiguration von eingehendem Datenverkehr

Um zu erfahren, wie Sie Ihre Dienste nur von einer Amazon VPC aus privat zugänglich machen können, sollten wir uns mit den folgenden Begriffen vertraut machen:

- VPC Ingress Connection: VPC Ingress Connection ist eine App Runner-Ressource, die einen App Runner-Endpunkt für eingehenden Datenverkehr bereitstellt. App Runner weist die VPC-Ingress-Verbindungsressource hinter den Kulissen zu, wenn Sie in der App Runner-Konsole den privaten Endpunkt für Ihren eingehenden Datenverkehr auswählen. Die Ressource VPC Ingress Connection verbindet Ihren App Runner-Service mit dem VPC-Schnittstellenendpunkt der Amazon VPC.

  > **Note**
  >
  > Wenn Sie die App Runner API verwenden, wird die VPC-Ingress-Verbindungsressource nicht automatisch erstellt.

- Privater Endpunkt: Der private Endpunkt ist eine App Runner-Konsolenoption, die Sie auswählen, um den eingehenden Netzwerkverkehr so zu konfigurieren, dass er nur von einer Amazon VPC aus zugänglich ist.

Netzwerkkonfigurationen für eingehenden Verkehr einrichten

Sie können Ihren Dienst so konfigurieren, dass er eingehenden Datenverkehr von privaten oder öffentlichen Endpunkten empfängt.

Ein öffentlicher Endpunkt ist die Standardkonfiguration. Es öffnet Ihren Dienst für jeden eingehenden Verkehr aus dem öffentlichen Internet. Es bietet Ihnen auch die Flexibilität, für Ihren Dienst zwischen dem Adresstyp Internet Protocol Version 4 (IPv4) oder Dual-Stack (IPv4 und IPv6) zu wählen.

Ein privater Endpunkt ermöglicht nur dem Datenverkehr von einer Amazon VPC den Zugriff auf Ihren App Runner-Service. Dies wird erreicht, indem Sie einen VPC-Schnittstellenendpunkt, eine AWS PrivateLink Ressource, für Ihren App Runner-Dienst einrichten. Dadurch wird eine private Verbindung zwischen der Amazon VPC und Ihrem App Runner-Service hergestellt.

> **Note**
>
> App Runner unterstützt derzeit den Dual-Stack-Adresstyp (IPv4 und IPv6) nur für öffentliche Endgeräte. Wird nur IPv4 für private Endgeräte unterstützt.

Die folgenden Themen werden im Rahmen der Einrichtung Ihrer Netzwerkkonfigurationen für eingehenden Datenverkehr behandelt:

- So konfigurieren Sie Ihren eingehenden Datenverkehr so, dass Ihr Service nur von einer Amazon VPC aus privat verfügbar ist. Weitere Informationen finden Sie unter Privaten Endpunkt für eingehenden Datenverkehr aktivieren.
- So konfigurieren Sie Ihren Dienst für den Empfang von Internetverkehr vom Dual-Stack-Adresstyp. Weitere Informationen finden Sie unter Dual-Stack-Aktivierung für eingehenden öffentlichen Datenverkehr.

Überschriften

Mit App Runner können Sie auf die ursprüngliche Quelle IPv4 und die IPv6 Adressen des Datenverkehrs zugreifen, der in Ihre Anwendung eingeht. Die ursprünglichen Quell-IP-Adressen werden beibehalten, indem ihnen der `X-Forwarded-For` Anforderungsheader zugewiesen wird. Auf diese Weise können Ihre Anwendungen bei Bedarf die ursprünglichen Quell-IP-Adressen abrufen.

> **Note**
>
> Wenn Ihr Dienst für die Verwendung eines privaten Endpunkts konfiguriert ist, kann der `X-Forwarded-For` Anforderungsheader nicht für den Zugriff auf die ursprünglichen Quell-IP-Adressen verwendet werden. Bei Verwendung werden falsche Werte abgerufen.

Privaten Endpunkt für eingehenden Datenverkehr aktivieren

Wenn Sie einen AWS App Runner Service erstellen, ist der Service standardmäßig über das Internet zugänglich. Sie können Ihren App Runner-Service jedoch auch privat machen und nur von einer Amazon Virtual Private Cloud (Amazon VPC) aus zugänglich machen.

Wenn Ihr App Runner-Service privat ist, haben Sie die vollständige Kontrolle über den eingehenden Datenverkehr und fügen so eine zusätzliche Sicherheitsebene hinzu. Dies ist in einer Vielzahl

von Anwendungsfällen hilfreich, z. B. bei der Ausführung interner APIs, unternehmensinterner Webanwendungen oder bei Anwendungen, die sich noch in der Entwicklung befinden und ein höheres Maß an Datenschutz und Sicherheit erfordern oder bestimmte Compliance-Anforderungen erfüllen müssen.

> **Note**
>
> Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln.
> Weitere Informationen zur Infrastruktursicherheit und zu Sicherheitsgruppen, einschließlich bewährter Methoden, finden Sie in den folgenden Themen im Amazon VPC-Benutzerhandbuch: Steuern des Netzwerkverkehrs und Steuern des Datenverkehrs zu Ihren AWS-Ressourcen mithilfe von Sicherheitsgruppen.

Wenn Ihr App Runner-Service privat ist, können Sie von einer Amazon VPC aus auf Ihren Service zugreifen. Ein Internet-Gateway, ein NAT-Gerät oder eine VPN-Verbindung sind nicht erforderlich.

> **Note**
>
> App Runner unterstützt derzeit den Dual-Stack-Adresstyp (IPv4 und IPv6) nur für eingehenden öffentlichen Datenverkehr. Es wird nur IPv4 für ausgehenden Verkehr und privaten eingehenden Verkehr unterstützt.

Überlegungen

- Bevor Sie einen VPC-Schnittstellenendpunkt für App Runner einrichten, lesen Sie die Überlegungen im AWS PrivateLink Handbuch.
- VPC-Endpunktrichtlinien werden für App Runner nicht unterstützt. Standardmäßig ist der vollständige Zugriff auf App Runner über den VPC-Schnittstellenendpunkt zulässig. Alternativ können Sie den Endpunkt-Netzwerkschnittstellen eine Sicherheitsgruppe zuordnen, um den Datenverkehr zu App Runner über den VPC-Schnittstellenendpunkt zu steuern.

- Wenn Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs benötigt, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln.
- Nachdem Sie einen privaten Endpunkt aktiviert haben, ist Ihr Service nur von Ihrer VPC aus zugänglich und kann nicht über das Internet aufgerufen werden.
- Für eine höhere Verfügbarkeit wird empfohlen, mindestens zwei Subnetze in der Availability Zone auszuwählen, die sich für den VPC-Schnittstellenendpunkt unterscheiden. Es wird nicht empfohlen, nur ein Subnetz zu verwenden.
- Sie können denselben VPC-Schnittstellenendpunkt verwenden, um auf mehrere App Runner-Dienste in einer VPC zuzugreifen.

Informationen zu den in diesem Abschnitt verwendeten Begriffen finden Sie unter Terminologie.

Berechtigungen

Im Folgenden finden Sie eine Liste der Berechtigungen, die zur Aktivierung des privaten Endpunkts erforderlich sind:

- ec2: CreateTags
- ec2: CreateVpcEndpoint
- ec2: ModifyVpcEndpoint
- ec2: DeleteVpcEndpoints
- ec2: DescribeSubnets
- ec2: DescribeVpcEndpoints
- ec2: DescribeVpcs

Endpunkt der VPC-Schnittstelle

Ein VPC-Schnittstellenendpunkt ist eine AWS PrivateLinkRessource, die eine Amazon-VPC mit einem Endpunktservice verbindet. Sie können angeben, in welcher Amazon VPC Ihr App Runner-Service zugänglich sein soll, indem Sie einen VPC-Schnittstellenendpunkt übergeben. Um einen VPC-Schnittstellenendpunkt zu erstellen, geben Sie Folgendes an:

- Die Amazon VPC zur Aktivierung der Konnektivität.
- Sicherheitsgruppen hinzufügen. Standardmäßig ist dem VPC-Schnittstellenendpunkt eine Sicherheitsgruppe zugewiesen. Sie können eine benutzerdefinierte Sicherheitsgruppe zuordnen, um den eingehenden Netzwerkverkehr weiter zu kontrollieren.
- Fügen Sie Subnetze hinzu. Um eine höhere Verfügbarkeit zu gewährleisten, wird empfohlen, mindestens zwei Subnetze für jede Availability Zone auszuwählen, von der aus Sie auf den App Runner-Dienst zugreifen. In jedem Subnetz, das Sie für den VPC-Schnittstellenendpunkt aktivieren, wird ein Netzwerkschnittstellen-Endpunkt erstellt. Dabei handelt es sich um vom Anforderer verwaltete Netzwerkschnittstellen, die als Einstiegspunkt für den Datenverkehr dienen, der für App Runner bestimmt ist. Eine vom Anforderer verwaltete Netzwerkschnittstelle ist eine Netzwerkschnittstelle, die ein AWS -Service in Ihrer VPC für Sie erstellt.
- Wenn Sie die API verwenden, fügen Sie den App Runner VPC-Schnittstellenendpunkt `Servicename` hinzu. Zum Beispiel

```
com.amazonaws.region.apprunner.requests
```

Sie können einen VPC-Schnittstellenendpunkt mit einem der folgenden AWS Dienste erstellen:

- App Runner-Konsole. Weitere Informationen finden Sie unter Private Endpoints verwalten.
- Amazon VPC-Konsole oder API und AWS Command Line Interface (AWS CLI). Weitere Informationen finden Sie AWS PrivateLink im AWS PrivateLink Handbuch unter Access AWS-Services through.

> **Note**
>
> Ihnen wird jeder VPC-Schnittstellen-Endpunkt, den Sie verwenden, auf der Grundlage der AWS PrivateLink Preisgestaltung in Rechnung gestellt. Aus Kostengründen können Sie daher denselben VPC-Schnittstellenendpunkt verwenden, um auf mehrere App Runner-Dienste innerhalb einer VPC zuzugreifen. Für eine bessere Isolierung sollten Sie jedoch erwägen, jedem Ihrer App Runner-Dienste einen anderen VPC-Schnittstellenendpunkt zuzuordnen.

VPC-Ingress-Verbindung

Eine VPC-Ingress-Verbindung ist eine App Runner-Ressource, die einen App Runner-Endpunkt für eingehenden Datenverkehr angibt. App Runner weist die VPC-Ingress-Verbindungsressource hinter den Kulissen zu, wenn Sie in der App Runner-Konsole den privaten Endpunkt für Ihren eingehenden Datenverkehr auswählen. Wählen Sie diese Option, damit nur Traffic von einer Amazon VPC auf Ihren App Runner-Service zugreifen kann. Die Ressource VPC Ingress Connection verbindet Ihren App Runner-Service mit dem VPC-Schnittstellenendpunkt der Amazon VPC. Sie können eine VPC-Ingress-Verbindungsressource nur erstellen, wenn Sie die API-Operationen verwenden, um die Netzwerkeinstellungen für eingehenden Datenverkehr zu konfigurieren. Weitere Informationen zum Erstellen einer VPC-Ingress-Verbindungsressource finden Sie CreateVpcIngressConnectionin der AWS App Runner API-Referenz.

> **Note**
>
> Eine VPC-Ingress-Verbindungsressource des App Runner kann eine Verbindung zu einem VPC-Schnittstellenendpunkt der Amazon VPC herstellen. Außerdem können Sie für jeden App Runner-Dienst nur eine VPC-Ingress-Verbindungsressource erstellen.

Privater Endpunkt

Private Endpoint ist eine App Runner-Konsolenoption, die Sie wählen können, wenn Sie nur eingehenden Datenverkehr von einer Amazon VPC empfangen möchten. Wenn Sie in der App Runner-Konsole die Option Privater Endpunkt auswählen, haben Sie die Möglichkeit, Ihren Service mit einer VPC zu verbinden, indem Sie dessen VPC-Schnittstellenendpunkt konfigurieren. Im Hintergrund weist App Runner dem von Ihnen konfigurierten VPC-Schnittstellenendpunkt eine VPC-Ingress-Verbindungsressource zu.

> **Note**
>
> Für den privaten Endpunkt wird nur IPv4 Netzwerkverkehr unterstützt.

Übersicht

Machen Sie Ihren Service privat, indem Sie nur Traffic von einer Amazon VPC auf Ihren App Runner-Service zugreifen lassen. Um dies zu erreichen, erstellen Sie mit App Runner oder Amazon

VPC einen VPC-Schnittstellenendpunkt für die ausgewählte Amazon VPC. In der App Runner-Konsole erstellen Sie einen VPC-Schnittstellenendpunkt, wenn Sie den privaten Endpunkt für den eingehenden Datenverkehr aktivieren. App Runner erstellt dann automatisch eine VPC-Ingress-Verbindungsressource und stellt eine Verbindung zum VPC-Schnittstellenendpunkt und Ihrem App Runner-Dienst her. Dadurch wird eine private Dienstverbindung erstellt, die sicherstellt, dass nur Traffic von der ausgewählten VPC auf Ihren App Runner-Dienst zugreifen kann.

Private Endgeräte verwalten

Verwalten Sie den privaten Endpunkt für den eingehenden Datenverkehr mit einer der folgenden Methoden:

- the section called “App Runner-Konsole”
- the section called “App Runner API oder AWS CLI”

> **Note**
>
> Wenn Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs benötigt, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln. Weitere Informationen zur Infrastruktursicherheit und zu Sicherheitsgruppen, einschließlich bewährter Methoden, finden Sie in den folgenden Themen im Amazon VPC-Benutzerhandbuch: Steuern des Netzwerkverkehrs und Steuern des Datenverkehrs zu Ihren AWS-Ressourcen mithilfe von Sicherheitsgruppen.

App Runner-Konsole

Wenn Sie einen Dienst mit der App Runner-Konsole erstellen oder seine Konfiguration später aktualisieren, können Sie wählen, ob Sie den eingehenden Datenverkehr konfigurieren möchten.

Wählen Sie eine der folgenden Optionen, um Ihren eingehenden Datenverkehr zu konfigurieren.

- Öffentlicher Endpunkt: Um Ihren Dienst für alle Dienste über das Internet zugänglich zu machen. Standardmäßig ist der öffentliche Endpunkt ausgewählt.

- Privater Endpunkt: Um Ihren App Runner-Service nur von einer Amazon VPC aus zugänglich zu machen.

> **Note**
>
> Derzeit unterstützt App Runner IPv6 nur öffentliche Endpunkte. IPv6 Endpunkte werden für App Runner-Services, die in einer Amazon Virtual Private Cloud (Amazon VPC) gehostet werden, nicht unterstützt. Wenn Sie einen Service, der einen öffentlichen Dual-Stack-Endpunkt verwendet, auf einen privaten Endpunkt aktualisieren, unterstützt Ihr App Runner-Service standardmäßig nur Datenverkehr von IPv4 Endpunkten und empfängt keinen Datenverkehr von Endpunkten. IPv6

Privaten Endpunkt aktivieren

Aktivieren Sie einen privaten Endpunkt, indem Sie ihn dem VPC-Schnittstellenendpunkt der Amazon VPC zuordnen, auf die Sie zugreifen möchten. Sie können entweder einen neuen VPC-Schnittstellenendpunkt erstellen oder einen vorhandenen auswählen.

So erstellen Sie einen VPC-Schnittstellenendpunkt

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie unter Dienst konfigurieren zum Abschnitt Netzwerk.
3. Wählen Sie Private Endpoint für Eingehenden Netzwerkverkehr. Optionen zum Herstellen einer Verbindung zu einem VCP über den VPC-Schnittstellenendpunkt werden geöffnet.
4. Wählen Sie Neuen Endpunkt erstellen aus. Das Dialogfeld Neuen VPC-Schnittstellenendpunkt erstellen wird geöffnet.
5. Geben Sie einen Namen für den VPC-Schnittstellenendpunkt ein.
6. Wählen Sie den erforderlichen VPC-Schnittstellenendpunkt aus der verfügbaren Dropdownliste aus.
7. Wählen Sie eine Sicherheitsgruppe aus der Drop-down-Liste aus. Das Hinzufügen von Sicherheitsgruppen bietet eine zusätzliche Sicherheitsebene für den VPC-Schnittstellenendpunkt. Es wird empfohlen, zwei oder mehr Sicherheitsgruppen auszuwählen. Wenn Sie keine Sicherheitsgruppe auswählen, weist App Runner dem VPC-Schnittstellenendpunkt eine Standardsicherheitsgruppe zu. Stellen Sie sicher, dass die

Sicherheitsgruppenregeln nicht die Ressourcen blockieren, die mit Ihrem App Runner-Dienst kommunizieren möchten. Die Sicherheitsgruppenregeln müssen Ressourcen zulassen, die mit Ihrem App Runner-Dienst interagieren.

> **Note**
>
> Wenn Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs benötigt, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln.
> Weitere Informationen zur Infrastruktursicherheit und zu Sicherheitsgruppen, einschließlich bewährter Methoden, finden Sie in den folgenden Themen im Amazon VPC-Benutzerhandbuch: Steuern des Netzwerkverkehrs und Steuern des Datenverkehrs zu Ihren AWS-Ressourcen mithilfe von Sicherheitsgruppen.

8. Wählen Sie die erforderlichen Subnetze aus der Drop-down-Liste aus. Es wird empfohlen, mindestens zwei Subnetze für jede Availability Zone auszuwählen, von der aus Sie auf den App Runner-Dienst zugreifen.
9. (Optional) Wählen Sie Neues Tag hinzufügen und geben Sie den Tag-Schlüssel und den Tag-Wert ein.
10. Wählen Sie Create (Erstellen) aus. Die Seite Dienst konfigurieren wird geöffnet und in der oberen Leiste wird die Meldung angezeigt, dass der VPC-Schnittstellenendpunkt erfolgreich erstellt wurde.

So wählen Sie einen vorhandenen VPC-Schnittstellenendpunkt aus

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie unter Dienst konfigurieren zum Abschnitt Netzwerk.
3. Wählen Sie Private Endpoint für Eingehenden Netzwerkverkehr. Optionen zum Herstellen einer Verbindung mit einer VPC über den VPC-Schnittstellenendpunkt werden geöffnet. Eine Liste der verfügbaren VPC-Schnittstellenendpunkte wird angezeigt.

4. Wählen Sie den erforderlichen VPC-Schnittstellenendpunkt aus, der unter VPC-Schnittstellenendpunkte aufgeführt ist.
5. Wählen Sie Weiter, um Ihren Service zu erstellen. App Runner aktiviert den privaten Endpunkt.

> **Note**
>
> Nachdem Ihr Service erstellt wurde, können Sie bei Bedarf die Sicherheitsgruppen und Subnetze bearbeiten, die dem VPC-Schnittstellenendpunkt zugeordnet sind.

Um die Details des privaten Endpunkts zu überprüfen, gehen Sie zu Ihrem Service und erweitern Sie den Bereich Netzwerk auf der Registerkarte Konfiguration. Es zeigt Details zur VPC und zum VPC-Schnittstellenendpunkt, der dem privaten Endpunkt zugeordnet ist.

VPC-Schnittstellenendpunkt aktualisieren

Nachdem Ihr App Runner-Dienst erstellt wurde, können Sie den VPC-Schnittstellenendpunkt bearbeiten, der dem privaten Endpunkt zugeordnet ist.

> **Note**
>
> Sie können den Endpunktnamen und die VPC-Felder nicht aktualisieren.

So aktualisieren Sie den VPC-Schnittstellenendpunkt

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie zu Ihrem Dienst und wählen Sie im linken Bereich Netzwerkkonfigurationen aus.
3. Wählen Sie Eingehender Verkehr, um die VPC-Schnittstellenendpunkte anzuzeigen, die den jeweiligen Diensten zugeordnet sind.
4. Wählen Sie den VPC-Schnittstellen-Endpunkt aus, den Sie bearbeiten möchten.
5. Wählen Sie Edit (Bearbeiten) aus. Das Dialogfeld zum Bearbeiten des VPC-Schnittstellenendpunkts wird geöffnet.
6. Wählen Sie die erforderlichen Sicherheitsgruppen und Subnetze aus und klicken Sie auf Aktualisieren. Die Seite mit den Details zum VPC-Schnittstellen-Endpunkt wird mit der Meldung

über die erfolgreiche Aktualisierung des VPC-Schnittstellenendpunkts in der oberen Leiste geöffnet.

> **Note**
>
> Wenn Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs benötigt, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln.
> Weitere Informationen zur Infrastruktursicherheit und zu Sicherheitsgruppen, einschließlich bewährter Methoden, finden Sie in den folgenden Themen im Amazon VPC-Benutzerhandbuch: Steuern des Netzwerkverkehrs und Steuern des Datenverkehrs zu Ihren AWS-Ressourcen mithilfe von Sicherheitsgruppen.

VPC-Schnittstellenendpunkt löschen

Wenn Sie nicht möchten, dass Ihr App Runner-Dienst privat zugänglich ist, können Sie Ihren eingehenden Datenverkehr auf Öffentlich setzen. Wenn Sie zu Öffentlich wechseln, wird der private Endpunkt entfernt, der Endpunkt der VPC-Schnittstelle wird jedoch nicht gelöscht

Um den VPC-Schnittstellenendpunkt zu löschen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie zu Ihrem Dienst und wählen Sie im linken Bereich Netzwerkkonfigurationen aus.
3. Wählen Sie Eingehender Verkehr, um die VPC-Schnittstellenendpunkte anzuzeigen, die den jeweiligen Diensten zugeordnet sind.

> **Note**
>
> Bevor Sie einen VPC-Schnittstellenendpunkt löschen, entfernen Sie ihn aus allen Diensten, mit denen er verbunden ist, indem Sie Ihren Service aktualisieren.

4. Wählen Sie Löschen.

Wenn Dienste mit dem VPC-Schnittstellenendpunkt verbunden sind, erhalten Sie die Meldung „VPC-Schnittstellenendpunkt kann nicht gelöscht werden“. Wenn keine Dienste mit dem VPC-Schnittstellenendpunkt verbunden sind, erhalten Sie eine Meldung zur Bestätigung des Löschvorgangs.

5. Wählen Sie Löschen. Die Seite Netzwerkkonfigurationen für den eingehenden Datenverkehr wird geöffnet. In der oberen Leiste wird die Meldung angezeigt, dass der VPC-Schnittstellenendpunkt erfolgreich gelöscht wurde.

App Runner API oder AWS CLI

Sie können eine Anwendung auf App Runner bereitstellen, auf die nur von einer Amazon VPC aus zugegriffen werden kann.

Informationen zu den Berechtigungen, die erforderlich sind, um Ihren Service privat zu machen, finden Sie unterthe section called “Berechtigungen”.

> **Note**
>
> Derzeit unterstützt App Runner IPv6 nur öffentliche Endpunkte. IPv6 Endpunkte werden für App Runner-Services, die in einer Amazon Virtual Private Cloud (Amazon VPC) gehostet werden, nicht unterstützt. Wenn Sie einen Service, der einen öffentlichen Dual-Stack-Endpunkt verwendet, auf einen privaten Endpunkt aktualisieren, unterstützt Ihr App Runner-Service standardmäßig nur Datenverkehr von IPv4 Endpunkten und empfängt keinen Datenverkehr von Endpunkten. IPv6

So erstellen Sie eine private Serviceverbindung zu Amazon VPC

1. Erstellen Sie einen VPC-Schnittstellenendpunkt, eine AWS PrivateLink Ressource, um eine Verbindung zu App Runner herzustellen. Geben Sie dazu Subnetze und Sicherheitsgruppen an, die der Anwendung zugeordnet werden sollen. Im Folgenden finden Sie ein Beispiel für die Erstellung eines VPC-Schnittstellenendpunkts.

   > **Note**
   >
   > Wenn Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs benötigt, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs Dies liegt daran, dass wir derzeit

die Weiterleitung von IP-Quelldaten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Aus diesem Grund entsprechen die Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, ACLs nicht den IP-basierten Regeln.
Weitere Informationen zur Infrastruktursicherheit und zu Sicherheitsgruppen, einschließlich bewährter Methoden, finden Sie in den folgenden Themen im Amazon VPC-Benutzerhandbuch: Steuern des Netzwerkverkehrs und Steuern des Datenverkehrs zu Ihren AWS-Ressourcen mithilfe von Sicherheitsgruppen.

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. Referenzieren Sie den VPC-Schnittstellenendpunkt, indem Sie die API-Aktionen CreateServiceoder UpdateServiceApp Runner über die CLI verwenden. Konfigurieren Sie Ihren Service so, dass er nicht öffentlich zugänglich ist. `False`Im `IngressConfiguration` Element des `NetworkConfiguration` Parameters auf festgelegt`IsPubliclyAccessible`. Im Folgenden finden Sie ein Beispiel für die Referenzierung eines VPC-Schnittstellenendpunkts.

   Example

   ```
   aws apprunner create-service
    --network-configuration: ingress-configuration=<ingress_configuration>
    --service-name: com.amazonaws.us-east-1.apprunner.requests
    --source-configuration: <source_configuration>
    # Ingress Configuration
    {
    "IsPubliclyAccessible": False
    }
   ```

3. Rufen Sie die `create-vpc-ingress-connection` API-Aktion auf, um die VPC-Ingress-Verbindungsressource für App Runner zu erstellen und sie dem VPC-Schnittstellenendpunkt zuzuordnen, den Sie im vorherigen Schritt erstellt haben. Es gibt einen Domainnamen zurück, der für den Zugriff auf Ihren Service in der angegebenen VPC verwendet wird. Im Folgenden finden Sie ein Beispiel für die Erstellung einer VPC-Ingress-Verbindungsressource.

Example Anforderung

```
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 Antwort

```
{
   "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>
}
```

VPC-Eingangsverbindung aktualisieren

Sie können die Ressource VPC Ingress Connection aktualisieren. Die VPC-Ingress-Verbindung muss sich in einem der folgenden Zustände befinden, damit sie aktualisiert werden kann:

- VERFÜGBAR
- FAILED_CREATION
- FEHLGESCHLAGENES UPDATE

Im Folgenden finden Sie ein Beispiel für die Aktualisierung einer VPC-Ingress-Verbindungsressource.

Example Anforderung

```
aws apprunner update-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Antwort

```
{
    "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>
}
```

VPC-Eingangsverbindung löschen

Sie können die Ressource VPC Ingress Connection löschen, wenn Sie die private Verbindung zur Amazon VPC nicht mehr benötigen.

Die VPC-Eingangsverbindung muss sich in einem der folgenden Zustände befinden, um gelöscht zu werden:

- VERFÜGBAR
- FEHLER BEI DER ERSTELLUNG
- AKTUALISIERUNG FEHLGESCHLAGEN
- FEHLGESCHLAGENES LÖSCHEN

Im Folgenden finden Sie ein Beispiel für das Löschen einer VPC-Ingress-Verbindung

Example Anforderung

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Antwort

```
{
   "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>
}
```

Verwenden Sie die folgenden App Runner-API-Aktionen, um den privaten eingehenden Datenverkehr für Ihren Service zu verwalten.

- CreateVpcIngressConnection— Erstellen Sie eine neue VPC-Ingress-Verbindungsressource. App Runner benötigt diese Ressource, wenn Sie Ihren App-Runner-Service einem Amazon-VPC-Endpunkt zuordnen möchten.
- ListVpcIngressConnections— Gibt eine Liste der AWS App Runner VPC Ingress Connection-Endpunkte zurück, die Ihrem Konto zugeordnet sind. AWS
- DescribeVpcIngressConnection— Gibt eine vollständige Beschreibung der AWS App Runner VPC Ingress Connection-Ressource zurück.
- UpdateVpcIngressConnection— Aktualisieren Sie die AWS App Runner VPC Ingress Connection-Ressource.
- DeleteVpcIngressConnection— Löscht eine App Runner VPC Ingress Connection-Ressource, die dem App Runner-Dienst zugeordnet ist.

Weitere Informationen zur Verwendung der App Runner API finden Sie im App Runner API-Referenzhandbuch.

Aktivierung IPv6 für eingehenden öffentlichen Verkehr

Wenn Sie möchten, dass Ihr Dienst eingehenden Netzwerkverkehr von IPv6 Adressen oder von beiden IPv4 IPv6 Adressen empfängt, wählen Sie den Dual-Stack-Adresstyp für den öffentlichen Endpunkt. Wenn Sie eine neue Anwendung erstellen, finden Sie diese Einstellung im Abschnitt Dienst konfigurieren > Netzwerk. Weitere Informationen zur Aktivierung IPv6 mithilfe der App Runner-Konsole oder der App Runner-API finden Sie unterthe section called “Dual-Stack für öffentliche Endgeräte verwalten”.

Weitere Informationen zur Adoption IPv6 von AWS finden Sie IPv6 unter AWS.

App Runner unterstützt Dual-Stack nur für öffentliche App Runner-Dienstendpunkte. Für alle privaten App Runner-Dienste IPv4 wird nur unterstützt.

> **Note**
>
> Wenn Sie den IP-Adresstyp auf Dual-Stack eingestellt haben und Ihre Netzwerkkonfiguration von einem öffentlichen auf einen privaten Endpunkt ändern, ändert App Runner Ihren Adresstyp automatisch in IPv4. Das liegt daran, dass App Runner IPv6 nur öffentliche Endpunkte unterstützt.

Erfahren Sie Hintergrundinformationen zu vs IPv4 IPv6

Die IPv4 Netzwerkschicht, die üblicherweise zur Weiterleitung von Netzwerkverkehr über das Internet verwendet wird, verwendet ein 32-Bit-Adressschema. Dieser Adressraum ist begrenzt und kann bei einer großen Anzahl von Netzwerkgeräten erschöpft sein. Aus diesem Grund wird Network Address Translation (NAT) in der Regel verwendet, um mehrere IPv4 Adressen über eine einzige öffentliche Netzwerkadresse weiterzuleiten.

IPv6, eine neuere Version des Internetprotokolls, baut auf dem Adressraum auf IPv4 und erweitert ihn um ein 128-Bit-Adressierungsschema. Mit IPv6 können Sie ein Netzwerk mit einer nahezu unbegrenzten Anzahl verbundener Geräte aufbauen. Aufgrund der großen Anzahl an Netzwerkadressen wird NAT von nicht benötigt IPv6.

IPv4 und IPv6 Endpunkte sind nicht miteinander kompatibel, da IPv4 Endpunkte keinen eingehenden IPv6 Verkehr empfangen können und umgekehrt. Dual Stack bietet eine praktische Lösung, bei der IPv4 sowohl der Netzwerkverkehr als auch der IPv6 Netzwerkverkehr gleichzeitig unterstützt werden können.

Verwaltung des Dual-Stacks für eingehenden öffentlichen Datenverkehr

Verwalten Sie den Dual-Stack-Adresstyp für eingehenden öffentlichen Verkehr mit einer der folgenden Methoden:

- the section called “App Runner-Konsole”
- the section called “App Runner API oder AWS CLI”

App Runner-Konsole

Sie können den Dual-Stack-Adresstyp für den eingehenden Internetverkehr wählen, wenn Sie einen Dienst mit der App Runner-Konsole erstellen oder wenn Sie seine Konfiguration später aktualisieren.

Um den Dual-Stack-Adresstyp zu aktivieren

1. Wenn Sie einen Dienst erstellen oder aktualisieren, erweitern Sie den Abschnitt Netzwerk unter Dienst konfigurieren.
2. Wählen Sie Öffentlicher Endpunkt für Eingehenden Netzwerkverkehr. Die Option IP-Adresstyp für öffentliche Endgeräte wird geöffnet.
3. Erweitern Sie den IP-Adresstyp für öffentliche Endgeräte, um die folgenden IP-Adresstypen anzuzeigen.

   - IPv4
   - Dual-Stack (IPv4 und IPv6)

   > **Note**
   >
   > Wenn Sie den IP-Adresstyp des öffentlichen Endpunkts nicht erweitern, um eine Auswahl zu treffen, weist App Runner die Konfiguration IPv4 als Standardkonfiguration zu.

4. Wählen Sie Dual-Stack (IPv4 und IPv6).
5. Wählen Sie Weiter und dann Create & Deploy, wenn Sie einen Service erstellen. Andernfalls wählen Sie Änderungen speichern, wenn Sie einen Dienst aktualisieren.

   Wenn der Dienst bereitgestellt wird, empfängt Ihre Anwendung Netzwerkverkehr sowohl von Endpunkten als auch IPv4 von IPv6 Endpunkten.

> **Note**
>
> Derzeit unterstützt App Runner IPv6 nur öffentliche Endpunkte. IPv6 Endpunkte werden für App Runner-Services, die in einer Amazon Virtual Private Cloud (Amazon VPC) gehostet werden, nicht unterstützt. Wenn Sie einen Service, der einen öffentlichen Dual-Stack-Endpunkt verwendet, auf einen privaten Endpunkt aktualisieren, unterstützt Ihr App Runner-Service standardmäßig nur Datenverkehr von IPv4 Endpunkten und empfängt keinen Datenverkehr von Endpunkten. IPv6

Um den Adresstyp zu ändern

1. Folgen Sie den Schritten, um einen Dienst zu aktualisieren, und navigieren Sie zu Netzwerk.

2. Navigieren Sie unter Eingehender Netzwerkverkehr zum Typ der öffentlichen Endpunkt-IP-Adresse und wählen Sie den erforderlichen Adresstyp aus.
3. Wählen Sie Änderungen speichern. Ihr Dienst wird mit Ihrer Auswahl aktualisiert.

App Runner API oder AWS CLI

Wenn Sie die API-Aktionen CreateServiceoder UpdateServiceApp Runner aufrufen, verwenden Sie das `IpAddressType` Mitglied des `NetworkConfiguration` Parameters, um den Adresstyp anzugeben. Die unterstützten Werte, die Sie angeben können, sind `IPv4` und`DUAL_STACK`. Geben Sie an, `DUAL_STACK` ob Ihr Dienst Internetverkehr von IPv4 und IPv6 Endpunkten empfangen soll. Wenn Sie keinen Wert für angeben`IpAddressType`, IPv4 wird standardmäßig verwendet.

Im Folgenden finden Sie ein Beispiel für die Erstellung eines Dienstes mit dem Dual-Stack als IP-Adresse. In diesem Beispiel wird eine `input.json` Datei aufgerufen.

Example Anfrage zur Erstellung eines Dienstes mit Dual-Stack-Unterstützung

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

Example Inhalt von **`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 Antwort

```
{
```

```
  "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"
}
```

> **Note**
>
> Derzeit unterstützt App Runner IPv6 nur öffentliche Endpunkte. IPv6 Endpunkte werden für App Runner-Services, die in einer Amazon Virtual Private Cloud (Amazon VPC) gehostet werden, nicht unterstützt. Wenn Sie einen Service, der einen öffentlichen Dual-Stack-Endpunkt verwendet, auf einen privaten Endpunkt aktualisieren, unterstützt Ihr App Runner-Service standardmäßig nur Datenverkehr von IPv4 Endpunkten und empfängt keinen Datenverkehr von Endpunkten. IPv6

Weitere Informationen zum API-Parameter finden Sie unter. NetworkConfiguration

VPC-Zugriff für ausgehenden Verkehr aktivieren

Standardmäßig kann Ihre AWS App Runner Anwendung Nachrichten an öffentliche Endpunkte senden. Dazu gehören Ihre eigenen Lösungen und alle anderen öffentlichen Websites oder Webdienste. AWS-Services Ihre Anwendung kann sogar Nachrichten an öffentliche Endpunkte von Anwendungen senden, die in einer VPC von Amazon Virtual Private Cloud (Amazon VPC) ausgeführt werden. Wenn Sie beim Starten Ihrer Umgebung keine VPC konfigurieren, verwendet App Runner die Standard-VPC, die öffentlich ist.

Sie können sich dafür entscheiden, Ihre Umgebung in einer benutzerdefinierten VPC zu starten, um die Netzwerk- und Sicherheitseinstellungen für ausgehenden Datenverkehr anzupassen. Sie können Ihren AWS App Runner Service für den Zugriff auf Anwendungen aktivieren, die in einer privaten VPC von Amazon Virtual Private Cloud (Amazon VPC) aus ausgeführt werden. Danach kann sich Ihre Anwendung mit anderen Anwendungen verbinden und Nachrichten an diese senden, die in einer Amazon Virtual Private Cloud (Amazon VPC) gehostet werden. Beispiele sind eine Amazon RDS-Datenbank ElastiCache, Amazon und andere private Dienste, die in einer privaten VPC gehostet werden.

VPC-Anschluss

Sie können Ihren Service mit einer VPC verknüpfen, indem Sie in der App Runner-Konsole einen VPC-Endpunkt namens VPC Connector erstellen. Um einen VPC-Connector zu erstellen, geben Sie die VPC, ein oder mehrere Subnetze und optional eine oder mehrere Sicherheitsgruppen an. Nachdem Sie einen VPC Connector konfiguriert haben, können Sie ihn mit einem oder mehreren App Runner-Diensten verwenden.

Einmalige Latenz

Wenn Sie Ihren App Runner-Dienst mit einem benutzerdefinierten VPC-Connector für ausgehenden Datenverkehr konfigurieren, kann es zu einer einmaligen Startlatenz von zwei bis fünf Minuten kommen. Der Startvorgang wartet, bis der VPC Connector bereit ist, eine Verbindung zu anderen Ressourcen herzustellen, bevor er den Dienststatus auf Wird ausgeführt setzt. Sie können einen Dienst mit einem benutzerdefinierten VPC-Connector konfigurieren, wenn Sie ihn zum ersten Mal erstellen, oder Sie können dies später tun, indem Sie ein Service-Update durchführen.

Beachten Sie, dass es zu keiner Latenz kommt, wenn Sie dieselbe VPC-Connectorkonfiguration für einen anderen Dienst wiederverwenden. Die VPC-Connectorkonfiguration basiert auf der Kombination aus Sicherheitsgruppe und Subnetz. Bei einer bestimmten VPC-Connector-Konfiguration tritt die Latenz nur einmal auf, nämlich bei der ersten Erstellung der VPC-Connector-Hyperplane ENIs (elastische Netzwerkschnittstellen).

Mehr über benutzerdefinierte VPC-Konnektoren und Hyperplane AWS

Die VPC-Konnektoren in App Runner basieren auf AWS Hyperplane, dem internen Amazon-Netzwerksystem, das hinter mehreren AWS Ressourcen wie Network Load Balancer, NAT Gateway und AWS steht. PrivateLink Die AWS Hyperplane-Technologie bietet hohen Durchsatz und niedrige Latenz sowie einen höheren Grad an gemeinsamer Nutzung. Eine Hyperplane-ENI wird in Ihren Subnetzen erstellt, wenn Sie einen VPC-Connector erstellen und ihn Ihrem Service zuordnen. Eine VPC-Connectorkonfiguration basiert auf einer Kombination aus Sicherheitsgruppe und Subnetz, und Sie können in mehreren App Runner-Diensten auf denselben VPC Connector verweisen. Daher werden die zugrunde liegenden Hyperplane von Ihren App ENIs Runner-Diensten gemeinsam genutzt. Diese gemeinsame Nutzung ist machbar, auch wenn Sie die Anzahl der Aufgaben erhöhen, die zur Bewältigung der Anforderungslast erforderlich sind, und führt zu einer effizienteren Nutzung des IP-Raums in Ihrer VPC. Weitere Informationen finden Sie im AWS-Container-Blog unter Deep Dive on AWS App Runner VPC Networking.

Subnetz

Jedes Subnetz befindet sich in einer bestimmten Availability Zone. Für eine hohe Verfügbarkeit empfehlen wir, Subnetze für mindestens drei Availability Zones auszuwählen. Wenn die Region weniger als drei Availability Zones hat, empfehlen wir Ihnen, Ihre Subnetze für alle unterstützten Availability Zones auszuwählen.

Achten Sie bei der Auswahl eines Subnetzes für Ihre VPC darauf, dass Sie ein privates Subnetz und kein öffentliches Subnetz wählen. Dies liegt daran, dass der App Runner-Dienst beim Erstellen eines VPC-Connectors in jedem der Subnetze eine Hyperplane-ENI erstellt. Jedem Hyperplane-ENI wird nur eine private IP-Adresse zugewiesen und sie ist mit einem Schlüssel-Tag versehen. AWSAppRunnerManaged Wenn Sie ein öffentliches Subnetz wählen, treten beim Ausführen Ihres App Runner-Dienstes Fehler auf. Wenn Ihr Dienst jedoch auf Dienste zugreifen muss, die sich im Internet oder in anderen öffentlichen Bereichen befinden AWS-Services, finden Sie weitere Informationen unterthe section called “Überlegungen bei der Auswahl eines Subnetzes ”.

Überlegungen bei der Auswahl eines Subnetzes

- Wenn Sie Ihren Service mit einer VPC verbinden, hat der ausgehende Datenverkehr keinen Zugriff auf das öffentliche Internet. Der gesamte ausgehende Datenverkehr von Ihrer Anwendung wird über die VPC geleitet, mit der Ihr Service verbunden ist. Alle Netzwerkregeln für die VPC gelten für den ausgehenden Datenverkehr Ihrer Anwendung. Dies bedeutet, dass Ihre Dienste nicht auf das öffentliche Internet zugreifen können und. AWS APIs Um Zugriff zu erhalten, führen Sie einen der folgenden Schritte aus:
  - Connect die Subnetze über ein NAT-Gateway mit dem Internet.
  - Richten Sie VPC-Endpunkte für die ein AWS-Services , auf die Sie zugreifen möchten. Ihr Service bleibt innerhalb der Amazon VPC, indem Sie AWS PrivateLink
- In einigen Availability Zones werden die Subnetze, die mit App Runner-Diensten verwendet werden können, AWS-Regionen nicht unterstützt. Wenn Sie Subnetze in diesen Availability Zones auswählen, kann Ihr Dienst nicht erstellt oder aktualisiert werden. In diesen Situationen bietet App Runner eine detaillierte Fehlermeldung, die auf die nicht unterstützten Subnetze und Availability Zones hinweist. In diesem Fall können Sie das Problem beheben, indem Sie die nicht unterstützten Subnetze aus Ihrer Anfrage entfernen und es dann erneut versuchen.

Sicherheitsgruppe

Sie können optional die Sicherheitsgruppen angeben, die App Runner für den Zugriff auf die angegebenen AWS Subnetze verwendet. Wenn Sie keine Sicherheitsgruppen angeben, verwendet App Runner die Standardsicherheitsgruppe der VPC. Die Standard-Sicherheitsgruppe lässt den gesamten ausgehenden Datenverkehr zu.

Das Hinzufügen einer Sicherheitsgruppe bietet eine zusätzliche Sicherheitsebene für die VCP Connectors, sodass Sie mehr Kontrolle über den Netzwerkverkehr haben. Der VPC-Connector wird nur für die ausgehende Kommunikation aus Ihrer Anwendung verwendet. Sie verwenden Regeln für ausgehenden Datenverkehr, um die Kommunikation mit den gewünschten Zielendpunkten zu ermöglichen. Sie müssen außerdem sicherstellen, dass für alle Sicherheitsgruppen, die der Zielressource zugeordnet sind, die entsprechenden Regeln für eingehenden Datenverkehr gelten. Andernfalls können diese Ressourcen keinen Datenverkehr akzeptieren, der von den VPC Connector-Sicherheitsgruppen stammt.

> **Note**
>
> Wenn Sie Ihren Service mit einer VPC verknüpfen, ist der folgende Datenverkehr nicht betroffen:
>
> - Eingehender Verkehr — Eingehende Nachrichten, die Ihre Anwendung empfängt, werden von einer zugehörigen VPC nicht beeinflusst. Die Nachrichten werden über den öffentlichen Domainnamen weitergeleitet, der mit Ihrem Service verknüpft ist, und interagieren nicht mit der VPC.
> - App Runner-Verkehr — App Runner verwaltet verschiedene Aktionen in Ihrem Namen, z. B. das Abrufen von Quellcode und Bildern, das Übertragen von Protokollen und das Abrufen von Geheimnissen. Der Traffic, den diese Aktionen generieren, wird nicht über Ihre VPC geleitet.

Weitere Informationen zur AWS App Runner Integration mit Amazon VPC finden Sie unter AWS App Runner VPC Networking.

> **Note**
>
> Für ausgehenden Datenverkehr unterstützt App Runner derzeit nur. IPv4

VPC-Zugriff verwalten

> **Note**
>
> Wenn Sie einen VPC-Connector für ausgehenden Verkehr für einen Dienst erstellen, tritt beim darauffolgenden Dienststartvorgang eine einmalige Latenz auf. Sie können diese Konfiguration für einen neuen Dienst bei der Erstellung oder später mit einem Service-Update festlegen. Weitere Informationen finden Sie Einmalige Latenz im Kapitel Networking with App Runner in diesem Handbuch.

Verwalten Sie den VPC-Zugriff für Ihre App Runner-Dienste mit einer der folgenden Methoden:

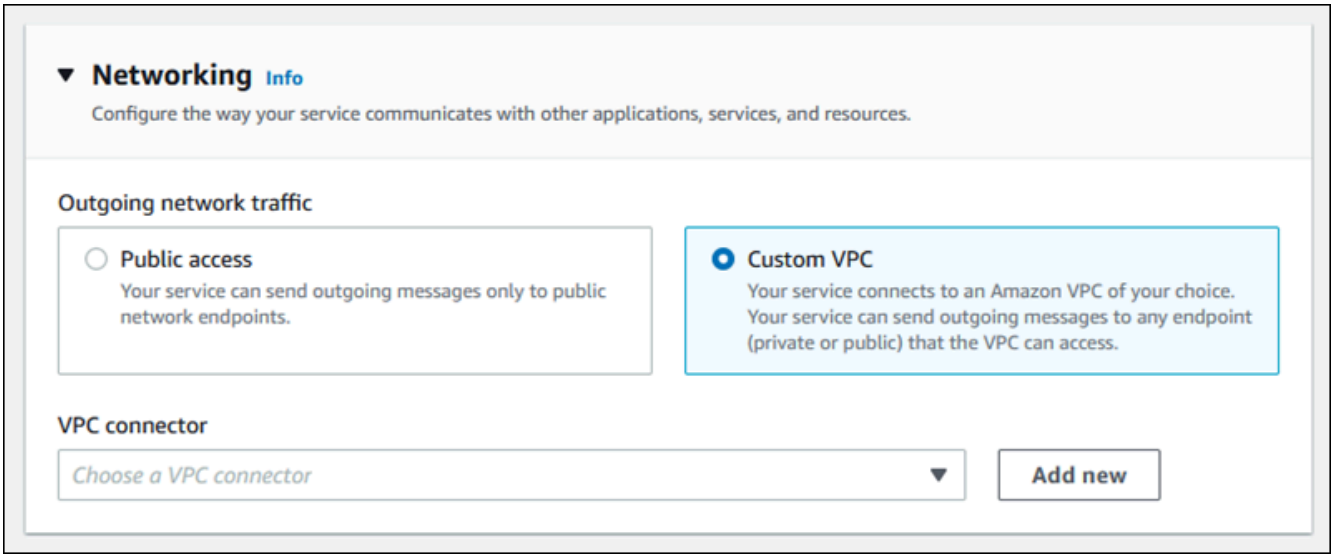
App Runner console

Wenn Sie einen Dienst mit der App Runner-Konsole erstellen oder seine Konfiguration später aktualisieren, können Sie wählen, ob Sie Ihren ausgehenden Datenverkehr konfigurieren möchten. Suchen Sie auf der Konsolenseite nach dem Abschnitt Netzwerkkonfiguration. Wählen Sie für ausgehenden Netzwerkverkehr eine der folgenden Optionen aus:

- Öffentlicher Zugriff: Um Ihren Dienst mit öffentlichen Endpunkten anderer AWS-Services zu verknüpfen.
- Benutzerdefinierte VPC: Um Ihren Service mit einer VPC von Amazon VPC zu verknüpfen. Ihre Anwendung kann sich mit anderen Anwendungen verbinden und Nachrichten an diese senden, die in einer Amazon VPC gehostet werden.

So aktivieren Sie Custom VPC

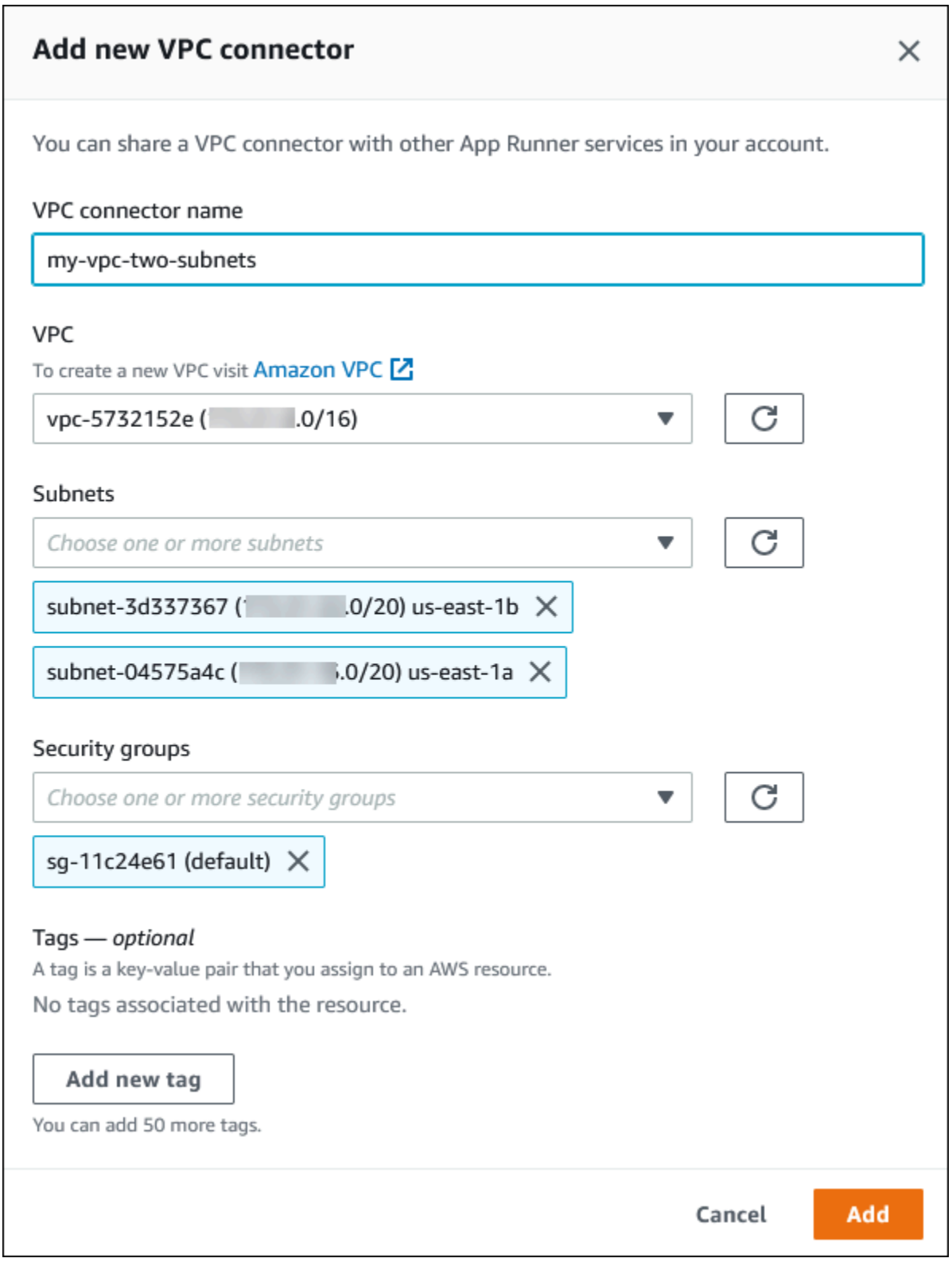
1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie unter Dienst konfigurieren zum Abschnitt Netzwerk.



3. Wählen Sie Custom VPC für ausgehenden Netzwerkverkehr.
4. Wählen Sie im Navigationsbereich VPC-Connector aus.

   Wenn Sie die VPC-Connectors erstellt haben, zeigt die Konsole eine Liste der VPC-Connectors in Ihrem Konto an. Sie können einen vorhandenen VPC-Connector auswählen und Weiter wählen, um Ihre Konfiguration zu überprüfen. Fahren Sie dann mit dem letzten Schritt fort. Alternativ können Sie mithilfe der folgenden Schritte einen neuen VPC-Connector hinzufügen.
5. Wählen Sie Neu hinzufügen, um einen neuen VPC-Connector für Ihren Service zu erstellen.

   Anschließend wird das Dialogfeld Neuen VPC-Connector hinzufügen geöffnet.



6. Geben Sie einen Namen für Ihren VPC-Connector ein und wählen Sie die erforderliche VPC aus der verfügbaren Liste aus.

7. Wählen Sie für Subnetze ein Subnetz für jede Availability Zone aus, von der aus Sie auf den App Runner-Dienst zugreifen möchten. Wählen Sie für eine bessere Verfügbarkeit drei Subnetze aus. Oder, wenn es weniger als drei Subnetze gibt, wählen Sie alle verfügbaren Subnetze aus.

   > **Note**
   >
   > Stellen Sie sicher, dass Sie dem VPC-Connector private Subnetze zuweisen. Wenn Sie dem VPC-Connector öffentliche Subnetze zuweisen, kann Ihr Service während eines Updates keine automatische Erstellung oder ein Rollback durchführen.

8. (Optional) Wählen Sie unter Sicherheitsgruppe die Sicherheitsgruppen aus, die den Endpunkt-Netzwerkschnittstellen zugeordnet werden sollen.
9. (Optional) Sie fügen ein Tag hinzu, indem Sie neues Tag hinzufügen auswählen und den Schlüssel und den Wert für das Tag eingeben.
10. Wählen Sie Hinzufügen aus.

    Die Details des VPC-Connectors, den Sie erstellt haben, werden unter VPC-Connector angezeigt.
11. Wählen Sie Weiter, um Ihre Konfiguration zu überprüfen, und wählen Sie dann Create and deploy aus.

    App Runner erstellt eine VPC-Connector-Ressource für Sie und ordnet sie dann Ihrem Service zu. Wenn der Dienst erfolgreich erstellt wurde, zeigt die Konsole das Service-Dashboard mit einer Serviceübersicht über den neuen Service an.

App Runner API or AWS CLI

Wenn Sie die API-Aktionen CreateServiceoder UpdateServiceApp Runner aufrufen, verwenden Sie das `EgressConfiguration` Mitglied des `NetworkConfiguration` Parameters, um eine VPC-Connector-Ressource für Ihren Service anzugeben.

Verwenden Sie die folgenden App Runner API-Aktionen, um Ihre VPC Connector-Ressourcen zu verwalten.

- CreateVpcConnector— Erzeugt einen neuen VPC-Connector.
- ListVpcConnectors— Gibt eine Liste der VPC-Konnektoren zurück, die Ihrem AWS-Konto zugeordnet sind. Die Liste enthält vollständige Beschreibungen.

- DescribeVpcConnector— Gibt eine vollständige Beschreibung eines VPC-Connectors zurück.
- DeleteVpcConnector— Löscht einen VPC-Connector. Wenn Sie das VPC-Connector-Kontingent für Ihren erreichen AWS-Konto, müssen Sie möglicherweise nicht benötigte VPC-Connectors löschen.

Um eine Anwendung auf App Runner bereitzustellen, die ausgehenden Zugriff auf eine VPC hat, müssen Sie zunächst einen VPC-Connector erstellen. Sie können dies tun, indem Sie ein oder mehrere Subnetze und Sicherheitsgruppen angeben, die der Anwendung zugeordnet werden sollen. Sie können dann in Create oder UpdateServiceüber die CLI auf den VPC-Connector verweisen, wie im folgenden Beispiel dargestellt:

```
            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
```

Observability für Ihren App Runner-Dienst

AWS App Runner lässt sich in mehrere AWS Dienste integrieren, um Ihnen eine umfangreiche Suite von Observability-Tools für Ihren App Runner-Service zur Verfügung zu stellen. In den Themen dieses Kapitels werden diese Funktionen beschrieben.

Themen

Verfolgen der App Runner-Dienstaktivitäten

AWS App Runner verwendet eine Liste von Vorgängen, um die Aktivitäten in Ihrem App Runner-Dienst zu verfolgen. Eine Operation stellt einen asynchronen Aufruf einer API-Aktion dar, z. B. das Erstellen eines Dienstes, das Aktualisieren einer Konfiguration und das Bereitstellen eines Dienstes. In den folgenden Abschnitten erfahren Sie, wie Sie Aktivitäten in der App Runner-Konsole und mithilfe der API verfolgen.

Verfolgen Sie die Aktivität des App Runner-Dienstes

Verfolgen Sie Ihre App Runner-Dienstaktivitäten mit einer der folgenden Methoden:

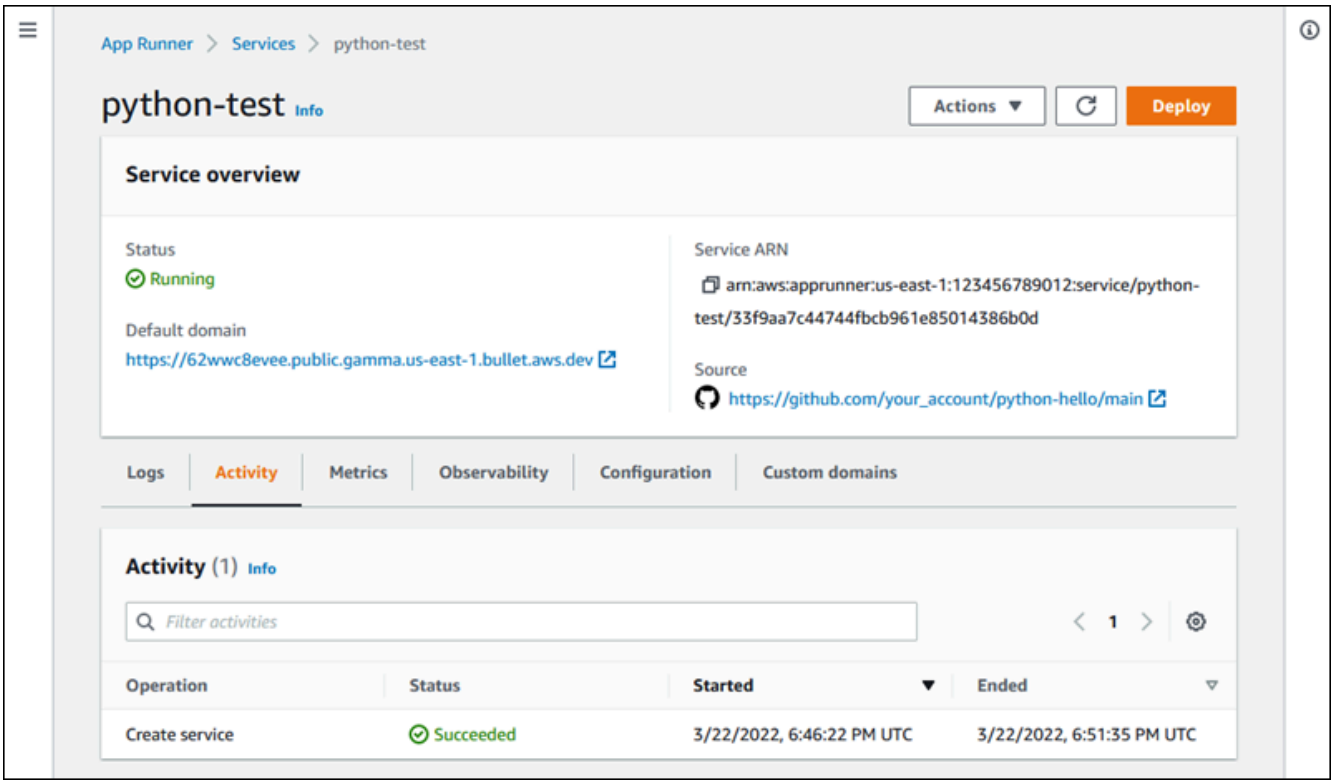
App Runner console

Die App Runner-Konsole zeigt Ihre App Runner-Dienstaktivitäten an und bietet weitere Möglichkeiten, Abläufe zu erkunden.

Um die Aktivität Ihres Dienstes zu sehen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite den Tab Aktivität aus, falls er nicht bereits ausgewählt ist.

   Die Konsole zeigt eine Liste von Vorgängen an.

4. Um nach bestimmten Vorgängen zu suchen, suchen Sie in der Liste nach einem Suchbegriff. Sie können nach jedem Wert suchen, der in der Tabelle erscheint.

5. Wählen Sie eine der aufgeführten Operationen aus, um das zugehörige Protokoll anzuzeigen oder herunterzuladen.

App Runner API or AWS CLI

Die ListOperationsAktion gibt unter Angabe des Amazon-Ressourcennamens (ARN) eines App Runner-Dienstes eine Liste von Vorgängen zurück, die in diesem Service aufgetreten sind. Jedes Listenelement enthält eine Vorgangs-ID und einige Tracking-Details.

App Runner-Protokolle anzeigen, die in Logs gestreamt wurden CloudWatch

Sie können Amazon CloudWatch Logs verwenden, um Protokolldateien zu überwachen, zu speichern und darauf zuzugreifen, die Ihre Ressourcen in verschiedenen AWS Diensten generieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.

AWS App Runner sammelt die Ergebnisse Ihrer Anwendungsbereitstellungen und Ihres aktiven Dienstes und streamt sie in CloudWatch Logs. In den folgenden Abschnitten werden App Runner-Protokollstreams aufgeführt und es wird gezeigt, wie Sie sie in der App Runner-Konsole anzeigen können.

App Runner protokolliert Gruppen und Streams

CloudWatch Logs speichert Protokolldaten in Protokollströmen, die es weiter in Protokollgruppen organisiert. Ein Protokollstream ist eine Abfolge von Protokollereignissen aus einer bestimmten Quelle. Eine Protokollgruppe ist eine Gruppe von Protokollstreams, die dieselben Einstellungen für die Aufbewahrung, Überwachung und Zugriffskontrolle besitzen.

App Runner definiert zwei CloudWatch Logs-Protokollgruppen mit jeweils mehreren Protokollstreams für jeden App Runner-Dienst in Ihrem AWS-Konto.

Dienstprotokolle

Die Dienstprotokollgruppe enthält Protokollausgaben, die von App Runner generiert werden, während App Runner Ihren App Runner-Dienst verwaltet und darauf reagiert.

Name der Protokollgruppe	Beispiel
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bc b23da/service`

Innerhalb der Dienstprotokollgruppe erstellt App Runner einen Ereignisprotokollstream, um Aktivitäten im Lebenszyklus Ihres App Runner-Dienstes zu erfassen. Dies könnte beispielsweise bedeuten, dass Ihre Anwendung gestartet oder angehalten wird.

Darüber hinaus erstellt App Runner einen Protokollstream für jeden lang andauernden asynchronen Vorgang, der sich auf Ihren Dienst bezieht. Der Name des Protokollstreams spiegelt den Vorgangstyp und die spezifische Vorgangs-ID wider.

Eine Bereitstellung ist eine Art von Operation. Bereitstellungsprotokolle enthalten die Protokollausgabe der Build- und Bereitstellungsschritte, die App Runner ausführt, wenn Sie einen Dienst erstellen oder eine neue Version Ihrer Anwendung bereitstellen. Die Namen der Bereitstellungsprotokolldatenströme beginnen mit `deployment/` und enden mit der ID des Vorgangs, der die Bereitstellung durchführt. Bei diesem Vorgang handelt es sich entweder um einen CreateServiceAufruf für die erste Anwendungsbereitstellung oder um einen StartDeploymentAufruf für jede weitere Bereitstellung.

Innerhalb eines Bereitstellungsprotokolls beginnt jede Protokollnachricht mit einem Präfix:

- `[AppRunner]`— Ausgabe, die App Runner während der Bereitstellung generiert.
- `[Build]`— Ausgabe Ihrer eigenen Build-Skripte.

Name des Log-Streams	Beispiel
`events`	N/A (fester Name)
operation-type `/`*operation-id*	`deployment/c2c8eeedea164f45 9cf78f12a8953390`

Anwendungsprotokolle

Die Anwendungsprotokollgruppe enthält die Ausgabe Ihres laufenden Anwendungscodes.

Name der Protokollgruppe	Beispiel
`/aws/apprunner/` *service-n ame* `/`*service-id* `/application`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application`

Innerhalb der Anwendungsprotokollgruppe erstellt App Runner einen Protokollstream für jede Instanz (Skalierungseinheit), auf der Ihre Anwendung ausgeführt wird.

Name des Protokollstreams	Beispiel
`instance/ instance-id`	`instance/1a80bc9134a84699b7 b3432ebeebb591`

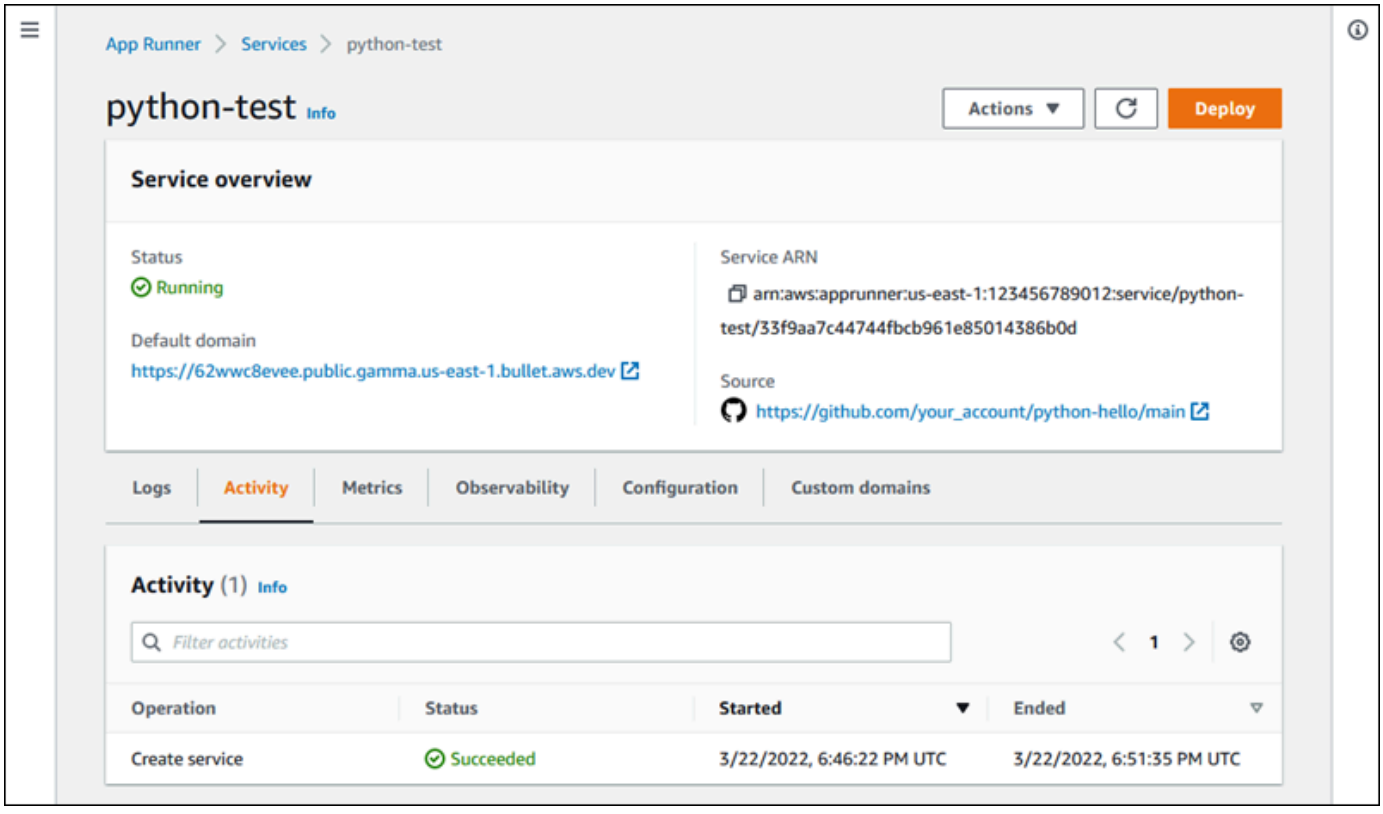
App Runner-Protokolle in der Konsole anzeigen

Die App Runner-Konsole zeigt eine Zusammenfassung aller Protokolle für Ihren Dienst an und ermöglicht es Ihnen, sie anzusehen, zu durchsuchen und herunterzuladen.

Um Protokolle für Ihren Service anzuzeigen

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

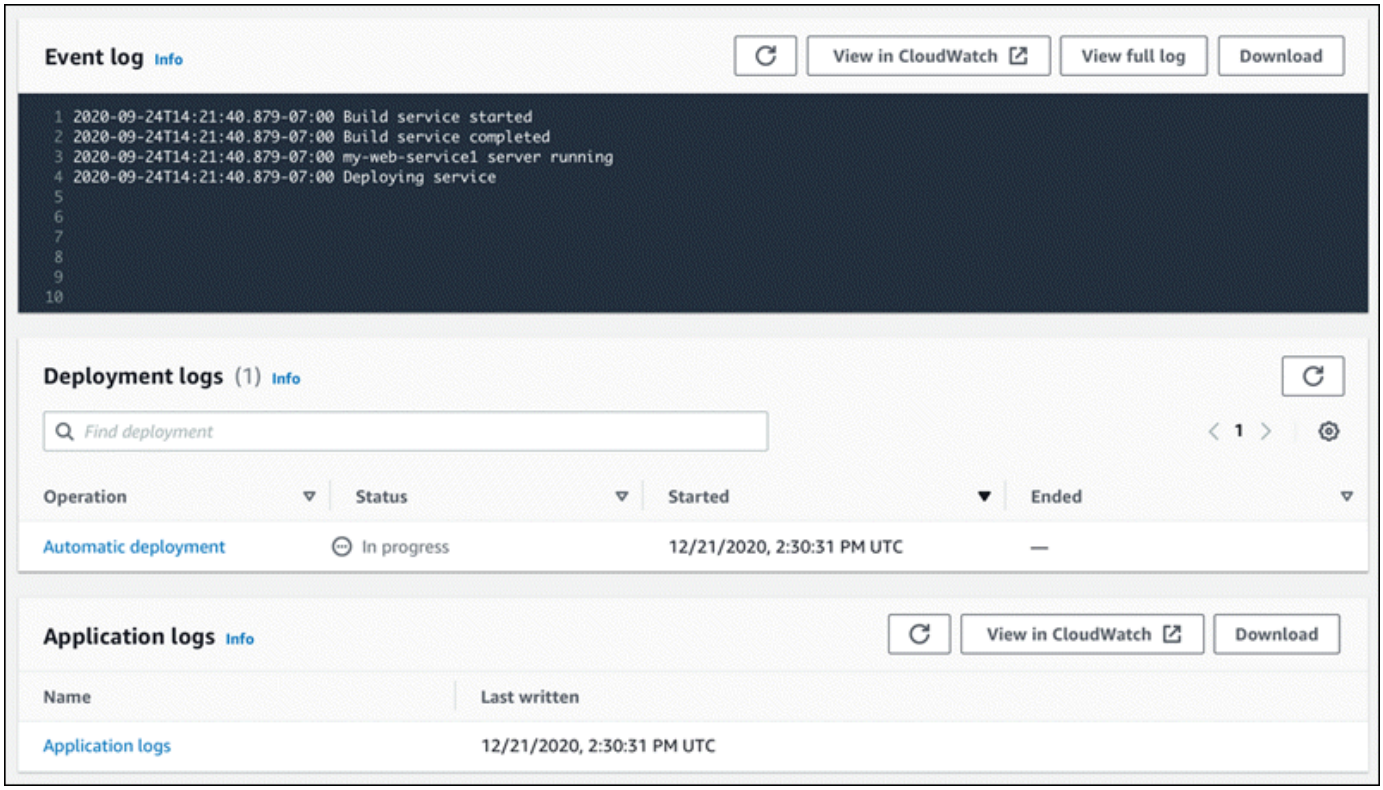
   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite den Tab Logs aus.

Die Konsole zeigt einige Arten von Protokollen in verschiedenen Abschnitten an:

- Ereignisprotokoll — Aktivität im Lebenszyklus Ihres App Runner-Dienstes. Die Konsole zeigt die neuesten Ereignisse an.
- Bereitstellungsprotokolle — Quell-Repository-Bereitstellungen für Ihren App Runner-Dienst. Die Konsole zeigt für jede Bereitstellung einen separaten Protokollstream an.
- Anwendungsprotokolle — Die Ausgabe der Webanwendung, die für Ihren App Runner-Dienst bereitgestellt wurde. Die Konsole kombiniert die Ausgabe aller laufenden Instanzen in einem einzigen Protokollstream.



4. Um nach bestimmten Bereitstellungen zu suchen, suchen Sie in der Liste der Bereitstellungsprotokolle nach unten, indem Sie einen Suchbegriff eingeben. Sie können nach jedem Wert suchen, der in der Tabelle erscheint.
5. Um den Inhalt eines Protokolls anzuzeigen, wählen Sie Vollständiges Protokoll anzeigen (Ereignisprotokoll) oder den Namen des Protokolldatenstroms (Bereitstellungs- und Anwendungsprotokolle).
6. Wählen Sie Herunterladen, um ein Protokoll herunterzuladen. Wählen Sie für einen Bereitstellungsprotokollstream zunächst einen Protokollstream aus.

7. Wählen Sie Anzeigen in CloudWatch, um die CloudWatch Konsole zu öffnen und alle Funktionen zu nutzen, um Ihre App Runner-Serviceprotokolle zu durchsuchen. Wählen Sie für einen Deployment-Log-Stream zunächst einen Log-Stream aus.

> **Note**
>
> Die CloudWatch Konsole ist besonders nützlich, wenn Sie anstelle des kombinierten Anwendungsprotokolls Anwendungsprotokolle bestimmter Instanzen anzeigen möchten.

App Runner-Servicemetriken anzeigen, an die gemeldet wurden CloudWatch

Amazon CloudWatch überwacht Ihre Amazon Web Services (AWS) -Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können CloudWatch damit Metriken sammeln und verfolgen. Dabei handelt es sich um Variablen, die Sie für Ihre Ressourcen und Anwendungen messen können. Sie können es auch verwenden, um Alarme zu erstellen, die Metriken überwachen. Wenn ein bestimmter Schwellenwert erreicht ist, werden Benachrichtigungen CloudWatch gesendet oder automatisch Änderungen an den überwachten Ressourcen vorgenommen. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

AWS App Runner sammelt eine Vielzahl von Metriken, die Ihnen einen besseren Einblick in die Nutzung, Leistung und Verfügbarkeit Ihrer App Runner-Dienste bieten. Einige Metriken verfolgen einzelne Instanzen, auf denen Ihr Webservice ausgeführt wird, während andere sich auf die allgemeine Service-Ebene beziehen. In den folgenden Abschnitten werden App Runner-Metriken aufgeführt und Sie erfahren, wie Sie sie in der App Runner-Konsole anzeigen können.

App Runner-Metriken

App Runner sammelt die folgenden Metriken zu Ihrem Service und veröffentlicht sie CloudWatch im `AWS/AppRunner` Namespace.

> **Note**
>
> Vor dem 23. August 2023 basierten die Metriken zur CPU-Auslastung und zur Speicherauslastung auf den genutzten vCPU-Einheiten und Megabyte an Arbeitsspeicher und nicht auf der prozentualen Auslastung, wie heute berechnet. Wenn Ihre Anwendung vor diesem Datum auf App Runner ausgeführt wurde und Sie die Metriken für dieses Datum

entweder im App Runner oder auf der CloudWatch Konsole erneut anzeigen möchten, werden die Messwerte in beiden Einheiten angezeigt, was auch zu Unregelmäßigkeiten führt.

Important

Sie müssen alle CloudWatch Alarme, die auf den Metrikwerten CPU-Auslastung und Speicherauslastung basieren, vor dem 23. August 2023 aktualisieren. Aktualisieren Sie die Alarme so, dass sie auf der Grundlage der prozentualen Auslastung und nicht auf der Grundlage von vCPU oder Megabyte ausgelöst werden. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Metriken auf Instance-Ebene werden für jede Instance (Skalierungseinheit) einzeln erfasst.

Was wird gemessen?	Metrik	Beschreibung
CPU utilization	`CPUUtilization`	Der prozentuale Anteil der durchschnittlichen CPU-Auslastung über Zeiträume von einer Minute an der gesamten CPU-Auslastung, die für die Dienstkonfiguration reserviert ist.
Memory utilization	`MemoryUtilization`	Der prozentuale Anteil der durchschnittlichen Speicherauslastung über Zeiträume von einer Minute am gesamten durch die Servicekonfiguration reservierten Speicher.

Service-Level-Metriken werden für den gesamten Service erfasst.

Was wird gemessen?	Metrik	Beschreibung
CPU utilization	`CPUUtilization`	Der prozentuale Anteil der aggregierten CPU-Auslastung aller Instances während eines Zeitraums von einer Minute an der gesamten CPU-

Was wird gemessen?	Metrik	Beschreibung
		Auslastung, die durch die Dienstkonfiguration reserviert ist.
Memory utilization	`MemoryUtilization`	Der prozentuale Anteil der aggregierten Speicherauslastung aller Instances während eines Zeitraums von einer Minute am gesamten durch die Dienstkonfiguration reservierten Speicher.
Concurrency	`Concurrency`	Die ungefähre Anzahl gleichzeitiger Anfragen, die vom Dienst bearbeitet werden.
HTTP request count	`Requests`	Die Anzahl der HTTP-Anfragen, die der Dienst erhalten hat.

HTTP status counts	`2xxStatusResponses`
	`4xxStatusResponses`
	`5xxStatusResponses`
		Die Anzahl der HTTP-Anfragen, die jeden Antwortstatus zurückgegeben haben, gruppiert nach Kategorien (2XX, 4XX, 5XX).

HTTP request latency	`RequestLatency`	Die Zeit in Millisekunden, die Ihr Webservice für die Verarbeitung von HTTP-Anfragen benötigt hat.
Instance counts	`ActiveInstances`	Die Anzahl der Instanzen, die HTTP-Anfragen für Ihren Service verarbeiten. **Note** Wenn die `ActiveInstances` Metrik Null anzeigt, bedeutet dies, dass keine Anfragen für den Service vorliegen. Dies bedeutet nicht, dass die Anzahl der Instanzen für Ihren Service Null ist.

App Runner-Metriken in der Konsole anzeigen

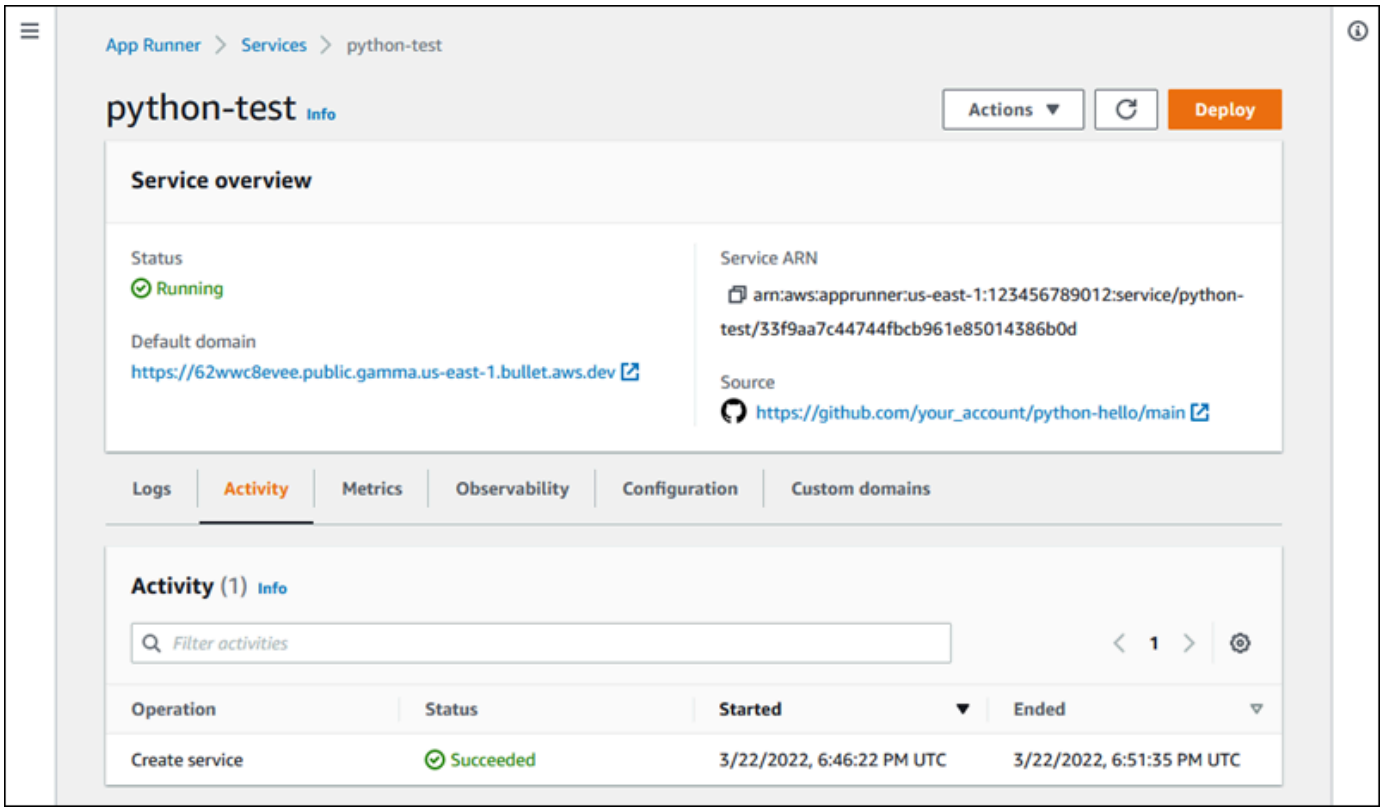
Die App Runner-Konsole zeigt die Metriken, die App Runner für Ihren Service erfasst, grafisch an und bietet weitere Möglichkeiten, sie zu untersuchen.

> **Note**
>
> Derzeit zeigt die Konsole nur Servicemetriken an. Verwenden Sie die CloudWatch Konsole, um Instanzmetriken anzuzeigen.

Um Logs für Ihren Service einzusehen

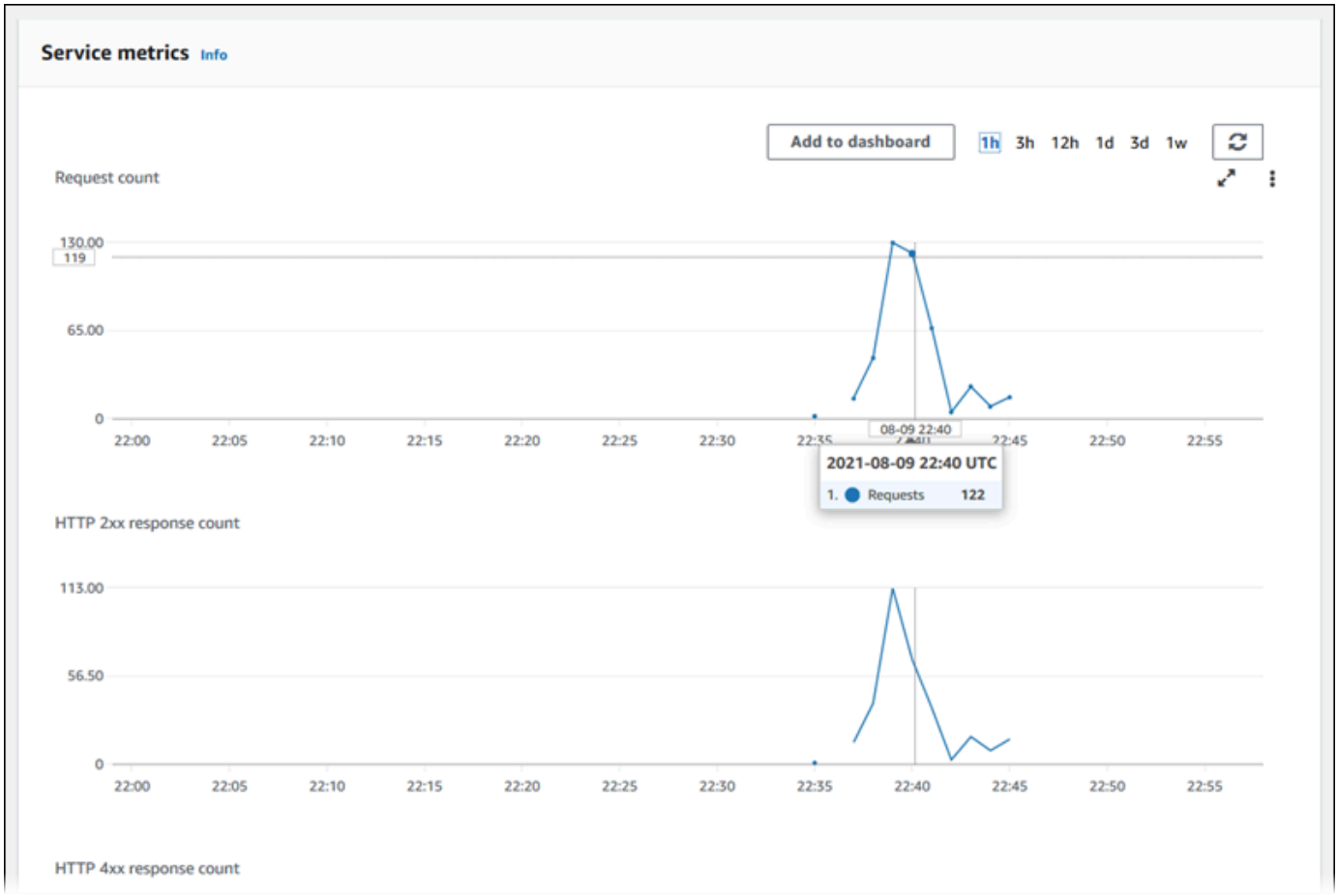
1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Wählen Sie im Navigationsbereich Dienste und dann Ihren App Runner-Dienst aus.

   In der Konsole wird das Service-Dashboard mit einer Serviceübersicht angezeigt.



3. Wählen Sie auf der Service-Dashboard-Seite den Tab Metriken aus.

   Die Konsole zeigt eine Reihe von Metrikdiagrammen an.



4. Wählen Sie eine Dauer (z. B. 12 Stunden), um die Metrikdiagramme auf den letzten Zeitraum dieser Dauer einzubeziehen.
5. Wählen Sie oben in einem der Diagrammbereiche die Option Zum Dashboard hinzufügen aus oder verwenden Sie das Menü eines beliebigen Diagramms, um die relevanten Metriken zur weiteren Untersuchung zu einem Dashboard in der CloudWatch Konsole hinzuzufügen.

Umgang mit App Runner-Ereignissen in EventBridge

Mit Amazon EventBridge können Sie ereignisgesteuerte Regeln einrichten, die einen Stream von Echtzeitdaten aus Ihrem AWS App Runner Service auf bestimmte Muster hin überwachen. Wenn ein Muster für eine Regel übereinstimmt, wird eine Aktion in einem Ziel wie AWS Lambda Amazon ECS und Amazon AWS Batch SNS EventBridge initiiert. Sie können beispielsweise eine Regel für das Versenden von E-Mail-Benachrichtigungen festlegen, indem Sie ein Amazon SNS SNS-Thema signalisieren, wenn eine Bereitstellung für Ihren Service fehlschlägt. Oder du kannst eine Lambda-Funktion einrichten, um einen Slack-Channel zu benachrichtigen, wenn ein Service-

Update fehlschlägt. Weitere Informationen zu EventBridge finden Sie im EventBridge Amazon-Benutzerhandbuch.

App Runner sendet die folgenden Ereignistypen an EventBridge

- Änderung des Dienststatus — Eine Änderung des Status eines App Runner-Dienstes. Zum Beispiel wurde ein Dienststatus in geändert`DELETE_FAILED`.
- Änderung des Status des Dienstvorgangs — Eine Änderung des Status eines langen, asynchronen Vorgangs in einem App Runner-Dienst. Zum Beispiel, wenn mit der Erstellung eines Dienstes begonnen wurde, ein Dienstupdate erfolgreich abgeschlossen wurde oder eine Dienstbereitstellung mit Fehlern abgeschlossen wurde.

Eine EventBridge Regel erstellen, die auf App Runner-Ereignisse reagiert

Ein EventBridge Ereignis ist ein Objekt, das einige EventBridge Standardfelder wie den AWS Quelldienst und den Detailtyp (Ereignis) sowie einen ereignisspezifischen Satz von Feldern mit den Ereignisdetails definiert. Um eine EventBridge Regel zu erstellen, verwenden Sie die EventBridge Konsole, um ein Ereignismuster zu definieren (welche Ereignisse verfolgt werden sollen) und eine Zielaktion (was bei einem Spiel geschehen soll) festzulegen. Ein Ereignismuster ähnelt den Ereignissen, denen es entspricht. Sie geben eine Teilmenge von Feldern an, die abgeglichen werden sollen, und für jedes Feld geben Sie eine Liste möglicher Werte an. Dieses Thema enthält Beispiele für App Runner-Ereignisse und -Ereignismuster.

Weitere Informationen zum Erstellen von EventBridge Regeln finden Sie unter Regel für einen AWS Service erstellen im EventBridge Amazon-Benutzerhandbuch.

> **Note**
>
> Einige Dienste unterstützen vordefinierte Muster in EventBridge. Dies vereinfacht die Erstellung eines Ereignismusters. Sie wählen Feldwerte in einem Formular aus und EventBridge generieren das Muster für Sie. Derzeit unterstützt App Runner keine vordefinierten Muster. Sie müssen das Muster als JSON-Objekt eingeben. Sie können die Beispiele in diesem Thema als Ausgangspunkt verwenden.

Beispiele für App Runner-Ereignisse

Dies sind einige Beispiele für Ereignisse, an die App Runner sendet EventBridge.

- Ein Ereignis zur Änderung des Dienststatus. Insbesondere ein Dienst, der vom in `OPERATION_IN_PROGRESS` den `RUNNING` Status geändert wurde.

```
{
  "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"
  }
}
```

- Ein Ereignis zur Änderung des Betriebsstatus. Insbesondere ein `UpdateService` Vorgang, der erfolgreich abgeschlossen wurde.

```
{
  "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"
  }
}
```

Beispiele für App Runner-Ereignismuster

Die folgenden Beispiele zeigen Ereignismuster, die Sie in EventBridge Regeln verwenden können, um einem oder mehreren App Runner-Ereignissen zuzuordnen. Ein Ereignismuster ähnelt einem Ereignis. Schließen Sie nur die Felder ein, die Sie zuordnen möchten, und stellen Sie für jedes Feld eine Liste statt eines Skalars bereit.

- Ordnet alle Ereignisse zur Änderung des Dienststatus für Dienste eines bestimmten Kontos zu, bei dem sich der Dienst nicht mehr im `RUNNING` Status befindet.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- Ordnet alle Ereignisse zur Änderung des Vorgangsstatus für Dienste eines bestimmten Kontos zu, bei dem der Vorgang fehlgeschlagen ist.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
```

```
    ]
  }
}
```

Referenz zum App Runner-Ereignis

Änderung des Dienststatus

Ein Ereignis zur Änderung des Servicestatus wurde auf `detail-type` gesetztAppRunner `Service Status Change`. Es hat die folgenden Detailfelder und Werte:

```
"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"
```

Änderung des Betriebsstatus

Ein Ereignis zur Änderung des Betriebsstatus wurde auf `detail-type` gesetztAppRunner `Service Operation Status Change`. Es hat die folgenden Detailfelder und Werte:

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

In der folgenden Tabelle sind alle möglichen Statuscodes und zugehörigen Meldungen aufgeführt.

Status	Fehlermeldung
`CreateServiceStarted`	Die Erstellung des Dienstes wurde gestartet.
`CreateServiceCompletedSuccessfully`	Die Erstellung des Dienstes wurde erfolgreich abgeschlossen.

Status	Fehlermeldung
`CreateServiceFailed`	Die Diensterstellung ist fehlgeschlagen. Einzelheiten finden Sie in den Dienstprotokollen.
`DeleteServiceStarted`	Das Löschen des Dienstes wurde gestartet.
`DeleteServiceCompletedSuccessfully`	Das Löschen des Dienstes wurde erfolgreich abgeschlossen.
`DeleteServiceFailed`	Das Löschen des Dienstes ist fehlgeschlagen.
`UpdateServiceStarted`	
`UpdateServiceCompletedSuccessfully`	Das Service-Update wurde erfolgreich abgeschlossen. Neue Anwendung und Konfiguration wurden bereitgestellt.
	Das Service-Update wurde erfolgreich abgeschlossen. Die neue Konfiguration wurde bereitgestellt.
`UpdateServiceFailed`	Das Service-Update ist fehlgeschlagen. Einzelheiten finden Sie in den Serviceprotokollen.
`DeploymentStarted`	Die Bereitstellung wurde gestartet.
`DeploymentCompletedSuccessfully`	Die Bereitstellung wurde erfolgreich abgeschlossen.
`DeploymentFailed`	Bereitstellung fehlgeschlagen. Einzelheiten finden Sie in den Serviceprotokollen.
`PauseServiceStarted`	Die Dienstpause wurde gestartet.
`PauseServiceCompletedSuccessfully`	Die Dienstpause wurde erfolgreich abgeschlossen.
`PauseServiceFailed`	Die Dienstpause ist fehlgeschlagen.
`ResumeServiceStarted`	Die Wiederaufnahme des Dienstes wurde gestartet.

Status	Fehlermeldung
`ResumeServiceCompletedSuccessfully`	Die Wiederaufnahme des Dienstes wurde erfolgreich abgeschlossen.
`ResumeServiceFailed`	Die Wiederaufnahme des Dienstes ist fehlgeschlagen.

App Runner API-Aufrufe protokollieren mit AWS CloudTrail

App Runner ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in App Runner ausgeführt wurden. CloudTrailerfasst alle API-Aufrufe für App Runner als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der App Runner-Konsole und Codeaufrufen für die App Runner-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für App Runner. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an App Runner gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Informationen zu App Runner finden Sie unter CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn in App Runner Aktivitäten auftreten, wird diese Aktivität zusammen mit anderen CloudTrail AWS Dienstereignissen im Ereignisverlauf in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem anzeigen, suchen und herunterladen AWS-Konto. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich der Ereignisse für App Runner, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail

Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle App Runner-Aktionen werden von der AWS App Runner API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Aufrufe der `StartDeployment` Aktionen `CreateServiceDeleteConnection`, und generieren beispielsweise Einträge in den CloudTrail Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Gibt an, ob die Anforderung mit Root- oder IAM-Benutzer-Anmeldeinformationen ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Grundlegendes zu App Runner-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion sowie Anforderungsparameter. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateService` Aktion demonstriert.

> **Note**
>
> Aus Sicherheitsgründen werden einige Eigenschaftswerte in den Protokollen geschwärzt und durch den Text `HIDDEN_DUE_TO_SECURITY_REASONS` ersetzt. Dies verhindert die unbeabsichtigte Offenlegung geheimer Informationen. Sie können jedoch immer noch sehen, dass diese Eigenschaften in der Anfrage übergeben oder in der Antwort zurückgegeben wurden.

Beispiel für einen CloudTrail Protokolleintrag für die **`CreateService`** App Runner-Aktion

```
{
  "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"
}
```

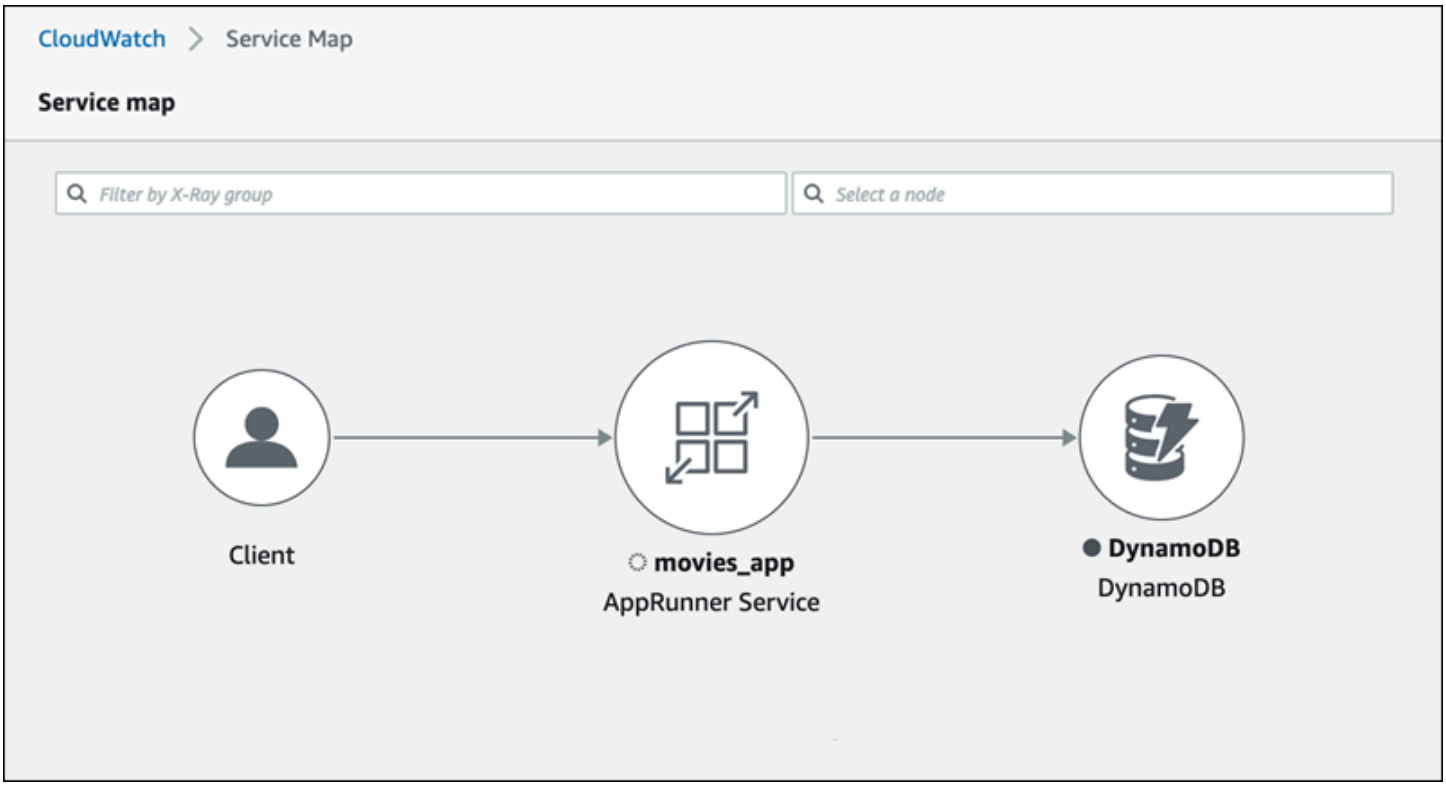

Tracing für Ihre App Runner-Anwendung mit X-Ray

AWS X-Ray ist ein Dienst, der Daten über Anfragen sammelt, die Ihre Anwendung bearbeitet, und Tools bereitstellt, mit denen Sie diese Daten anzeigen, filtern und Einblicke in sie gewinnen können,

um Probleme und Optimierungsmöglichkeiten zu identifizieren. Zu jeder verfolgten Anfrage an Ihre Anwendung können Sie detaillierte Informationen nicht nur über die Anfrage und Antwort abrufen, sondern auch über Aufrufe, die Ihre Anwendung an nachgelagerte AWS Ressourcen, Microservices, Datenbanken und das HTTP-Web sendet. APIs

X-Ray verwendet Trace-Daten aus den AWS Ressourcen, die Ihre Cloud-Anwendungen unterstützen, um ein detailliertes Service-Diagramm zu erstellen. Das Servicediagramm zeigt den Client, Ihren Frontend-Service und die Backend-Services, die Ihr Frontend-Services für die Verarbeitung von Anforderungen und Persistenzdaten aufruft. Sie können das Servicediagramms zur Ermittlung von Engpässen, Latenzspitzen und anderen Problemen verwenden, die Sie zur Verbesserung der Leistung Ihrer Anwendungen beheben können.

Weitere Informationen zu X-Ray finden Sie im Entwicklerhandbuch für AWS X-Ray.



Instrumentieren Sie Ihre Anwendung für die Rückverfolgung

Instrumentieren Sie Ihre App Runner-Dienstanwendung für die Ablaufverfolgung mithilfe OpenTelemetryeiner portablen Telemetriespezifikation. Derzeit unterstützt App Runner AWS Distro for OpenTelemetry (ADOT), eine OpenTelemetry Implementierung, die Telemetriedaten mithilfe von Diensten sammelt und präsentiert. AWS X-Ray implementiert die Tracing-Komponente.

Abhängig vom spezifischen ADOT-SDK, das Sie in Ihrer Anwendung verwenden, unterstützt ADOT bis zu zwei Instrumentierungsansätze: automatisch und manuell. Weitere Informationen zur Instrumentierung mit Ihrem SDK finden Sie in der ADOT-Dokumentation. Wählen Sie im Navigationsbereich Ihr SDK aus.

Runtime-Setup

Im Folgenden finden Sie allgemeine Anweisungen zur Runtime-Setup, mit denen Sie Ihre App Runner-Dienstanwendung für die Ablaufverfolgung instrumentieren können.

So richten Sie die Ablaufverfolgung für Ihre Laufzeit ein

1. Folgen Sie den Anweisungen für Ihre Laufzeit in AWS Distro for OpenTelemetry (ADOT), um Ihre Anwendung zu instrumentieren.
2. Installieren Sie die erforderlichen `OTEL` Abhängigkeiten im `build` Abschnitt der `apprunner.yaml` Datei, wenn Sie das Quellcode-Repository verwenden, oder im Dockerfile, wenn Sie ein Container-Image verwenden.
3. Richten Sie Ihre Umgebungsvariablen in der `apprunner.yaml` Datei ein, wenn Sie das Quellcode-Repository verwenden, oder im Dockerfile, wenn Sie ein Container-Image verwenden.

   Example Umgebungsvariablen

   > **Note**
   >
   > Das folgende Beispiel listet die wichtigen Umgebungsvariablen auf, die der `apprunner.yaml` Datei hinzugefügt werden sollen. Fügen Sie diese Umgebungsvariablen zu Ihrem Dockerfile hinzu, wenn Sie ein Container-Image verwenden. Jede Laufzeit kann jedoch ihre eigenen Eigenheiten haben, und Sie müssen der folgenden Liste möglicherweise weitere Umgebungsvariablen hinzufügen. Weitere Informationen zu Ihren runtime-spezifischen Anweisungen und Beispiele zur Einrichtung Ihrer Anwendung für Ihre Runtime finden Sie unter AWS Distribution for OpenTelemetry und Go to your runtime unter Getting Started.

```
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`ist eine wichtige Umgebungsvariable für App Runner, da der App Runner Otel Collector keine Metrikprotokollierung akzeptiert. Er akzeptiert nur die Nachverfolgung von Metriken.

Beispiel für die Einrichtung einer Laufzeit

Das folgende Beispiel zeigt die automatische Instrumentierung Ihrer Anwendung mit dem ADOT Python SDK. Das SDK erzeugt automatisch Spans mit Telemetriedaten, die die Werte beschreiben, die von den Python-Frameworks in Ihrer Anwendung verwendet werden, ohne eine einzige Zeile Python-Code hinzuzufügen. Sie müssen nur ein paar Zeilen in zwei Quelldateien hinzufügen oder ändern.

Fügen Sie zunächst einige Abhängigkeiten hinzu, wie im folgenden Beispiel gezeigt.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

Instrumentieren Sie dann Ihre Anwendung. Die Art und Weise, wie Sie dies tun, hängt von Ihrer Service-Quelle ab — Quellbild oder Quellcode.

Source image

Wenn es sich bei Ihrer Servicequelle um ein Image handelt, können Sie das Dockerfile, das die Erstellung Ihres Container-Images und die Ausführung der Anwendung im Image steuert, direkt instrumentieren. Das folgende Beispiel zeigt ein instrumentiertes Dockerfile für eine Python-Anwendung. Hinzufügungen zur Instrumentierung sind fett hervorgehoben.

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

Wenn Ihre Dienstquelle ein Repository ist, das Ihre Anwendungsquelle enthält, instrumentieren Sie Ihr Image indirekt mithilfe der Einstellungen der App Runner-Konfigurationsdatei. Diese Einstellungen steuern das Dockerfile, das App Runner generiert und verwendet, um das Image für Ihre Anwendung zu erstellen. Das folgende Beispiel zeigt eine instrumentierte App Runner-Konfigurationsdatei für eine Python-Anwendung. Hinzufügungen zur Instrumentierung sind fett hervorgehoben.

Example apprunner.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'
```

X-Ray-Berechtigungen zu Ihrer App Runner-Dienstinstanzrolle hinzufügen

Um X-Ray Tracing mit Ihrem App Runner-Dienst verwenden zu können, müssen Sie den Instanzen des Dienstes Berechtigungen für die Interaktion mit dem X-Ray-Dienst erteilen. Dazu ordnen Sie Ihrem Service eine Instanzrolle zu und fügen eine verwaltete Richtlinie mit X-Ray-Berechtigungen hinzu. Weitere Informationen zu einer App Runner-Instanzrolle finden Sie unterthe section called “Instanzrolle”. Fügen Sie die `AWSXRayDaemonWriteAccess` verwaltete Richtlinie zu Ihrer Instanzrolle hinzu und weisen Sie sie bei der Erstellung Ihrem Service zu.

X-Ray-Tracing für Ihren App Runner-Dienst aktivieren

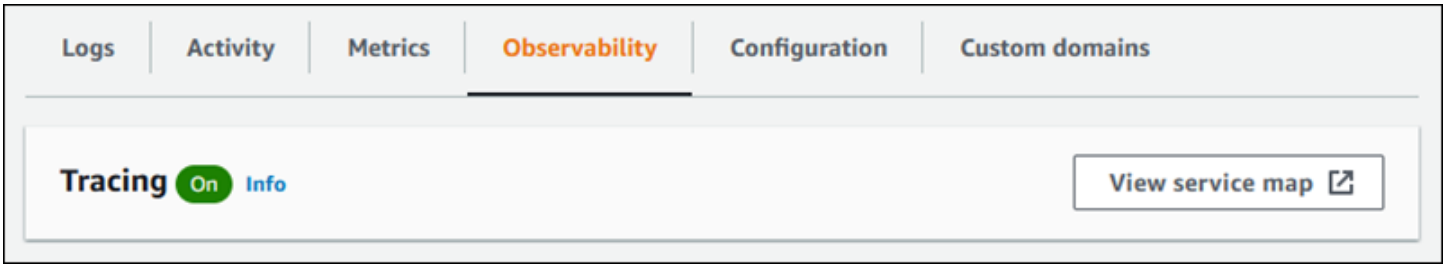
Wenn Sie einen Dienst erstellen, deaktiviert App Runner die Ablaufverfolgung standardmäßig. Sie können X-Ray Tracing für Ihren Service als Teil der Konfiguration von Observability aktivieren. Weitere Informationen finden Sie unter the section called “Beobachtbarkeit verwalten”.

Wenn Sie die App Runner-API oder die verwenden AWS CLI, enthält das TraceConfigurationObjekt innerhalb des ObservabilityConfigurationRessourcenobjekts Tracing-Einstellungen. Um die Ablaufverfolgung weiterhin zu deaktivieren, geben Sie kein `TraceConfiguration` Objekt an.

Stellen Sie sowohl bei der Konsole als auch bei der API sicher, dass Sie Ihre im vorherigen Abschnitt beschriebene Instanzrolle Ihrem App Runner-Dienst zuordnen.

Sehen Sie sich die X-Ray-Tracing-Daten für Ihren App Runner-Dienst an

Wählen Sie auf der Registerkarte Observability der Service-Dashboard-Seite in der App Runner-Konsole die Option Service Map anzeigen aus, um zur CloudWatch Amazon-Konsole zu navigieren.



Verwenden Sie die CloudWatch Amazon-Konsole, um Service-Maps und Traces für Anfragen anzuzeigen, die Ihre Anwendung bearbeitet. Service Maps zeigen Informationen wie die Latenz von Anfragen und Interaktionen mit anderen Anwendungen und AWS Diensten. Mit den benutzerdefinierten Anmerkungen, die Sie Ihrem Code hinzufügen, können Sie ganz einfach nach Spuren suchen. Weitere Informationen finden Sie unter Verwendung ServiceLens zur Überwachung des Zustands Ihrer Anwendungen im CloudWatch Amazon-Benutzerhandbuch.

Eine AWS WAF Web-ACL mit Ihrem Service verknüpfen

AWS WAF ist eine Firewall für Webanwendungen, mit der Sie Ihren App Runner-Dienst sichern können. Mit AWS WAF Web-Zugriffskontrolllisten (Web ACLs) können Sie Ihre App Runner-Dienstendpunkte vor gängigen Web-Exploits und unerwünschten Bots schützen.

Eine Web-ACL bietet Ihnen eine detaillierte Kontrolle über alle eingehenden Webanfragen an Ihren App Runner-Dienst. Sie können in einer Web-ACL Regeln definieren, um den Webverkehr zuzulassen, zu blockieren oder zu überwachen, um sicherzustellen, dass nur autorisierte und legitime Anfragen Ihre Webanwendungen erreichen und. APIs Sie können die Web-ACL-Regeln an Ihre spezifischen Geschäfts- und Sicherheitsanforderungen anpassen. Weitere Informationen zur Infrastruktursicherheit und zu bewährten Methoden für die Netzwerkanwendung ACLs finden Sie unter Steuern des Netzwerkverkehrs im Amazon VPC-Benutzerhandbuch.

> **Important**
>
> Quell-IP-Regeln für private App Runner-Services, die mit dem WAF-Web verknüpft sind, entsprechen ACLs nicht den IP-basierten Regeln. Dies liegt daran, dass wir derzeit die Weiterleitung von Quell-IP-Daten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs

Ablauf eingehender Webanfragen

Wenn eine AWS WAF Web-ACL einem App Runner-Dienst zugeordnet ist, durchlaufen eingehende Webanfragen den folgenden Prozess:

1. App Runner leitet den Inhalt der ursprünglichen Anfrage an AWS WAF weiter.
2. AWS WAF untersucht die Anfrage und vergleicht ihren Inhalt mit den Regeln, die Sie in Ihrer Web-ACL angegeben haben.
3. AWS WAF Gibt auf der Grundlage der Prüfung eine `block` Oder-Antwort an `allow` App Runner zurück.
   - Wenn eine `allow` Antwort zurückgegeben wird, leitet App Runner die Anfrage an Ihre Anwendung weiter.

- Wenn eine `block` Antwort zurückgegeben wird, blockiert App Runner, dass die Anfrage Ihre Webanwendung erreicht. Es leitet die `block` Antwort von AWS WAF an Ihre Anwendung weiter.

  > **Note**
  >
  > Standardmäßig blockiert App Runner die Anfrage, wenn keine Antwort von AWS WAF zurückgegeben wird.

Weitere Informationen zum AWS WAF Web ACLs finden Sie unter Web Access Control Lists (Web ACLs) im AWS WAF Developer Guide.

> **Note**
>
> Sie zahlen den AWS WAF Standardpreis. Für die Nutzung des AWS WAF Internets ACLs für Ihre App Runner-Dienste fallen keine zusätzlichen Kosten an. Weitere Informationen zur Preisgestaltung finden Sie unter AWS WAF Preise.

WAF-Web mit Ihrem App ACLs Runner-Dienst verknüpfen

Im Folgenden finden Sie den allgemeinen Prozess zur Verknüpfung einer AWS WAF Web-ACL mit Ihrem App Runner-Dienst:

1. Erstellen Sie eine Web-ACL in der AWS WAF Konsole. Weitere Informationen finden Sie unter Erstellen einer Web-ACL im AWS WAF Entwicklerhandbuch.
2. Aktualisieren Sie Ihre AWS Identity and Access Management (IAM-) Berechtigungen für AWS WAF. Weitere Informationen finden Sie unter -Berechtigungen.
3. Ordnen Sie die Web-ACL mithilfe einer der folgenden Methoden dem App Runner-Dienst zu:
   - App Runner-Konsole: Ordnen Sie mithilfe der App Runner-Konsole eine vorhandene Web-ACL zu, wenn Sie einen App Runner-Dienst erstellen oder aktualisieren. Anweisungen finden Sie unter AWS WAF Web verwalten ACLs.
   - AWS WAF Konsole: Ordnen Sie die Web-ACL mithilfe der AWS WAF Konsole für einen vorhandenen App Runner-Dienst zu. Weitere Informationen finden Sie unter Zuordnen oder Aufheben der Zuordnung einer Web-ACL zu einer AWS-Ressource im AWS WAF Entwicklerhandbuch.

- AWS CLI: Ordnen Sie die Web-ACL über die Öffentlichkeit zu. AWS WAF APIs Weitere Informationen zu AWS WAF Public APIs finden Sie unter AssociateWebACL im AWS WAF API-Referenzhandbuch.

Überlegungen

- Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, entsprechen ACLs nicht den IP-basierten Regeln. Dies liegt daran, dass wir derzeit die Weiterleitung von Quell-IP-Daten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs
- Ein App Runner-Dienst kann nur einer Web-ACL zugeordnet werden. Sie können jedoch eine Web-ACL mehreren App Runner-Diensten und mehreren AWS Ressourcen zuordnen. Beispiele hierfür sind Amazon Cognito Cognito-Benutzerpools und Application Load Balancer Balancer-Ressourcen.
- Wenn Sie eine Web-ACL erstellen, vergeht eine kurze Zeit, bis die Web-ACL vollständig verbreitet ist und App Runner zur Verfügung steht. Die Übertragungszeit kann zwischen einigen Sekunden und mehreren Minuten liegen. AWS WAF gibt a zurück`WAFUnavailableEntityException`, wenn Sie versuchen, eine Web-ACL zuzuordnen, bevor sie vollständig weitergegeben wurde.

  Wenn Sie den Browser aktualisieren oder die App Runner-Konsole verlassen, bevor die Web-ACL vollständig weitergegeben wurde, schlägt die Zuordnung fehl. Sie können jedoch innerhalb der App Runner-Konsole navigieren.
- AWS WAF gibt einen `WAFNonexistentItemException` Fehler zurück, wenn Sie eine der folgenden Optionen AWS WAF APIs für einen App Runner-Dienst aufrufen, der sich in einem ungültigen Zustand befindet:
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  Zu den ungültigen Status für Ihren App Runner-Dienst gehören:
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`

- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`Der Status ist nur dann ungültig, wenn Ihr App Runner-Dienst gelöscht wird.

- Ihre Anfrage könnte zu einer Nutzlast führen, die die Grenzen dessen, was überprüft werden AWS WAF kann, überschreitet. Weitere Informationen zum Umgang AWS WAF mit übergroßen Anfragen von App Runner finden Sie unter Behandlung von übergroßen Anforderungskomponenten im AWS WAF Entwicklerhandbuch. Dort erfahren Sie, wie mit übergroßen Anfragen von App Runner AWS WAF umgegangen wird.
- Wenn Sie keine geeigneten Regeln festlegen oder sich Ihre Datenverkehrsmuster ändern, ist eine Web-ACL möglicherweise nicht so effektiv, um Ihre Anwendung zu schützen.

Berechtigungen

Um mit einer Web-ACL zu arbeiten AWS App Runner, fügen Sie die folgenden IAM-Berechtigungen hinzu für AWS WAF:

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

Weitere Informationen zu IAM-Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.

Im Folgenden finden Sie ein Beispiel für die aktualisierte IAM-Richtlinie für. AWS WAF Diese IAM-Richtlinie beinhaltet die erforderlichen Berechtigungen für die Arbeit mit einem App Runner-Dienst.

Example

```
{
  {
   "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**
>
> Auch wenn Sie IAM-Berechtigungen erteilen müssen, sind die aufgelisteten Aktionen nur für Berechtigungen bestimmt und entsprechen keiner API-Operation.

Verwaltung des AWS WAF Webs ACLs

Verwalten Sie das AWS WAF Web ACLs für Ihren App Runner-Dienst mit einer der folgenden Methoden:

- the section called “App Runner-Konsole”
- the section called “AWS CLI”

App Runner-Konsole

Wenn Sie in der App Runner-Konsole einen Dienst erstellen oder einen vorhandenen aktualisieren, können Sie eine AWS WAF Web-ACL zuordnen oder die Zuordnung aufheben.

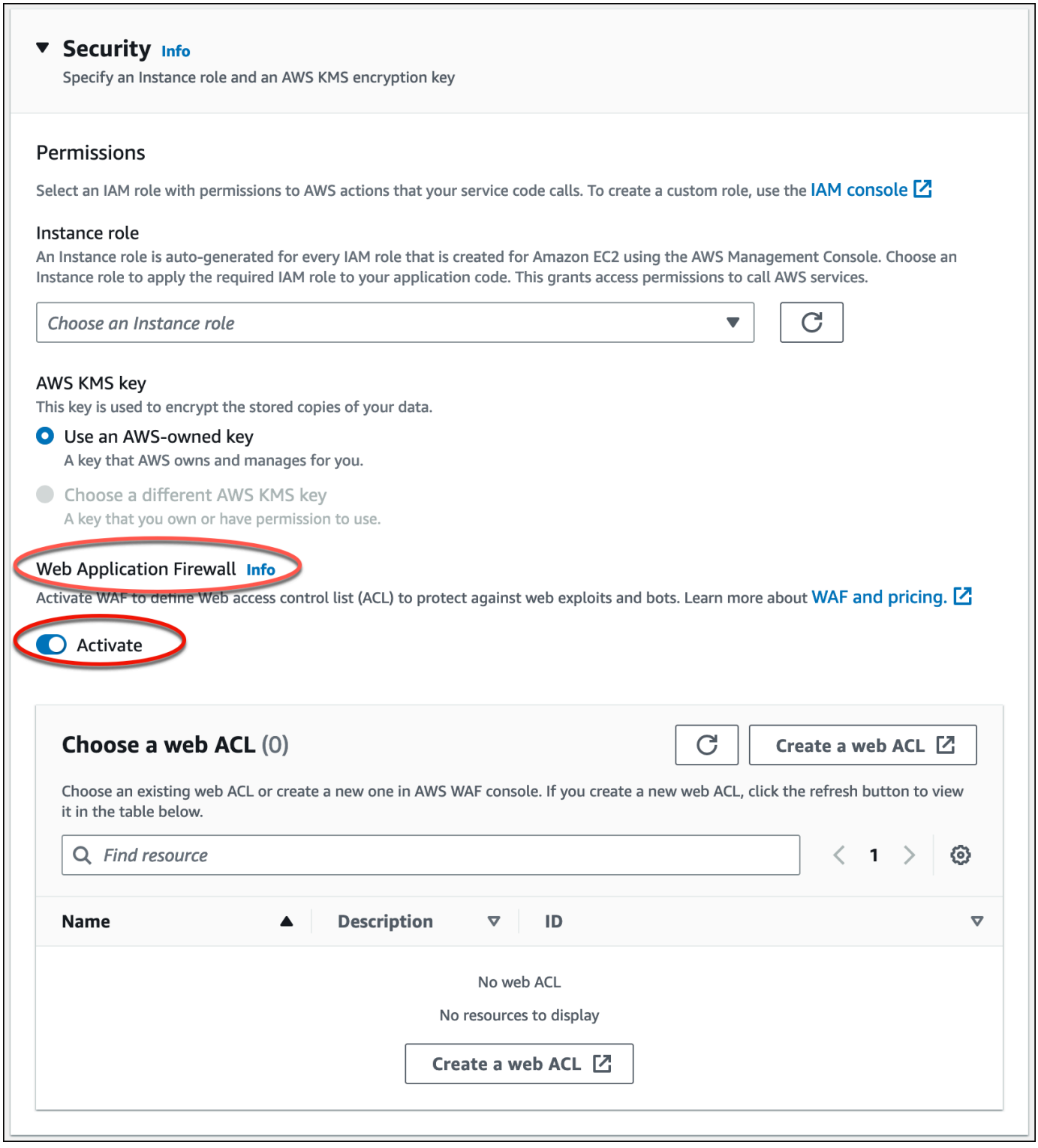
> **Note**
>
> - Ein App Runner-Dienst kann nur einer Web-ACL zugeordnet werden. Sie können jedoch eine Web-ACL zusätzlich zu anderen AWS Ressourcen mit mehr als einem App Runner-Dienst verknüpfen.
> - Bevor Sie eine Web-ACL zuordnen, stellen Sie sicher, dass Sie Ihre IAM-Berechtigungen für AWS WAF aktualisieren. Weitere Informationen finden Sie unter -Berechtigungen.

Web-ACL zuordnen AWS WAF

> **Important**
>
> Quell-IP-Regeln für private App Runner-Dienste, die mit WAF Web verknüpft sind, entsprechen ACLs nicht IP-basierten Regeln. Dies liegt daran, dass wir derzeit die Weiterleitung von Quell-IP-Daten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs

Um eine Web-ACL zuzuordnen AWS WAF

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre aus AWS-Region.
2. Je nachdem, ob Sie einen Dienst erstellen oder aktualisieren, führen Sie einen der folgenden Schritte aus:
   - Wenn Sie einen neuen Dienst erstellen, wählen Sie App Runner-Dienst erstellen und gehen Sie zu Dienst konfigurieren.
   - Wenn Sie einen vorhandenen Dienst aktualisieren, wählen Sie den Tab Konfiguration und dann unter Dienst konfigurieren die Option Bearbeiten aus.
3. Gehen Sie unter Sicherheit zu Web Application Firewall.
4. Wählen Sie die Umschaltfläche Aktivieren, um die Optionen anzuzeigen.



5. Führen Sie einen der folgenden Schritte aus:

   - Um eine bestehende Web-ACL zuzuordnen: Wählen Sie die erforderliche Web-ACL aus der Tabelle Wählen Sie eine Web-ACL aus, die Sie Ihrem App Runner-Dienst zuordnen möchten.

- Um eine neue Web-ACL zu erstellen: Wählen Sie Web-ACL erstellen, um mit der AWS WAF Konsole eine neue Web-ACL zu erstellen. Weitere Informationen finden Sie unter Erstellen einer Web-ACL im AWS WAF Entwicklerhandbuch.

  1. Klicken Sie auf die Schaltfläche „Aktualisieren“, um die neu erstellte Web-ACL in der Tabelle Web-ACL auswählen anzuzeigen.
  2. Wählen Sie die erforderliche Web-ACL aus.

6. Wählen Sie Weiter, wenn Sie einen neuen Service erstellen, oder Änderungen speichern, wenn Sie einen vorhandenen Service aktualisieren. Die ausgewählte Web-ACL ist mit Ihrem App Runner-Dienst verknüpft.
7. Um die Web-ACL-Zuordnung zu überprüfen, wählen Sie die Registerkarte Konfiguration Ihres Dienstes und gehen Sie zu Dienst konfigurieren. Scrollen Sie unter Sicherheit zur Web Application Firewall, um die Details der Web-ACL zu sehen, die Ihrem Service zugeordnet ist.

> **Note**
>
> Wenn Sie eine Web-ACL erstellen, vergeht eine kurze Zeit, bis die Web-ACL vollständig verbreitet ist und App Runner zur Verfügung steht. Die Übertragungszeit kann zwischen einigen Sekunden und mehreren Minuten liegen. AWS WAF gibt a zurück`WAFUnavailableEntityException`, wenn Sie versuchen, eine Web-ACL zuzuordnen, bevor sie vollständig weitergegeben wurde.
> Wenn Sie den Browser aktualisieren oder die App Runner-Konsole verlassen, bevor die Web-ACL vollständig weitergegeben wurde, schlägt die Zuordnung fehl. Sie können jedoch innerhalb der App Runner-Konsole navigieren.

Aufheben der Zuordnung einer Web-ACL AWS WAF

Sie können die Zuordnung zu AWS WAF Websites ACl , die Sie nicht mehr benötigen, trennen, indem Sie Ihren App Runner-Dienst aktualisieren.

So trennen Sie die Zuordnung zu einem Web AWS WAF ACl

1. Öffnen Sie die App Runner-Konsole und wählen Sie in der Liste der Regionen Ihre AWS-Region aus.
2. Gehen Sie zur Registerkarte Konfiguration des Dienstes, den Sie aktualisieren möchten, und wählen Sie unter Dienst konfigurieren die Option Bearbeiten aus.

3. Gehen Sie unter Sicherheit zur Web Application Firewall.
4. Deaktivieren Sie die Umschaltfläche Aktivieren. Sie erhalten eine Nachricht zur Bestätigung des Löschvorgangs.
5. Wählen Sie Bestätigen aus. Die Web-ACL ist von Ihrem App Runner-Dienst getrennt.

> **Note**
>
> - Wenn Sie Ihren Service mit einer anderen Web-ACL verknüpfen möchten, wählen Sie eine Web-ACL aus der Tabelle Web-ACL auswählen aus. App Runner trennt die Verknüpfung der aktuellen Web-ACL und startet den Vorgang zur Verknüpfung mit der ausgewählten Web-ACL.
> - Wenn keine anderen App Runner-Dienste oder -Ressourcen eine getrennte Web-ACL verwenden, sollten Sie erwägen, die Web-ACL zu löschen. Andernfalls werden Ihnen weiterhin Kosten entstehen. Weitere Informationen über die Preise finden Sie unter AWS WAF – Preise. Anweisungen zum Löschen einer Web-ACL finden Sie unter DeleteWebACL in der AWS WAF API-Referenz.
> - Sie können keine Web-ACL löschen, die mit anderen aktiven App Runner-Diensten oder anderen Ressourcen verknüpft ist.

AWS CLI

Sie können eine AWS WAF Web-ACL mithilfe der AWS WAF öffentlichen APIs URL zuordnen oder deren Zuordnung aufheben. Der App Runner-Dienst, dem Sie eine Web-ACL zuordnen oder die Zuordnung aufheben möchten, muss sich in einem gültigen Status befinden.

AWS WAF gibt einen `WAFNonexistentItemException` Fehler zurück, wenn Sie eine der folgenden Optionen AWS WAF APIs für einen App Runner-Dienst aufrufen, der sich in einem ungültigen Zustand befindet:

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

Zu den ungültigen Status für Ihren App Runner-Dienst gehören:

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`Der Status ist nur dann ungültig, wenn Ihr App Runner-Dienst gelöscht wird.

Weitere Informationen zu AWS WAF public APIs finden Sie im AWS WAF API-Referenzhandbuch.

> **Note**
>
> Aktualisieren Sie Ihre IAM-Berechtigungen für AWS WAF. Weitere Informationen finden Sie unter -Berechtigungen.

AWS WAF Web-ACL zuordnen mit AWS CLI

> **Important**
>
> Quell-IP-Regeln für private App Runner-Dienste, die mit WAF Web verknüpft sind, halten sich ACLs nicht an IP-basierte Regeln. Dies liegt daran, dass wir derzeit die Weiterleitung von Quell-IP-Daten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs

Um eine Web-ACL zuzuordnen AWS WAF

1. Erstellen Sie eine AWS WAF Web-ACL für Ihren Service mit Ihren bevorzugten Regelaktionen `Allow` oder `Block` den Webanfragen an Ihren Service. Weitere Informationen AWS WAF APIs dazu finden Sie unter CreateWebACL im AWS WAF API-Referenzhandbuch.

Example Erstellen Sie eine Web-ACL — Anfrage

```
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. Ordnen Sie die von Ihnen erstellte Web-ACL mithilfe der `associate-web-acl` AWS WAF öffentlichen API dem App Runner-Dienst zu. Weitere Informationen AWS WAF APIs dazu finden Sie unter AssociateWebACL im AWS WAF API-Referenzhandbuch.

> **Note**
>
> Wenn Sie eine Web-ACL erstellen, vergeht eine kurze Zeit, bis die Web-ACL vollständig propagiert wird und App Runner zur Verfügung steht. Die Übertragungszeit kann zwischen einigen Sekunden und mehreren Minuten liegen. AWS WAF gibt a zurück`WAFUnavailableEntityException`, wenn Sie versuchen, eine Web-ACL zuzuordnen, bevor sie vollständig weitergegeben wurde.
> Wenn Sie den Browser aktualisieren oder die App Runner-Konsole verlassen, bevor die Web-ACL vollständig weitergegeben wurde, schlägt die Zuordnung fehl. Sie können jedoch innerhalb der App Runner-Konsole navigieren.

Example Zuordnen einer Web-ACL — Anfrage

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. Stellen Sie mithilfe der `get-web-acl-for-resource` AWS WAF öffentlichen API sicher, dass die Web-ACL mit Ihrem App Runner-Dienst verknüpft ist. Weitere Informationen AWS WAF APIs dazu finden Sie unter GetWebACLForRessource im AWS WAF API-Referenzhandbuch.

Example Überprüfen Sie die Web-ACL für die Ressource — Anfrage

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

Wenn Ihrem Service kein Web ACLs zugeordnet ist, erhalten Sie eine leere Antwort.

Löschen einer AWS WAF Web-ACL mit AWS CLI

Sie können eine AWS WAF Web-ACL nicht löschen, wenn sie mit einem App Runner-Dienst verknüpft ist.

Um eine AWS WAF Web-ACL zu löschen

1. Trennen Sie die Web-ACL mithilfe der `disassociate-web-acl` AWS WAF öffentlichen API von Ihrem App Runner-Dienst. Weitere Informationen dazu finden Sie AWS WAF APIs unter DisassociateWebACL im AWS WAF API-Referenzhandbuch.

   Example Trennen einer Web-ACL — Anfrage

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. Stellen Sie mithilfe der öffentlichen API sicher, dass die Web-ACL von Ihrem App Runner-Dienst getrennt wurde. `get-web-acl-for-resource` AWS WAF

   Example Stellen Sie sicher, dass die Web-ACL getrennt ist — Anfrage

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   Die getrennte Web-ACL ist für Ihren App Runner-Dienst nicht aufgeführt. Wenn Ihrem Dienst kein Web ACLs zugeordnet ist, erhalten Sie eine leere Antwort.

3. Löschen Sie die getrennte Web-ACL mithilfe der `delete-web-acl` AWS WAF öffentlichen API. Weitere Informationen dazu finden Sie AWS WAF APIs unter DeleteWebACL im AWS WAF API-Referenzhandbuch.

   Example Eine Web-ACL löschen — Anfrage

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   --id <web_acl_id>
   --lock-token <web_acl_lock_token>
   --region <region>
   ```

4. Stellen Sie sicher, dass die Web-ACL mithilfe der `list-web-acl` AWS WAF öffentlichen API gelöscht wurde. Weitere Informationen AWS WAF APIs dazu finden Sie ListWebACLsim AWS WAF API-Referenzhandbuch.

   Example Stellen Sie sicher, dass die Web-ACL gelöscht wurde — Anfrage

   ```
   aws wafv2 list-web-acls
   --scope REGIONAL
   --region <region>
   ```

   Die gelöschte Web-ACL ist nicht mehr aufgeführt.

   > **Note**
   >
   > Wenn eine Web-ACL mit anderen aktiven App Runner-Diensten oder anderen Ressourcen wie Amazon Cognito Cognito-Benutzerpools verknüpft ist, kann die Web-ACL nicht gelöscht werden.

Listet App Runner-Dienste auf, die mit einer Web-ACL verknüpft sind

Eine Web-ACL kann mehreren App Runner-Diensten und anderen Ressourcen zugeordnet werden. Listet die App Runner-Dienste auf, die einer Web-ACL mithilfe der `list-resources-for-web-acl` AWS WAF öffentlichen API zugeordnet sind. Weitere Informationen AWS WAF APIs dazu finden Sie unter ListResourcesForWebACL im AWS WAF API-Referenzhandbuch.

Example Listet die App Runner-Dienste auf, die mit einer Web-ACL verknüpft sind — Anfrage

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example Listet die App Runner-Dienste auf, die einer Web-ACL zugeordnet sind — Antwort

Das folgende Beispiel veranschaulicht die Reaktion, wenn keine App Runner-Dienste vorhanden sind, die einer Web-ACL zugeordnet sind.

```
{
  "ResourceArns": []
}
```

Example Listet die App Runner-Dienste auf, die einer Web-ACL zugeordnet sind — Antwort

Das folgende Beispiel veranschaulicht die Reaktion, wenn App Runner-Dienste mit einer Web-ACL verknüpft sind.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

AWS WAF Web testen und protokollieren ACLs

Wenn Sie eine Regelaktion in Ihrer Web-ACL auf Count setzen, wird die Anfrage einer Anzahl von Anfragen AWS WAF hinzugefügt, die der Regel entsprechen. Um eine Web-ACL mit Ihrem App Runner-Dienst zu testen, setzen Sie die Regelaktionen auf Count und berücksichtigen Sie die Anzahl der Anfragen, die jeder Regel entsprechen. Sie legen beispielsweise eine Regel für die `Block` Aktion fest, die einer großen Anzahl von Anfragen entspricht, bei denen es sich Ihrer Meinung nach um normalen Benutzerverkehr handelt. In diesem Fall müssen Sie Ihre Regel möglicherweise neu konfigurieren. Weitere Informationen finden Sie im AWS WAF Entwicklerhandbuch unter Testen und AWS WAF Optimieren Ihrer Schutzmaßnahmen.

Sie können auch so konfigurieren AWS WAF , dass Anforderungsheader in einer Amazon CloudWatch Logs-Protokollgruppe, einem Amazon Simple Storage Service (Amazon S3) -Bucket oder einer Amazon Data Firehose protokolliert werden. Weitere Informationen finden Sie unter Protokollieren des Web-ACL-Datenverkehrs im Entwicklerhandbuch zu AWS WAF .

Informationen zum Zugriff auf Protokolle im Zusammenhang mit der Web-ACL, die mit Ihrem App Runner-Service verknüpft ist, finden Sie in den folgenden Protokollfeldern:

- `httpSourceName`: Enthält `APPRUNNER`
- `httpSourceId`: Enthält `customeraccountid-apprunnerserviceid`

Weitere Informationen finden Sie unter Protokollbeispiele im AWS WAF Entwicklerhandbuch.

> **Important**
>
> Quell-IP-Regeln für private App Runner-Dienste, die mit dem WAF-Web verknüpft sind, entsprechen ACLs nicht den IP-basierten Regeln. Dies liegt daran, dass wir derzeit die Weiterleitung von Quell-IP-Daten von Anfragen an private App Runner-Dienste, die mit WAF verknüpft sind, nicht unterstützen. Wenn für Ihre App Runner-Anwendung Quell-IP/CIDR-Regeln zur Steuerung des eingehenden Datenverkehrs erforderlich sind, müssen Sie Sicherheitsgruppenregeln für private Endpunkte anstelle von WAF Web verwenden. ACLs

App Runner-Dienstoptionen mithilfe einer Konfigurationsdatei einrichten

> **Note**
>
> Konfigurationsdateien gelten nur für Dienste, die auf Quellcode basieren. Sie können Konfigurationsdateien nicht mit imagebasierten Diensten verwenden.

Wenn Sie einen AWS App Runner Dienst mithilfe eines Quellcode-Repositorys erstellen, AWS App Runner sind Informationen zum Erstellen und Starten Ihres Dienstes erforderlich. Sie können diese Informationen jedes Mal angeben, wenn Sie einen Dienst mithilfe der App Runner-Konsole oder API erstellen. Alternativ können Sie die Serviceoptionen mithilfe einer Konfigurationsdatei festlegen. Die Optionen, die Sie in einer Datei angeben, werden Teil Ihres Quell-Repositorys, und alle Änderungen an diesen Optionen werden ähnlich wie Änderungen am Quellcode nachverfolgt. Sie können die App Runner-Konfigurationsdatei verwenden, um mehr Optionen anzugeben, als die API unterstützt. Sie müssen keine Konfigurationsdatei bereitstellen, wenn Sie nur die grundlegenden Optionen benötigen, die die API unterstützt.

Die App Runner-Konfigurationsdatei ist eine YAML-Datei, die `apprunner.yaml` im Quellverzeichnis des Repositorys Ihrer Anwendung benannt ist. Sie bietet Build- und Runtime-Optionen für Ihren Service. Die Werte in dieser Datei weisen App Runner an, wie Sie Ihren Dienst erstellen und starten, und stellen Laufzeitkontext wie Netzwerkeinstellungen und Umgebungsvariablen bereit.

Die App Runner-Konfigurationsdatei enthält keine Betriebseinstellungen wie CPU und Speicher.

Beispiele für App Runner-Konfigurationsdateien finden Sie unterthe section called “Beispiele”. Ein vollständiges Referenzhandbuch finden Sie unterthe section called “Referenz”.

Themen

Beispiele für App Runner-Konfigurationsdateien

> **Note**
>
> Konfigurationsdateien gelten nur für Dienste, die auf Quellcode basieren. Sie können Konfigurationsdateien nicht mit imagebasierten Diensten verwenden.

Die folgenden Beispiele veranschaulichen AWS App Runner Konfigurationsdateien. Einige sind minimal und enthalten nur die erforderlichen Einstellungen. Andere sind vollständig, einschließlich aller Abschnitte der Konfigurationsdatei. Eine Übersicht über die App Runner-Konfigurationsdateien finden Sie unterApp Runner-Konfigurationsdatei.

Beispiele für Konfigurationsdateien

Minimale Konfigurationsdatei

Bei einer minimalen Konfigurationsdatei geht App Runner von den folgenden Annahmen aus:

- Während der Erstellung oder Ausführung sind keine benutzerdefinierten Umgebungsvariablen erforderlich.
- Die neueste Runtime-Version wird verwendet.
- Die Standard-Portnummer und die Port-Umgebungsvariable werden verwendet.

Example apprunner.yaml

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

Vollständige Konfigurationsdatei

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel im `apprunner.yaml` Originalformat mit einer verwalteten Laufzeit.

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"
```

Vollständige Konfigurationsdatei — (verwendet einen überarbeiteten Build)

Dieses Beispiel zeigt die Verwendung aller Konfigurationsschlüssel in der `apprunner.yaml` mit einer verwalteten Laufzeit.

Der `pre-run` Parameter wird nur vom überarbeiteten App Runner-Build unterstützt. Fügen Sie diesen Parameter nicht in Ihre Konfigurationsdatei ein, wenn Ihre Anwendung Runtime-Versionen verwendet, die vom ursprünglichen App Runner-Build unterstützt werden. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

> **Note**
>
> Da dieses Beispiel für Python 3.11 ist, verwenden wir die `python3` Befehle `pip3` und. Weitere Informationen finden Sie Callouts für bestimmte Runtime-Versionen im Thema Python-Plattform.

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"
```

Beispiele für spezifische verwaltete Laufzeitkonfigurationsdateien finden Sie im jeweiligen Unterthema zur Laufzeit unter. Codebasierter Dienst

Referenz zur App Runner-Konfigurationsdatei

> **Note**
>
> Konfigurationsdateien gelten nur für Dienste, die auf Quellcode basieren. Sie können Konfigurationsdateien nicht mit imagebasierten Diensten verwenden.

Dieses Thema ist ein umfassendes Referenzhandbuch zur Syntax und Semantik einer AWS App Runner Konfigurationsdatei. Eine Übersicht über die App Runner-Konfigurationsdateien finden Sie unterApp Runner-Konfigurationsdatei.

Die App Runner-Konfigurationsdatei ist eine YAML-Datei. Benennen Sie sie `apprunner.yaml` und platzieren Sie sie im Quellverzeichnis des Repositorys Ihrer Anwendung.

Überblick über die Struktur

Die App Runner-Konfigurationsdatei ist eine YAML-Datei. Benennen Sie sie `apprunner.yaml` und platzieren Sie sie im Quellverzeichnis des Repositorys Ihrer Anwendung.

Die App Runner-Konfigurationsdatei enthält die folgenden Hauptteile:

- Oberer Bereich — Enthält Schlüssel der obersten Ebene
- Abschnitt „Build“ — Konfiguriert die Build-Phase
- Abschnitt „Ausführen“ — Konfiguriert die Laufzeitphase

Oberer Abschnitt

Die Schlüssel oben in der Datei enthalten allgemeine Informationen über die Datei und Ihre Dienstlaufzeit. Die folgenden Schlüssel sind verfügbar:

- `version`— Erforderlich. Die Version der App Runner-Konfigurationsdatei. Verwenden Sie idealerweise die neueste Version.

  Syntax

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`— Erforderlich. Der Name der Runtime, die Ihre Anwendung verwendet. Informationen zu den verfügbaren Laufzeiten für die verschiedenen Programmierplattformen, die App Runner anbietet, finden Sie unterCodebasierter Dienst.

  > **Note**
  >
  > Die Benennungskonvention einer verwalteten Laufzeit lautet*<language-name><major-version>*.

  Syntax

  ```
  runtime: runtime-name
  ```

  Example

  ```
  runtime: python3
  ```

Abschnitt erstellen

Im Build-Abschnitt wird die Buildphase der App Runner-Dienstbereitstellung konfiguriert. Sie können Build-Befehle und Umgebungsvariablen angeben. Build-Befehle sind erforderlich.

Der Abschnitt beginnt mit dem `build:` Schlüssel und hat die folgenden Unterschlüssel:

- `commands`— Erforderlich. Gibt die Befehle an, die App Runner in verschiedenen Buildphasen ausführt. Beinhaltet die folgenden Unterschlüssel:
  - `pre-build`— Fakultativ. Die Befehle, die App Runner vor dem Build ausführt. Installieren Sie beispielsweise npm Abhängigkeiten oder Testbibliotheken.
  - `build`— Erforderlich. Die Befehle, die App Runner ausführt, um Ihre Anwendung zu erstellen. Verwenden Sie beispielsweisepipenv.
  - `post-build`— Fakultativ. Die Befehle, die App Runner nach dem Build ausführt. Verwenden Sie Maven beispielsweise, um Build-Artefakte in eine JAR- oder WAR-Datei zu packen oder einen Test auszuführen.

  Syntax

```
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`— Fakultativ. Gibt benutzerdefinierte Umgebungsvariablen für die Erstellungsphase an. Definiert als skalare Name-Wert-Zuordnungen. Sie können in Ihren Build-Befehlen namentlich auf diese Variablen verweisen.

> **Note**
>
> In dieser Konfigurationsdatei gibt es zwei unterschiedliche `env` Einträge an zwei verschiedenen Stellen. Ein Satz befindet sich im Abschnitt Build und der andere im Abschnitt Run.
>
> - Auf den `env` Satz im Abschnitt Build kann während des Build-Vorgangs mit den `pre-run` Befehlen `pre-build` `buildpost-build`,, und verwiesen werden.
>
> Wichtig — Beachten Sie, dass sich die `pre-run` Befehle im Abschnitt Ausführen dieser Datei befinden, obwohl sie nur auf die Umgebungsvariablen zugreifen können, die im Abschnitt Build definiert sind.
> - Auf den `env` Satz im Abschnitt Run kann durch den `run` Befehl in der Laufzeitumgebung verwiesen werden.

Syntax

```
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"
```

Abschnitt „Ausführen“

Im Abschnitt run wird die Container-Ausführungsphase der App Runner-Anwendungsbereitstellung konfiguriert. Sie können die Laufzeitversion, Befehle vor der Ausführung (nur überarbeitetes Format), den Startbefehl, den Netzwerkport und Umgebungsvariablen angeben.

Der Abschnitt beginnt mit dem `run:` Schlüssel und hat die folgenden Unterschlüssel:

- `runtime-version`— Fakultativ. Gibt eine Runtime-Version an, die Sie für Ihren App Runner-Dienst sperren möchten.

  Standardmäßig ist nur die Hauptversion gesperrt. App Runner verwendet bei jeder Bereitstellung oder jedem Service-Update die neuesten Neben- und Patch-Versionen, die für die Laufzeit verfügbar sind. Wenn Sie Haupt- und Nebenversionen angeben, werden beide gesperrt, und App Runner aktualisiert nur Patch-Versionen. Wenn Sie Haupt-, Neben- und Patch-Versionen angeben, ist Ihr Dienst auf eine bestimmte Runtime-Version beschränkt und App Runner aktualisiert sie nie.

  Syntax

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  >
  > Die Laufzeiten einiger Plattformen haben unterschiedliche Versionskomponenten. Einzelheiten finden Sie in den spezifischen Plattformthemen.

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run`— Fakultativ. Nur überarbeitete Build-Nutzung. Gibt die Befehle an, die App Runner ausführt, nachdem Ihre Anwendung vom Build-Image in das Run-Image kopiert wurde. Sie können hier Befehle eingeben, um das Run-Image außerhalb des `/app` Verzeichnisses zu ändern. Wenn Sie beispielsweise zusätzliche globale Abhängigkeiten installieren müssen, die sich außerhalb des `/app` Verzeichnisses befinden, geben Sie dazu die erforderlichen Befehle in diesem Unterabschnitt

ein. Weitere Informationen zum App Runner-Erstellungsprozess finden Sie unter. Verwaltete Runtime-Versionen und der App Runner-Build

> **Note**
>
> - Wichtig — Obwohl die `pre-run` Befehle im Abschnitt Ausführen aufgeführt sind, können sie nur auf die Umgebungsvariablen verweisen, die im Abschnitt Build dieser Konfigurationsdatei definiert sind. Sie können nicht auf die in diesem Run-Abschnitt definierten Umgebungsvariablen verweisen.
> - Der `pre-run` Parameter wird nur vom überarbeiteten App Runner-Build unterstützt. Fügen Sie diesen Parameter nicht in Ihre Konfigurationsdatei ein, wenn Ihre Anwendung Runtime-Versionen verwendet, die vom ursprünglichen App Runner-Build unterstützt werden. Weitere Informationen finden Sie unter Verwaltete Runtime-Versionen und der App Runner-Build.

Syntax

```
run:
  pre-run:
      - command
      - …
```

- `command`— Erforderlich. Der Befehl, mit dem App Runner Ihre Anwendung nach Abschluss der Anwendungsentwicklung ausführt.

  Syntax

  ```
  run:
    command: command
  ```

- `network`— Fakultativ. Gibt den Port an, auf den Ihre Anwendung hört. Diese umfasst das Folgende:
  - `port`— Fakultativ. Falls angegeben, ist dies die Portnummer, die Ihre Anwendung abhört. Der Standardwert ist `8080`.
  - `env`— Fakultativ. Falls angegeben, übergibt App Runner die Portnummer in dieser Umgebungsvariablen an den Container und übergibt (nicht stattdessen) dieselbe Portnummer in

der Standardumgebungsvariablen,PORT. Mit anderen Worten, wenn Sie angebenenv, übergibt App Runner die Portnummer in zwei Umgebungsvariablen.

Syntax

```
run:
  network:
    port: port-number
    env: env-variable-name
```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- env— Fakultativ. Definition von benutzerdefinierten Umgebungsvariablen für die Ausführungsphase. Definiert als skalare Name-Wert-Zuordnungen. Sie können in Ihrer Laufzeitumgebung namentlich auf diese Variablen verweisen.

> **Note**
>
> In dieser Konfigurationsdatei gibt es zwei unterschiedliche env Einträge an zwei verschiedenen Stellen. Ein Satz befindet sich im Abschnitt Build und der andere im Abschnitt Run.
>
> - Auf den env Satz im Abschnitt Build kann während des Build-Vorgangs mit den pre-run Befehlen pre-build buildpost-build,, und verwiesen werden.
>
> Wichtig — Beachten Sie, dass sich die pre-run Befehle im Abschnitt Ausführen dieser Datei befinden, obwohl sie nur auf die Umgebungsvariablen zugreifen können, die im Abschnitt Build definiert sind.
>
> - Auf den env Satz im Abschnitt Run kann durch den run Befehl in der Laufzeitumgebung verwiesen werden.

Syntax

```
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"
```

Die App Runner API

Die AWS App Runner Anwendungsprogrammierschnittstelle (API) ist eine RESTful API für Anfragen an den App Runner-Dienst. Sie können die API verwenden, um App Runner-Ressourcen in Ihrem zu erstellen, aufzulisten, zu beschreiben, zu aktualisieren und zu löschen AWS-Konto.

Sie können die API direkt in Ihrem Anwendungscode aufrufen oder eine der AWS SDKs

Vollständige API-Referenzinformationen finden Sie in der AWS App Runner API-Referenz.

Weitere Informationen zu AWS Entwicklertools finden Sie unter Tools, auf denen Sie aufbauen können AWS.

Themen

- Verwenden Sie den AWS CLI , um mit App Runner zu arbeiten
- Wird verwendet AWS CloudShell , um damit zu arbeiten AWS App Runner

Verwenden Sie den AWS CLI , um mit App Runner zu arbeiten

Verwenden Sie bei Befehlszeilenskripten den, AWS CLIum den App Runner-Dienst aufzurufen. Vollständige AWS CLI Referenzinformationen finden Sie unter Apprunner in der AWS CLI Befehlsreferenz.

AWS CloudShell ermöglicht es Ihnen, die Installation AWS CLI in Ihrer Entwicklungsumgebung zu überspringen und sie stattdessen in der AWS Management Console zu verwenden. Sie vermeiden nicht nur die Installation, sondern müssen auch keine Anmeldeinformationen konfigurieren und keine Region angeben. Ihre AWS Management Console Sitzung bietet diesen Kontext für die AWS CLI. Weitere Informationen CloudShell und ein Anwendungsbeispiel finden Sie unterthe section called “Benutzen AWS CloudShell”.

Wird verwendet AWS CloudShell , um damit zu arbeiten AWS App Runner

AWS CloudShell ist eine browserbasierte, vorauthentifizierte Shell, die Sie direkt von der aus starten können. AWS Management Console Sie können AWS CLI Befehle für AWS Dienste (einschließlich

AWS App Runner) mithilfe Ihrer bevorzugten Shell (Bash PowerShell oder Z-Shell) ausführen. Und Sie können dies tun, ohne Befehlszeilentools herunterladen oder installieren zu müssen.

Sie starten AWS CloudShell von der aus AWS Management Console, und die AWS Anmeldeinformationen, mit denen Sie sich an der Konsole angemeldet haben, sind automatisch in einer neuen Shell-Sitzung verfügbar. Diese Vorauthentifizierung von AWS CloudShell Benutzern ermöglicht es Ihnen, die Konfiguration von Anmeldeinformationen zu überspringen, wenn Sie mit AWS Diensten wie App Runner interagieren, die AWS CLI Version 2 verwenden (vorinstalliert in der Rechenumgebung der Shell).

Themen

- Erhalt von IAM-Berechtigungen für AWS CloudShell
- Interaktion mit App Runner mithilfe AWS CloudShell
- Überprüfen Sie Ihren App Runner-Dienst mit AWS CloudShell

Erhalt von IAM-Berechtigungen für AWS CloudShell

Mithilfe der von bereitgestellten Ressourcen zur Zugriffsverwaltung können Administratoren IAM-Benutzern Berechtigungen erteilen AWS Identity and Access Management, sodass sie auf die Funktionen der Umgebung zugreifen AWS CloudShell und diese nutzen können.

Am schnellsten kann ein Administrator Benutzern Zugriff gewähren, indem er eine AWS verwaltete Richtlinie verwendet. Bei einer von AWS verwalteten Richtlinie handelt es sich um eine eigenständige Richtlinie, die von AWS erstellt und verwaltet wird. Die folgende AWS verwaltete Richtlinie für CloudShell kann an IAM-Identitäten angehängt werden:

- `AWSCloudShellFullAccess`: Erteilt die Erlaubnis zur Nutzung AWS CloudShell mit vollem Zugriff auf alle Funktionen.

Wenn Sie den Umfang der Aktionen einschränken möchten, die ein IAM-Benutzer ausführen kann AWS CloudShell, können Sie eine benutzerdefinierte Richtlinie erstellen, die die `AWSCloudShellFullAccess` verwaltete Richtlinie als Vorlage verwendet. Weitere Informationen zur Einschränkung der Aktionen, die Benutzern zur Verfügung stehen CloudShell, finden Sie im AWS CloudShell Benutzerhandbuch unter Verwaltung von AWS CloudShell Zugriff und Nutzung mit IAM-Richtlinien.

> **Note**
>
> Für Ihre IAM-Identität ist außerdem eine Richtlinie erforderlich, die die Erlaubnis erteilt, App Runner aufzurufen. Weitere Informationen finden Sie unter the section called “App Runner und IAM”.

Interaktion mit App Runner mithilfe AWS CloudShell

Nach dem Start AWS CloudShell von der AWS Management Console können Sie sofort mit der Interaktion mit App Runner über die Befehlszeilenschnittstelle beginnen.

Im folgenden Beispiel rufen Sie mithilfe von in Informationen zu einem Ihrer App Runner-Dienste ab CloudShell. AWS CLI

> **Note**
>
> Wenn Sie AWS CLI in verwenden AWS CloudShell, müssen Sie keine zusätzlichen Ressourcen herunterladen oder installieren. Da Sie außerdem bereits in der Shell authentifiziert sind, müssen Sie vor dem Tätigen von Anrufen keine Anmeldeinformationen konfigurieren.

Example App Runner-Dienstinformationen abrufen mit AWS CloudShell

1. Von der aus können Sie starten AWS Management Console, CloudShell indem Sie die folgenden Optionen auswählen, die in der Navigationsleiste verfügbar sind:
   - Wählen Sie das CloudShell Symbol.
   - Beginnen Sie mit der Eingabe **`cloudshell`** in das Suchfeld und wählen Sie dann die CloudShellOption aus, wenn Sie sie in den Suchergebnissen sehen.
2. Um alle aktuellen App Runner-Dienste in Ihrem AWS Konto in der AWS Region der Konsolensitzung aufzulisten, geben Sie den folgenden Befehl in die CloudShell Befehlszeile ein:

   ```
   $ aws apprunner list-services
   ```

   In der Ausgabe werden zusammenfassende Informationen zu Ihren Diensten aufgeführt.

```
{
  "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. Um eine detaillierte Beschreibung eines bestimmten App Runner-Dienstes zu erhalten, geben Sie den folgenden Befehl in die CloudShell Befehlszeile ein und verwenden dabei einen der im vorherigen Schritt ARNs abgerufenen Befehle:

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

In der Ausgabe wird eine detaillierte Beschreibung des von Ihnen angegebenen Dienstes aufgeführt.

```
{
  "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
    }
  }
}
```

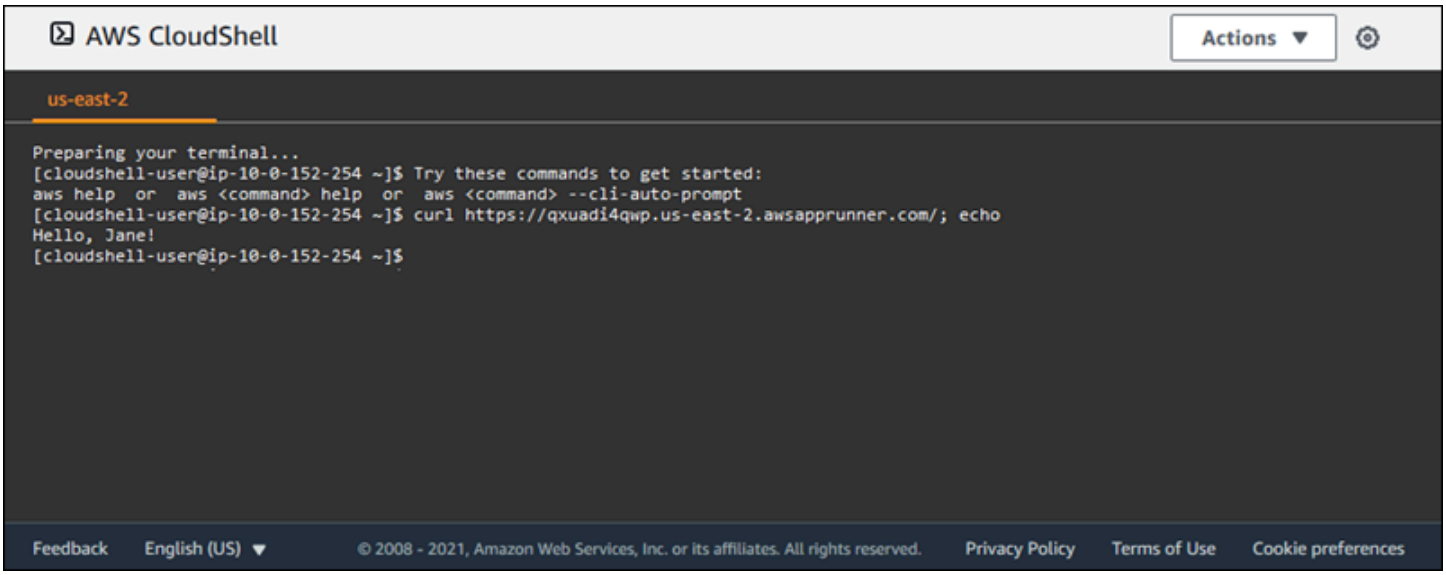
Überprüfen Sie Ihren App Runner-Dienst mit AWS CloudShell

Wenn Sie einen App Runner-Dienst erstellen, erstellt App Runner eine Standarddomain für die Website Ihres Dienstes und zeigt sie in der Konsole an (oder gibt sie im API-Aufrufergebnis zurück). Sie können damit Aufrufe CloudShell an Ihre Website tätigen und überprüfen, ob sie ordnungsgemäß funktioniert.

Nachdem Sie beispielsweise einen App Runner-Dienst wie unter beschrieben erstellt habenErste Schritte, führen Sie den folgenden Befehl in aus CloudShell:

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

Die Ausgabe sollte den erwarteten Seiteninhalt zeigen.



Fehlerbehebung

Dieses Kapitel enthält Schritte zur Behebung häufiger Fehler und Probleme, die bei der Nutzung Ihres AWS App Runner Dienstes auftreten können. Die Fehlermeldungen können in der Konsole, in der API oder auf der Registerkarte „Protokolle“ Ihrer Serviceseite angezeigt werden.

Weitere Tipps zur Fehlerbehebung und Antworten auf häufig gestellte Supportfragen finden Sie im - Wissenscenter.

Themen

Wenn der Dienst nicht erstellt werden kann

Wenn Ihr Versuch, einen App Runner-Dienst zu erstellen, fehlschlägt, wechselt der Dienst in einen `CREATE_FAILED` Status. Dieser Status wird auf der Konsole als Create failed angezeigt. Ein Dienst kann möglicherweise aufgrund von Problemen nicht erstellt werden, die mit einem oder mehreren der folgenden Probleme zusammenhängen:

- Ihr Anwendungscode
- Der Build-Prozess
- Konfiguration
- Ressourcenkontingente
- Temporäre Probleme mit dem Basiswert AWS-Services , den Ihr Service verwendet

Um einen Dienst zu beheben, der nicht erstellt werden kann, empfehlen wir Ihnen, wie folgt vorzugehen.

1. Lesen Sie die Ereignisse und Protokolle des Dienstes, um herauszufinden, warum der Dienst nicht erstellt werden konnte.

2. Nehmen Sie alle erforderlichen Änderungen an Ihrem Code oder Ihrer Konfiguration vor.
3. Wenn Sie Ihr Dienstkontingent erreicht haben, löschen Sie einen oder mehrere Dienste.
4. Wenn Sie ein anderes Ressourcenkontingent erreicht haben, können Sie es möglicherweise erhöhen, sofern es anpassbar ist.
5. Versuchen Sie erneut, den Dienst neu aufzubauen, nachdem Sie alle oben genannten Schritte ausgeführt haben. Informationen dazu, wie Sie Ihren Service neu aufbauen können, finden Sie unterthe section called “Fehlgeschlagenen Dienst neu aufbauen”.

> **Note**
>
> Eine der anpassbaren Ressourcenkontingente, die ein Problem verursachen könnten, ist die Fargate On-Demand-vCPU-Ressource.
> Die Anzahl der vCPU-Ressourcen bestimmt die Anzahl der Instanzen, die App Runner für Ihren Dienst bereitstellen kann. Dies ist ein einstellbarer Kontingentwert für die Fargate On-Demand-vCPU-Ressourcenanzahl, die sich im Service befindet. AWS Fargate Um die vCPU-Kontingenteinstellungen für Ihr Konto einzusehen oder eine Erhöhung des Kontingents zu beantragen, verwenden Sie die Konsole Service Quotas in der AWS Management Console. Weitere Informationen finden Sie unter AWS Fargate Service-Kontingente im Amazon Elastic Container Service Developer Guide.

> **Important**
>
> Für einen fehlgeschlagenen Service fallen für Sie keine zusätzlichen Kosten an, die über den ersten Erstellungsversuch hinausgehen. Auch wenn der ausgefallene Dienst nicht nutzbar ist, wird er dennoch auf Ihr Servicekontingent angerechnet. App Runner löscht den ausgefallenen Dienst nicht automatisch. Stellen Sie also sicher, dass Sie ihn löschen, wenn Sie mit der Analyse des Fehlers fertig sind.

Benutzerdefinierte Domainnamen

In diesem Abschnitt wird beschrieben, wie Sie verschiedene Fehler beheben und beheben können, die beim Verknüpfen mit einer benutzerdefinierten Domain auftreten können.

> **Note**
>
> Um die Sicherheit Ihrer App Runner-Anwendungen zu erhöhen, ist die Domain*.awsapprunner.com in der Public Suffix List (PSL) registriert. Aus Sicherheitsgründen empfehlen wir Ihnen, Cookies mit einem `__Host-` Präfix zu verwenden, falls Sie jemals sensible Cookies im Standard-Domainnamen für Ihre App Runner-Anwendungen einrichten müssen. Diese Vorgehensweise hilft Ihnen dabei, Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery Attempts, Anforderungsfälschung zwischen Websites) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite im Mozilla Developer Network.

Der Create Fail-Fehler für die benutzerdefinierte Domain wird angezeigt

- Überprüfen Sie, ob dieser Fehler auf ein Problem mit den CAA-Einträgen zurückzuführen ist. Wenn es in der DNS-Struktur keine CAA-Einträge gibt, erhalten Sie eine Nachricht und stellen ein Zertifikat aus`fail open`, AWS Certificate Manager um die benutzerdefinierte Domain zu verifizieren. Dadurch kann App Runner die benutzerdefinierte Domain akzeptieren. Wenn Sie CAA-Zertifizierungen in den DNS-Einträgen verwenden, stellen Sie sicher, dass die CAA-Einträge mindestens einer Domain Folgendes enthalten: `amazon.com` Andernfalls kann ACM kein Zertifikat ausstellen. Infolgedessen kann die benutzerdefinierte Domäne für App Runner nicht erstellt werden.

  Im folgenden Beispiel wird das DNS-Suchtool DiG verwendet, um CAA-Einträge anzuzeigen, bei denen ein erforderlicher Eintrag fehlt. Im Beispiel wird `example.com` als benutzerdefinierte Domäne verwendet. Führen Sie im Beispiel die folgenden Befehle aus, um die CAA-Einträge zu überprüfen.

  ```
  ...
  ;; 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...
  ```

- Korrigieren Sie die Domäneneinträge und stellen Sie sicher, dass mindestens ein CAA-Eintrag Folgendes enthält. `amazon.com`

- Versuchen Sie erneut, die benutzerdefinierte Domain mit App Runner zu verknüpfen.

Anweisungen zur Behebung von CAA-Fehlern finden Sie im Folgenden:

- Probleme mit der Autorisierung durch die Zertifizierungsstelle (CAA)
- Wie behebe ich CAA-Fehler bei der Ausstellung oder Verlängerung eines ACM-Zertifikats?

Fehler beim Ausstehen der DNS-Zertifikatsvalidierung für eine benutzerdefinierte Domain

- Prüfen Sie, ob Sie einen wichtigen Schritt bei der Einrichtung der benutzerdefinierten Domain übersprungen haben. Prüfen Sie außerdem, ob Sie mit einem DNS-Suchtool wie DiG einen DNS-Eintrag falsch konfiguriert haben. Achten Sie insbesondere auf die folgenden Fehler:
  - Alle verpassten Schritte.
  - Nicht unterstützte Zeichen wie doppelte Anführungszeichen in den DNS-Einträgen.
- Korrigieren Sie die Fehler.
- Versuchen Sie erneut, die benutzerdefinierte Domain mit App Runner zu verknüpfen.

Anweisungen zur Behebung von CAA-Validierungsfehlern finden Sie im Folgenden.

- DNS-Validierung
- the section called “Benutzerdefinierte Domainnamen”

Grundlegende Befehle zur Fehlerbehebung

- Vergewissern Sie sich, dass ein Dienst gefunden werden kann.

```
aws apprunner list-services
```

- Beschreiben Sie einen Dienst und überprüfen Sie seinen Status.

```
aws apprunner describe-service --service-arn
```

```
```

- Überprüfen Sie den Status der benutzerdefinierten Domain.

```
aws apprunner describe-custom-domains --service-arn
```

- Listet alle laufenden Operationen auf.

```
aws apprunner list-operations --service-arn
```

Verlängerung des benutzerdefinierten Domainzertifikats

Wenn Sie Ihrem Dienst eine benutzerdefinierte Domain hinzufügen, stellt Ihnen App Runner eine Reihe von CNAME-Einträgen zur Verfügung, die Sie Ihrem DNS-Server hinzufügen. Diese CNAME-Einträge enthalten Zertifikatsdatensätze. App Runner verwendet AWS Certificate Manager (ACM), um die Domain zu verifizieren. App Runner validiert diese DNS-Einträge, um sicherzustellen, dass Sie weiterhin Eigentümer dieser Domain sind. Wenn Sie die CNAME-Einträge aus Ihrer DNS-Zone entfernen, kann App Runner die DNS-Einträge nicht mehr validieren und das benutzerdefinierte Domain-Zertifikat kann nicht automatisch verlängert werden.

In diesem Abschnitt wird beschrieben, wie Sie die folgenden Probleme bei der Verlängerung von benutzerdefinierten Domainzertifikaten lösen können:

- the section called “Der CNAME wird vom DNS-Server entfernt”.
- the section called “Das Zertifikat ist abgelaufen”.

Der CNAME wird vom DNS-Server entfernt

- Rufen Sie Ihre CNAME-Einträge über die DescribeCustomDomainsAPI oder über die benutzerdefinierten Domain-Einstellungen in der App Runner-Konsole ab. Informationen zu gespeicherten Informationen finden Sie CNAMEs unter CertificateValidationRecords.
- Fügen Sie Ihrem DNS-Server die CNAME-Einträge zur Zertifikatsvalidierung hinzu. App Runner kann dann überprüfen, ob Sie Eigentümer der Domain sind. Nachdem Sie die CNAME-Einträge hinzugefügt haben, kann es bis zu 30 Minuten dauern, bis die DNS-Einträge weitergegeben

werden. Es kann auch mehrere Stunden dauern, bis App Runner und ACM den Prozess der Zertifikatserneuerung erneut versuchen. Anweisungen zum Hinzufügen von CNAME-Einträgen finden Sie unter. the section called “Verwalte benutzerdefinierte Domains”

Das Zertifikat ist abgelaufen

- Trennen Sie die Zuordnung (Verknüpfung aufheben) und ordnen Sie dann die benutzerdefinierte Domain für Ihren App Runner-Dienst mithilfe der App Runner-Konsole oder API zu (verknüpfen). App Runner erstellt neue CNAME-Einträge zur Zertifikatsvalidierung.
- Fügen Sie Ihrem DNS-Server die neuen CNAME-Einträge zur Zertifikatsvalidierung hinzu.

  Anweisungen zum Trennen (Aufheben der Verknüpfung) und zum Zuordnen (Verknüpfen) der benutzerdefinierten Domäne finden Sie unter. the section called “Verwalte benutzerdefinierte Domains”

Wie überprüfe ich, ob das Zertifikat erfolgreich erneuert wurde

Sie können den Status Ihrer Zertifikatsdatensätze überprüfen, um sicherzustellen, dass Ihr Zertifikat erfolgreich erneuert wurde. Sie können den Status der Zertifikate mithilfe von Tools wie curl überprüfen.

Weitere Informationen zur Verlängerung von Zertifikaten finden Sie unter den folgenden Links:

- Warum ist mein ACM-Zertifikat als nicht verlängerbar gekennzeichnet?
- Verwaltete Verlängerung von ACM-Zertifikaten
- DNS-Validierung

Fehler beim Routing von HTTP/HTTPS-Anfragen

In diesem Abschnitt wird beschrieben, wie Sie Fehler beheben und beheben können, die beim Routing von HTTP/HTTPS-Verkehr zu Ihren App Runner-Dienstendpunkten auftreten können.

Fehler 404 Nicht gefunden beim Senden von HTTP/HTTPS-Verkehr an App Runner-Dienstendpunkte

- Stellen Sie sicher, `Host Header` dass der auf die Service-URL in der HTTP-Anfrage verweist, da App Runner die Host-Header-Informationen verwendet, um Anfragen weiterzuleiten. Die meisten

ClientscURL, wie und Webbrowser, verweisen den Host-Header automatisch auf die Service-URL. Wenn Ihr Client die Service-URL nicht als die festlegt`Host Header`, erhalten Sie eine `404 Not Found` Fehlermeldung.

Example Falscher Host-Header

```
$ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
HTTP/1.1 404 Not Found
transfer-encoding: chunked
```

Example Richtiger Host-Header

```
$ ~ 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
```

- Stellen Sie sicher, dass Ihr Client den Server Name Indicator (SNI) für Anfragen, die an öffentliche oder private Dienste weitergeleitet werden, korrekt einstellt. Für die TLS-Terminierung und das Routing von Anfragen verwendet App Runner das in der HTTPS-Verbindung festgelegte SNI.

Wenn der Service keine Verbindung zu Amazon RDS oder einem Downstream-Service herstellen kann

Möglicherweise liegt ein Problem mit der Netzwerkkonfiguration bei Ihrem Service vor, wenn er keine Verbindung zu einer Amazon RDS-Datenbank oder einer anderen nachgelagerten Anwendung oder einem anderen nachgelagerten Dienst herstellen kann. In diesem Thema werden Sie durch einige Schritte geführt, um festzustellen, ob Probleme mit Ihrer Netzwerkkonfiguration vorliegen und welche Optionen zu deren Behebung zur Verfügung stehen. Weitere Informationen zur Konfiguration des ausgehenden Datenverkehrs für App Runner finden Sie unterVPC-Zugriff für ausgehenden Verkehr aktivieren .

> **Note**
>
> Um Ihre VPC Connector-Konfiguration anzuzeigen, wählen Sie im linken Navigationsbereich der App Runner-Konsole Netzwerkkonfiguration aus. Wählen Sie dann den Tab Ausgehender Verkehr aus. Wählen Sie einen VPC-Connector aus. Auf der nächsten Seite werden Details

zum VPC-Connector angezeigt. Auf dieser Seite können Sie Folgendes anzeigen und detailliert untersuchen: Subnetze, Sicherheitsgruppen und App Runner-Dienste, die die VPC verwenden.

Um die Ursache für die Unfähigkeit Ihrer Anwendung, eine Verbindung zu einem anderen Downstream-Dienst herzustellen, einzugrenzen

1. Stellen Sie sicher, dass es sich bei den in den VPC Connectors verwendeten Subnetzen um private Subnetze handelt. Wenn ein Connector mit einem öffentlichen Subnetz konfiguriert ist, treten bei Ihrem Service Fehler auf, da die zugrunde liegenden Hyperplane ENIs (elastische Netzwerkschnittstellen) für jedes Subnetz keinen öffentlichen IP-Bereich haben.

   Wenn Ihre VPC-Connectors öffentliche Subnetze verwenden, haben Sie die folgenden Optionen, um diese Konfiguration zu korrigieren:

   a. Erstellen Sie ein neues privates Subnetz und verwenden Sie es anstelle des öffentlichen Subnetzes für den VPC-Connector. Weitere Informationen finden Sie unter Subnetze für Ihre VPC im Amazon VPC-Benutzerhandbuch.

   b. Leiten Sie das bestehende öffentliche Subnetz über NAT-Gateways weiter. Weitere Informationen finden Sie unter NAT-Gateways im Amazon VPC-Benutzerhandbuch.

2. Stellen Sie sicher, dass die Sicherheitsgruppeneingangs- und Ausgangsregeln für den VPC-Connector korrekt sind. Wählen Sie im linken Navigationsbereich der App Runner-Konsole Netzwerkkonfiguration > Ausgehender Verkehr aus. Wählen Sie den VPC-Connector aus der Liste aus. Auf der nächsten Seite sind die Sicherheitsgruppen aufgeführt, die Sie zur Überprüfung auswählen können.

3. Stellen Sie sicher, dass die Regeln für eingehende und ausgehende Sicherheitsgruppen für die RDS-Instance oder einen anderen Downstream-Dienst, zu dem Sie eine Verbindung herstellen möchten, korrekt sind. Weitere Informationen finden Sie im Servicehandbuch für den Downstream-Dienst, zu dem Ihre App Runner-Anwendung versucht, eine Verbindung herzustellen.

4. Um sicherzustellen, dass außerhalb Ihrer App Runner-Konfigurationen kein anderes Netzwerk-Setup-Problem vorliegt, versuchen Sie, eine Verbindung zum RDS oder zum Downstream-Dienst außerhalb von App Runner herzustellen:

   a. Versuchen Sie von einer EC2 Amazon-Instance in derselben VPC aus, eine Verbindung zur RDS-Instance oder zum RDS-Dienst herzustellen.

b. Wenn Sie versuchen, eine Verbindung zu einem Service-VPC-Endpunkt herzustellen, überprüfen Sie die Konnektivität, indem Sie von einer EC2 Instanz in derselben VPC auf denselben Endpunkt zugreifen.

5. Wenn einer der Verbindungstests in Schritt 4 fehlschlägt, liegt höchstwahrscheinlich ein Problem außerhalb Ihrer App Runner-Konfigurationen mit einer anderen Ressource in Ihrem AWS Konto vor. Wenden Sie sich an den AWS Support, um Unterstützung bei der weiteren Isolierung und Behebung des Problems mit Ihren anderen Netzwerkkonfigurationen zu erhalten.
6. Wenn Sie mithilfe der Anweisungen in Schritt 4 erfolgreich eine Verbindung mit der RDS-Instance oder dem Downstream-Dienst hergestellt haben, fahren Sie mit den Anweisungen in diesem Schritt fort. Wir überprüfen, ob Datenverkehr in das ENI eingeht, indem wir die Hyperplane ENI Flow-Logs aktivieren und überprüfen.

> **Note**
>
> Um diese Schritte ausführen und die erforderlichen ENI-Datenflussprotokollinformationen abrufen zu können, muss der Verbindungsversuch zum RDS- oder Downstream-Dienst erfolgen, nachdem Ihr App Runner-Dienst erfolgreich gestartet wurde. Ihre Anwendung muss den Verbindungsvorgang mit dem RDS oder dem Downstream-Dienst ausführen, wenn sie sich im Status Running befindet. Andernfalls ENIs könnten sie im Rahmen der Rollback-Workflows von App Runner bereinigt werden. Dieser Ansatz stellt sicher, dass sie ENIs weiterhin für weitere Untersuchungen zur Verfügung stehen.

a. Starten Sie die AWS Konsole von der EC2 Konsole aus.
b. Wählen Sie im linken Navigationsbereich in der Gruppe Netzwerk und Sicherheit die Option Netzwerkschnittstellen aus.
c. Scrollen Sie zu den Spalten Schnittstellentyp und Beschreibung, um die ENIs in den mit dem VPC-Connector verknüpften Subnetzen zu suchen. Sie werden die folgenden Benennungsmuster haben.
   - Schnittstellentyp: Fargate
   - Beschreibung: beginnt mit AWSAppRunner ENI(Beispiel: AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)
d. Verwenden Sie die Kontrollkästchen am Anfang der Zeilen, um die ENIs zutreffenden auszuwählen.
e. Wählen Sie im Menü Aktionen die Option Flow-Protokoll erstellen aus.

f. Geben Sie die Informationen in die Eingabeaufforderungen ein und wählen Sie unten auf der Seite die Option Flow-Flog erstellen aus.

g. Prüfen Sie das generierte Flow-Protokoll.

- Wenn beim Testen der Verbindung Datenverkehr in die ENI einging, hat das Problem nichts mit der ENI-Konfiguration zu tun. Möglicherweise gibt es Probleme mit der Netzwerkkonfiguration mit einer anderen Ressource in Ihrem AWS Konto als den App Runner-Diensten. Wenden Sie sich an den AWS Support, um weitere Unterstützung zu erhalten.
- Wenn beim Testen der Verbindung kein Datenverkehr in das ENI einging, empfehlen wir Ihnen, sich an den AWS Support zu wenden, um zu erfahren, ob es bekannte Probleme mit dem Fargate-Dienst gibt.

h. Verwenden Sie das Tool Network Reachability Analyzer. Dieses Tool hilft bei der Identifizierung von Netzwerkfehlkonfigurationen, indem blockierende Komponenten identifiziert werden, wenn eine Quelle im virtuellen Netzwerkpfad nicht erreichbar ist. Weitere Informationen finden Sie unter Was ist Reachability Analyzer? im Amazon VPC Reachability Analyzer Guide.

   Geben Sie App Runner ENI als Quelle und RDS ENI als Ziel ein.

7. Wenn Sie das Problem nicht weiter eingrenzen können oder wenn Sie nach Abschluss der vorherigen Schritte immer noch keine Verbindung zum RDS oder zum Downstream-Dienst herstellen können, empfehlen wir Ihnen, sich an den Support zu wenden, um weitere AWS Unterstützung zu erhalten.

Wenn nicht genügend IP-Adressen für den Start von Instances oder die Skalierung vorhanden sind

> **Note**
>
> Für öffentliche Dienste erstellt App Runner kein Elastic Network Interface (ENI) in Ihrem VPCs, sodass Ihre öffentlichen Dienste von dieser Änderung nicht betroffen sind.

Dieses Handbuch hilft Ihnen bei der Behebung von IP-Erschöpfungsfehlern, die bei App Runner-Diensten auftreten können, wenn der VPC-Zugriff für ausgehenden Datenverkehr aktiviert ist.

App Runner startet Instances in den Subnetzen, die mit Ihrem VPC-Connector verknüpft sind. App Runner erstellt 1 ENI pro Instance in dem Subnetz, in dem Ihre Instance gestartet wird. Jede ENI verwendet eine private IP in diesem Subnetz. Abhängig vom CIDR-Block, der diesem Subnetz zugeordnet IPs ist, steht eine feste Anzahl verfügbarer Subnetze zur Verfügung. Wenn App Runner nicht in der Lage ist, Subnetze zu finden, die ausreichen, IPs um eine ENI zu erstellen, können keine neuen Instanzen für Ihren App Runner-Dienst gestartet werden. Dies kann zu Problemen bei der Skalierung Ihrer Dienste führen. In solchen Fällen werden App Runner-Ereignisprotokolle angezeigt, die darauf hinweisen, dass App Runner keine Subnetze mit verfügbaren IPs Subnetzen finden kann. Sie können Ihre Dienste mit den folgenden Anweisungen aktualisieren, um solche Fehler zu beheben.

Wie aktualisierst du deine Dienste, um mehr verfügbar zu haben IPs

Die Anzahl der in einem Subnetz verfügbaren IP-Adressen basiert auf dem CIDR-Block, der diesem Subnetz zugeordnet ist. CIDR-Blöcke, die einem Subnetz zugeordnet sind, können nach der Erstellung nicht aktualisiert werden. App Runner VPC-Connectors können auch nicht aktualisiert werden, sobald sie erstellt wurden. IPs Um mehr aus Ihren App Runner-Diensten herauszuholen, wenn der VPC-Zugriff für ausgehenden Datenverkehr aktiviert ist, gehen Sie wie folgt vor:

1. Erstellen Sie neue Subnetze mit einem größeren CIDR-Block.
2. Erstellen Sie einen neuen VPC-Connector mit den neuen Subnetzen.
3. Aktualisieren Sie Ihren App Runner-Dienst, sodass er den neuen VPC-Connector verwendet.

Berechnung, IPs die für Ihre Dienste erforderlich ist

Bevor Sie versuchen, neue Subnetze mit größeren CIDR-Blöcken zu erstellen, ermitteln Sie die Anzahl der Subnetze, die IPs Sie für Ihre App Runner-Dienste benötigen. Wir empfehlen, die Anzahl der IPs benötigten Verbindungen in Ihrem Connector wie folgt zu berechnen:

1. Notieren Sie sich für jeden Dienst, bei dem der VPC-Zugriff für ausgehenden Datenverkehr aktiviert ist, die maximale Größe (maximale Anzahl an Instanzen) in der Auto Scaling-Konfiguration.
2. Summieren Sie die Werte für alle Dienste.
3. Verdoppeln Sie diese Summe, um die neuen Instances zu berücksichtigen, die bei Bereitstellungen mit Blau-Grün gestartet wurden.

Beispiel

Stellen Sie sich zwei Dienste A und B vor, die denselben VPC-Connector verwenden.

1. Für Dienst A ist die maximale Größe auf 25 konfiguriert.
2. Für Dienst B ist die maximale Größe auf 15 konfiguriert.

Erforderlich IPs = 2 × (25 + 15) = 80

Stellen Sie sicher, dass in Ihren Subnetzen IPs zusammen mindestens 80 verfügbar sind.

Erstellen Sie neue Subnetze

1. Ermitteln Sie die CIDR-Blockgröße, die für die IPv4 Verwendung dieser Formel erforderlich ist (beachten Sie, dass 5 von AWS reserviert IPs sind: Subnet Sizing)

```
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. Erstellen Sie mithilfe der EC2 AWS-CLI ein neues Subnetz.

```
aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

   Beispiel (erstellt ein Subnetz mit IPs 4.096):

```
aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
```

3. Erstellen Sie einen neuen VPC-Connector. Siehe: VPC-Zugriff verwalten

4. Aktualisieren Sie Ihre Dienste mit ausgehendem Datenverkehr auf VPC, die für die Verwendung dieses neuen VPC-Connectors aktiviert sind. App Runner verwendet die neuen Subnetze, sobald Ihr Dienst aktualisiert wurde.

> **Note**
>
> VPCs sind auch in Bezug auf die Anzahl der verfügbaren IPs , die den Subnetzen durch CIDR-Blöcke zugewiesen werden können, begrenzt. Wenn Sie keine Subnetze mit größeren CIDR-Blöcken erstellen können, müssen Sie Ihre VPC möglicherweise mit sekundären CIDR-Blöcken aktualisieren, bevor Sie die neuen Subnetze erstellen.

Sekundäre CIDR-Blöcke an Ihre VPC anhängen

Ordnen Sie dieser VPC einen sekundären CIDR-Block zu.

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Beispiel:

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

Verifizierung

Sobald Sie Ihren Service aktualisiert haben. Sie können Folgendes verwenden, um Ihren Fix zu überprüfen

1. Überwachen Sie die Ereignisprotokolle: Überwachen Sie die Ereignisprotokolle Ihres App Runner-Dienstes, um sicherzustellen, dass keine neuen IP- oder ENI-Nichtverfügbarkeitsfehler angezeigt werden
2. Überprüfen Sie die Service-Skalierung:
    1. Skalieren Sie den Service vollständig, indem Sie die Mindestanzahl an Instanzen in Ihrer Autoscaling-Konfiguration ändern
    2. Stellen Sie sicher, dass alle neuen Instances ohne IP-bezogene Fehler gestartet werden
    3. Überwachen Sie mehrere Skalierungsereignisse, um eine konsistente Leistung sicherzustellen

3. Konsolenbanner: Wenn Sie die AWS-Managementkonsole verwenden, stellen Sie sicher, dass App Runner kein Warnbanner mehr anzeigt, dass es nicht ausreichend ist IPs.
4. VPC- und Subnetz-IP-Nutzung:
    1. Verwenden Sie das VPC-Dashboard oder die CLI-Befehle, um die IP-Adressnutzung in Ihren neuen Subnetzen zu überprüfen.
    2. Vergewissern Sie sich, dass IPs nach der Skalierung Ihres Dienstes immer noch ein ausreichender Spielraum verfügbar ist

Häufige Fallstricke

Beachten Sie bei der Bekämpfung der IP-Erschöpfung in App Runner-Diensten die folgenden potenziellen Probleme:

1. Unzureichende IP-Adressplanung: Wenn future IP-Anforderungen unterschätzt werden, kann dies zu wiederkehrenden Problemen mit der Erschöpfung führen. Führen Sie eine gründliche Kapazitätsplanung durch und berücksichtigen Sie dabei das potenzielle Servicewachstum und Szenarien mit Spitzennutzung.
2. VPC-weite IP-Nutzung übersehen: Denken Sie daran, dass andere AWS-Services innerhalb derselben VPC ebenfalls IP-Adressen verwenden. Berücksichtigen Sie bei der Planung Ihrer VPC- und Subnetzkonfigurationen die IP-Anforderungen aller Dienste.
3. Vernachlässigung der Aktualisierung von Diensten: Stellen Sie nach dem Erstellen neuer Subnetze oder VPC-Connectors sicher, dass Sie Ihre App Runner-Dienste aktualisieren, um die neuen Konfigurationen zu verwenden. Andernfalls wird der erschöpfte IP-Bereich weiterhin genutzt.
4. Missverständnis von CIDR-Blocküberlappungen: Achten Sie beim Hinzufügen sekundärer CIDR-Blöcke zu einer VPC darauf, dass sie sich nicht mit vorhandenen Blöcken überschneiden. Überlappende CIDR-Blöcke können zu Routing-Konflikten und Mehrdeutigkeiten bei IP-Adressen führen.
5. Überschreitung der VPC-Grenzwerte: Beachten Sie, dass eine VPC maximal 5 CIDR-Blöcke (1 primärer und 4 sekundärer) haben kann. Planen Sie die Erweiterung Ihres IP-Adressraums innerhalb dieser Einschränkungen.
6. Ignorieren der Subnetz-AZ-Verteilung: Stellen Sie beim Erstellen neuer Subnetze sicher, dass diese auf mehrere Availability Zones verteilt sind, um eine hohe Verfügbarkeit und Fehlertoleranz zu gewährleisten.

7. Nichtbeachtung der ENI-Grenzwerte: Denken Sie daran, dass die Anzahl der Instances, die Instanzen zugeordnet werden können ENIs, begrenzt ist. Stellen Sie sicher, dass Ihre AWS-Kontolimits mit Ihrer geplanten Netzwerkschnittstellennutzung übereinstimmen.

Wenn Sie sich dieser Fallstricke bewusst sind, können Sie Ihre VPC-Ressourcen effektiver verwalten und Probleme mit der IP-Erschöpfung in Ihren App Runner-Diensten vermeiden.

Weitere Ressourcen

1. AWS-VPC-Dokumentation
2. CIDR-Blöcke verstehen
3. App Runner VPC-Konnektoren

Glossar

1. ENI: Elastic Network Interface, eine virtuelle Netzwerkschnittstelle in AWS.
2. CIDR: Classless Inter-Domain Routing, eine Methode zur Zuweisung von IP-Adressen.
3. VPC-Connector: Eine Ressource, mit der App Runner eine Verbindung zu Ihrer VPC herstellen kann.

Sicherheit in App Runner

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der AWS Dienste in der ausgeführt AWS Cloud werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS App Runner, finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von App Runner anwenden können. In den folgenden Themen erfahren Sie, wie Sie App Runner so konfigurieren, dass Sie Ihre Sicherheits- und Compliance-Ziele erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer App Runner-Ressourcen helfen.

Themen

- Bewährte Sicherheitsmethoden für App Runner

Datenschutz in App Runner

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS App Runner. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit App

Runner oder anderen Geräten AWS-Services über die Konsole, API oder arbeiten. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Weitere Sicherheitsthemen zu App Runner finden Sie unterSicherheit.

Themen

- Datenschutz durch Verschlüsselung
- Richtlinie für den Datenverkehr zwischen Netzwerken

Datenschutz durch Verschlüsselung

AWS App Runner liest Ihre Anwendungsquelle (Quellbild oder Quellcode) aus einem von Ihnen angegebenen Repository und speichert sie für die Bereitstellung in Ihrem Service. Weitere Informationen finden Sie unter Architektur und Konzepte.

Datenschutz bezieht sich auf den Schutz von Daten während der Übertragung (während der Übertragung zu und von App Runner) und im Ruhezustand (während sie in AWS Rechenzentren gespeichert werden).

Weitere Informationen zum Datenschutz finden Sie unterthe section called “Datenschutz”.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Verschlüsselung während der Übertragung

Sie können den Datenschutz bei der Übertragung auf zwei Arten erreichen: Verschlüsseln Sie die Verbindung mit Transport Layer Security (TLS) oder verwenden Sie die clientseitige Verschlüsselung (wobei das Objekt vor dem Senden verschlüsselt wird). Beide Methoden sind für den Schutz Ihrer Anwendungsdaten gültig. Um die Verbindung zu sichern, verschlüsseln Sie sie mit TLS, wann immer Ihre Anwendung, ihre Entwickler und Administratoren und ihre Endbenutzer Objekte senden oder empfangen. App Runner richtet Ihre Anwendung so ein, dass sie Datenverkehr über TLS empfängt.

Die clientseitige Verschlüsselung ist keine gültige Methode zum Schutz des Quell-Images oder -codes, den Sie App Runner zur Bereitstellung zur Verfügung stellen. App Runner benötigt Zugriff auf Ihre Anwendungsquelle, sodass sie nicht verschlüsselt werden kann. Stellen Sie daher sicher,

dass die Verbindung zwischen Ihrer Entwicklungs- oder Bereitstellungsumgebung und App Runner gesichert ist.

Verschlüsselung im Ruhezustand und Schlüsselverwaltung

Um die Daten Ihrer Anwendung im Ruhezustand zu schützen, verschlüsselt App Runner alle gespeicherten Kopien Ihres Quellimages oder Quellpakets der Anwendung. Wenn Sie einen App Runner-Dienst erstellen, können Sie einen AWS KMS key bereitstellen. Wenn Sie einen angeben, verwendet App Runner Ihren bereitgestellten Schlüssel, um Ihre Quelle zu verschlüsseln. Wenn Sie keinen angeben, verwendet App Runner Von AWS verwalteter Schlüssel stattdessen einen.

Einzelheiten zu den Parametern für die Erstellung des App Runner-Dienstes finden Sie unter CreateService. Informationen zu AWS Key Management Service (AWS KMS) finden Sie im AWS Key Management Service Entwicklerhandbuch.

Richtlinie für den Datenverkehr zwischen Netzwerken

App Runner verwendet Amazon Virtual Private Cloud (Amazon VPC), um Grenzen zwischen Ressourcen in Ihrer App Runner-Anwendung zu erstellen und den Verkehr zwischen ihnen, Ihrem lokalen Netzwerk und dem Internet zu kontrollieren. Weitere Informationen zur Amazon VPC-Sicherheit finden Sie unter Datenschutz im Netzwerkdatenverkehr in Amazon VPC im Amazon VPC-Benutzerhandbuch.

Informationen zur Verknüpfung Ihrer App Runner-Anwendung mit einer benutzerdefinierten Amazon VPC finden Sie unter. the section called “Ausgehender Verkehr”

Informationen zum Sichern von Anfragen an App Runner mithilfe eines VPC-Endpunkts finden Sie unterthe section called “VPC-Endpunkte”.

Weitere Informationen zum Datenschutz finden Sie unterthe section called “Datenschutz”.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Identitäts- und Zugriffsmanagement für App Runner

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um App Runner-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in App Runner ausführen.

Dienstbenutzer — Wenn Sie den App Runner-Dienst für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Wenn Sie für Ihre Arbeit mehr App Runner-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie in App Runner nicht auf eine Funktion zugreifen können, finden Sie weitere Informationen unterFehlerbehebung bei Identität und Zugriff auf App Runner.

Dienstadministrator — Wenn Sie in Ihrem Unternehmen für die App Runner-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf App Runner. Es ist Ihre Aufgabe, zu bestimmen, auf welche App Runner-Funktionen und -Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit App Runner nutzen kann, finden Sie unterSo funktioniert App Runner mit IAM.

IAM-Administrator — Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf App Runner zu verwalten.

Beispiele für identitätsbasierte App Runner-Richtlinien, die Sie in IAM verwenden können, finden Sie unter. Beispiele für identitätsbasierte App Runner-Richtlinien

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und

dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb von Ihnen AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM

erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich

auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

So funktioniert App Runner mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf zu verwalten AWS App Runner, sollten Sie wissen, welche IAM-Funktionen für App Runner verfügbar sind. Einen allgemeinen Überblick darüber, wie App Runner und andere AWS Dienste mit IAM funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

Weitere Sicherheitsthemen zu App Runner finden Sie unter. Sicherheit

Themen

- Identitätsbasierte Richtlinien von App Runner
- Ressourcenbasierte App Runner-Richtlinien
- Autorisierung basierend auf App Runner-Tags
- App Runner-Benutzerberechtigungen
- App Runner IAM-Rollen

Identitätsbasierte Richtlinien von App Runner

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. App Runner unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Richtlinienaktionen in App Runner verwenden das folgende Präfix vor der Aktion:`apprunner:`. Um beispielsweise jemandem die Erlaubnis zu erteilen, eine EC2 Amazon-Instance mit dem EC2 `RunInstances` Amazon-API-Vorgang auszuführen, nehmen Sie die `ec2:RunInstances` Aktion in seine Richtlinie auf. Richtlinienanweisungen müssen entweder ein – `Action`oder ein `NotAction`-Element enthalten. App Runner definiert eigene Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service ausführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "apprunner:Describe*"
```

Eine Liste der App Runner-Aktionen finden Sie unter Aktionen definiert von AWS App Runner in der Referenz zur Serviceautorisierung.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

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

App Runner-Ressourcen haben die folgende ARN-Struktur:

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces in der. Allgemeine AWS-Referenz

Um beispielsweise den `my-service` Service in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN:

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

Um alle Dienste anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

Einige App Runner-Aktionen, z. B. zum Erstellen von Ressourcen, können nicht für eine bestimmte Ressource ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

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

Eine Liste der App Runner-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Ressourcen definiert von AWS App Runner in der Service Authorization Reference. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von AWS App Runner definierte Aktionen.

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

App Runner unterstützt die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

App Runner definiert eine Reihe von dienstspezifischen Bedingungsschlüsseln. Darüber hinaus unterstützt App Runner die Tag-basierte Zugriffskontrolle, die mithilfe von Bedingungsschlüsseln implementiert wird. Details hierzu finden Sie unter the section called “Autorisierung basierend auf App Runner-Tags”.

Eine Liste der App Runner-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS App Runner in der Serviceautorisierungsreferenz. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Aktionen definiert von AWS App Runner.

Beispiele

Beispiele für identitätsbasierte App Runner-Richtlinien finden Sie unter. Beispiele für identitätsbasierte App Runner-Richtlinien

Ressourcenbasierte App Runner-Richtlinien

App Runner unterstützt keine ressourcenbasierten Richtlinien.

Autorisierung basierend auf App Runner-Tags

Sie können Tags an App Runner-Ressourcen anhängen oder Tags in einer Anfrage an App Runner übergeben. Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `apprunner:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden. Weitere Informationen zum Taggen von App Runner-Ressourcen finden Sie unterthe section called “Konfiguration”.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Steuern des Zugriffs auf App Runner-Dienste anhand von Tags.

App Runner-Benutzerberechtigungen

Um App Runner verwenden zu können, benötigen IAM-Benutzer Berechtigungen für App Runner-Aktionen. Eine gängige Methode, Benutzern Berechtigungen zu gewähren, besteht darin, IAM-

Benutzern oder -Gruppen eine Richtlinie zuzuweisen. Weitere Informationen zur Verwaltung von Benutzerberechtigungen finden Sie unter Ändern von Berechtigungen für einen IAM-Benutzer im IAM-Benutzerhandbuch.

App Runner bietet zwei verwaltete Richtlinien, die Sie Ihren Benutzern zuordnen können.

- `AWSAppRunnerReadOnlyAccess`— Erteilt Berechtigungen zum Auflisten und Anzeigen von Details zu App Runner-Ressourcen.
- `AWSAppRunnerFullAccess`— Erteilt Berechtigungen für alle App Runner-Aktionen.

Für eine detailliertere Kontrolle der Benutzerberechtigungen können Sie eine benutzerdefinierte Richtlinie erstellen und sie an Ihre Benutzer anhängen. Einzelheiten finden Sie im IAM-Benutzerhandbuch unter Erstellen von IAM-Richtlinien.

Beispiele für Benutzerrichtlinien finden Sie unter. the section called “Benutzerrichtlinien”

AWSAppRunnerReadOnlyAccess

JSON

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

AWSAppRunnerFullAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/apprunner.amazonaws.com/
AWSServiceRoleForAppRunner",
        "arn:aws:iam::*:role/aws-service-role/networking.apprunner.amazonaws.com/
AWSServiceRoleForAppRunnerNetworking"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": [
            "apprunner.amazonaws.com",
            "networking.apprunner.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "apprunner.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AppRunnerAdminAccess",
      "Effect": "Allow",
      "Action": "apprunner:*",
      "Resource": "*"
    }
  ]
}
```

App Runner IAM-Rollen

Eine IAM-Rolle ist eine Entität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt.

Service-verknüpfte Rollen

Mit dienstbezogenen Rollen können AWS Dienste auf Ressourcen in anderen Diensten zugreifen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

App Runner unterstützt dienstbezogene Rollen. Informationen zum Erstellen oder Verwalten von dienstbezogenen App Runner-Rollen finden Sie unter. the section called “Verwenden von serviceverknüpften Rollen”

Servicerollen

Dieses Feature ermöglicht einem Service das Annehmen einer Servicerolle in Ihrem Namen. Diese Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Diensten, um eine Aktion in Ihrem Namen auszuführen. Servicerollen werden in Ihrem IAM-Konto angezeigt und gehören zum Konto. Das bedeutet, dass ein IAM-Benutzer die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktionalität des Dienstes beeinträchtigen.

App Runner unterstützt einige Servicerollen.

Rolle „Zugriff“

Die Zugriffsrolle ist eine Rolle, die App Runner für den Zugriff auf Bilder in Amazon Elastic Container Registry (Amazon ECR) in Ihrem Konto verwendet. Es ist erforderlich, um auf ein Bild in Amazon ECR zuzugreifen, und ist bei Amazon ECR Public nicht erforderlich. Bevor Sie einen Service erstellen, der auf einem Image in Amazon ECR basiert, verwenden Sie IAM, um eine Servicerolle zu erstellen und die darin enthaltene `AWSAppRunnerServicePolicyForECRAccess` verwaltete Richtlinie zu verwenden. Sie können diese Rolle dann an App Runner übergeben, wenn Sie die CreateServiceAPI im AuthenticationConfigurationMitglied des SourceConfigurationParameters aufrufen oder wenn Sie die App Runner-Konsole verwenden, um einen Service zu erstellen.

AWSAppRunnerServicePolicyForECRAccess

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Wenn Sie Ihre eigene benutzerdefinierte Richtlinie für Ihre Zugriffsrolle erstellen, stellen Sie sicher, dass Sie diese `"Resource": "*"` für die `ecr:GetAuthorizationToken` Aktion angeben. Tokens können für den Zugriff auf jede Amazon ECR-Registrierung verwendet werden, auf die Sie Zugriff haben.

Achten Sie beim Erstellen Ihrer Zugriffsrolle darauf, eine Vertrauensrichtlinie hinzuzufügen, die den App Runner-Service Principal `build.apprunner.amazonaws.com` als vertrauenswürdige Entität deklariert.

Vertrauensrichtlinie für eine Zugriffsrolle

JSON

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

```
}
```

Wenn Sie die App Runner-Konsole verwenden, um einen Dienst zu erstellen, kann die Konsole automatisch eine Zugriffsrolle für Sie erstellen und diese für den neuen Dienst auswählen. In der Konsole werden auch andere Rollen in Ihrem Konto aufgeführt, und Sie können bei Bedarf eine andere Rolle auswählen.

Instanzrolle

Die Instanzrolle ist eine optionale Rolle, die App Runner verwendet, um Berechtigungen für AWS Serviceaktionen bereitzustellen, die die Recheninstanzen Ihres Dienstes benötigen. Sie müssen App Runner eine Instanzrolle zur Verfügung stellen, wenn Ihr Anwendungscode AWS actions (APIs) aufruft. Betten Sie entweder die erforderlichen Berechtigungen in Ihre Instanzrolle ein oder erstellen Sie Ihre eigene benutzerdefinierte Richtlinie und verwenden Sie sie in der Instanzrolle. Wir können nicht vorhersehen, welche Aufrufe Ihr Code verwendet. Daher bieten wir zu diesem Zweck keine verwaltete Richtlinie an.

Bevor Sie einen App Runner-Dienst erstellen, verwenden Sie IAM, um eine Servicerolle mit den erforderlichen benutzerdefinierten oder eingebetteten Richtlinien zu erstellen. Sie können diese Rolle dann als Instanzrolle an App Runner übergeben, wenn Sie die CreateServiceAPI im `InstanceRoleArn` Member des InstanceConfigurationParameters aufrufen oder wenn Sie die App Runner-Konsole verwenden, um einen Dienst zu erstellen.

Achten Sie beim Erstellen Ihrer Instanzrolle darauf, eine Vertrauensrichtlinie hinzuzufügen, die den App Runner-Dienstprinzipal `tasks.apprunner.amazonaws.com` als vertrauenswürdige Entität deklariert.

Vertrauensrichtlinie für eine Instanzrolle

JSON

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

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Wenn Sie die App Runner-Konsole verwenden, um einen Dienst zu erstellen, listet die Konsole die Rollen in Ihrem Konto auf, und Sie können die Rolle auswählen, die Sie zu diesem Zweck erstellt haben.

Informationen zum Erstellen eines Dienstes finden Sie unterthe section called “Erstellung”.

Beispiele für identitätsbasierte App Runner-Richtlinien

Standardmäßig sind IAM-Benutzer und -Rollen nicht berechtigt, Ressourcen zu erstellen oder zu ändern. AWS App Runner Sie können auch keine Aufgaben mit der AWS Management Console AWS CLI, oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den IAM-Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Weitere Sicherheitsthemen zu App Runner finden Sie unterSicherheit.

Themen

- Bewährte Methoden für Richtlinien
- Benutzerrichtlinien
- Steuern des Zugriffs auf App Runner-Dienste anhand von Tags

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand App Runner-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Benutzerrichtlinien

Um auf die App Runner-Konsole zugreifen zu können, müssen IAM-Benutzer über Mindestberechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den App Runner-Ressourcen in Ihrem AWS-Konto aufzulisten und anzuzeigen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die restriktiver ist als die erforderlichen Mindestberechtigungen, funktioniert die Konsole für Benutzer mit dieser Richtlinie nicht wie vorgesehen.

App Runner bietet zwei verwaltete Richtlinien, die Sie Ihren Benutzern zuordnen können.

- `AWSAppRunnerReadOnlyAccess`— Erteilt Berechtigungen zum Auflisten und Anzeigen von Details zu App Runner-Ressourcen.
- `AWSAppRunnerFullAccess`— Erteilt Berechtigungen für alle App Runner-Aktionen.

Um sicherzustellen, dass Benutzer die App Runner-Konsole verwenden können, fügen Sie den Benutzern mindestens die `AWSAppRunnerReadOnlyAccess` verwaltete Richtlinie bei. Sie können stattdessen die `AWSAppRunnerFullAccess` verwaltete Richtlinie anhängen oder bestimmte zusätzliche Berechtigungen hinzufügen, damit Benutzer Ressourcen erstellen, ändern und löschen können. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Erlauben Sie stattdessen nur den Zugriff auf die Aktionen, die dem API-Vorgang entsprechen, den Sie Benutzern ermöglichen möchten.

Die folgenden Beispiele veranschaulichen benutzerdefinierte Benutzerrichtlinien. Sie können sie als Ausgangspunkt für die Definition Ihrer eigenen benutzerdefinierten Benutzerrichtlinien verwenden. Kopieren Sie das Beispiel und/oder entfernen Sie Aktionen, reduzieren Sie Ressourcen und fügen Sie Bedingungen hinzu.

Beispiel: Benutzerrichtlinie für die Konsolen- und Verbindungsverwaltung

Diese Beispielrichtlinie ermöglicht den Konsolenzugriff und ermöglicht die Erstellung und Verwaltung von Verbindungen. Sie erlaubt nicht die Erstellung und Verwaltung von App Runner-Diensten. Es kann einem Benutzer zugewiesen werden, dessen Rolle darin besteht, den Zugriff des App Runner-Dienstes auf Quellcode-Assets zu verwalten.

JSON

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

Beispiel: Benutzerrichtlinien, die Bedingungsschlüssel verwenden

Die Beispiele in diesem Abschnitt veranschaulichen bedingte Berechtigungen, die von einigen Ressourceneigenschaften oder Aktionsparametern abhängen.

Diese Beispielrichtlinie ermöglicht die Erstellung eines App Runner-Dienstes, verweigert jedoch die Verwendung einer Verbindung mit dem Namen`prod`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
      "Effect": "Allow",
      "Action": ":CreateService",
      "Resource": "*",
      "Condition": {
        "ArnNotLike": {
          ":ConnectionArn": "arn:aws::*:*:connection/prod/*"
        }
```

```
      }
    }
  ]
}
```

Diese Beispielrichtlinie ermöglicht die Aktualisierung eines App Runner-Dienstes, der `preprod` nur mit einer Auto Scaling-Konfiguration benannt ist`preprod`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
      "Effect": "Allow",
      "Action": ":UpdateService",
      "Resource": "arn:aws::*:*:service/preprod/*",
      "Condition": {
        "ArnLike": {
          ":AutoScalingConfigurationArn": "arn:aws::*:*:autoscalingconfiguration/
preprod/*"
        }
      }
    }
  ]
}
```

Steuern des Zugriffs auf App Runner-Dienste anhand von Tags

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf App Runner-Ressourcen anhand von Tags zu steuern. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die das Löschen eines App Runner-Dienstes ermöglicht. Die Berechtigung wird jedoch nur gewährt, wenn der Wert des `Owner`-Tags der Name des Benutzers ist. Diese Richtlinie gewährt auch die Berechtigungen, die für die Ausführung dieser Aktion auf der Konsole erforderlich sind.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": ":ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": ":DeleteService",
      "Resource": "arn:aws::*:*:service/*",
      "Condition": {
        "StringEquals": {":ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

Sie können diese Richtlinie den IAM-Benutzern in Ihrem Konto anfügen. Wenn ein benannter Benutzer `richard-roe` versucht, einen App Runner-Dienst zu löschen, muss der Dienst mit `0wner=richard-roe` oder gekennzeichnet werden`owner=richard-roe`. Andernfalls wird der Zugriff abgelehnt. Der Tag-Schlüssel `0wner` der Bedingung stimmt sowohl mit `0wner` als auch mit `owner` überein, da die Namen von Bedingungsschlüsseln nicht zwischen Groß- und Kleinschreibung unterscheiden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

Verwenden von dienstbezogenen Rollen für App Runner

AWS App Runner verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine dienstverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, die direkt mit App Runner verknüpft ist. Dienstbezogene Rollen sind von App Runner vordefiniert und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Themen

- Rollen für die Verwaltung verwenden
- Rollen für Netzwerke verwenden

Rollen für die Verwaltung verwenden

AWS App Runner verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine dienstverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, die direkt mit App Runner verknüpft ist. Dienstbezogene Rollen sind von App Runner vordefiniert und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstverknüpfte Rolle erleichtert die Einrichtung von App Runner, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. App Runner definiert die Berechtigungen seiner dienstbezogenen Rollen, und sofern nicht anders definiert, kann nur App Runner seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre App Runner-Ressourcen geschützt, da Sie die Zugriffsberechtigung für die Ressourcen nicht versehentlich entfernen können.

Informationen zu anderen Services, die serviceverknüpften Rollen unterstützen, finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Mit dem Dienst verknüpfte Rollenberechtigungen für App Runner

App Runner verwendet die angegebene dienstverknüpfte Rolle. AWSServiceRoleForAppRunner

Mit dieser Rolle kann App Runner die folgenden Aufgaben ausführen:

- Senden Sie Protokolle an Amazon CloudWatch Logs-Protokollgruppen weiter.
- Erstellen Sie Amazon CloudWatch Events-Regeln, um Image-Pushs von Amazon Elastic Container Registry (Amazon ECR) zu abonnieren.
- Senden Sie die Nachverfolgungsinformationen an. AWS X-Ray

Die AWSService RoleForAppRunner dienstverknüpfte Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `apprunner.amazonaws.com`

Die Berechtigungsrichtlinien der AWSService RoleForAppRunner dienstbezogenen Rolle enthalten alle Berechtigungen, die App Runner benötigt, um Aktionen in Ihrem Namen auszuführen.

AppRunnerServiceRolePolicy verwaltete Richtlinie

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutRetentionPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/apprunner/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/apprunner/*:log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets",
        "events:DescribeRule",
        "events:EnableRule",
        "events:DisableRule"
      ],
```

```
      "Resource": "arn:aws:events:*:*:rule/AWSAppRunnerManagedRule*"
    }
  ]
}
```

Richtlinie für die Röntgenverfolgung

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigungen im IAM-Benutzerhandbuch.

Eine dienstbezogene Rolle für App Runner erstellen

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie einen App Runner-Dienst in der AWS Management Console, der oder der AWS API erstellen AWS CLI, erstellt App Runner die dienstverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie einen App Runner-Dienst erstellen, erstellt App Runner die dienstverknüpfte Rolle erneut für Sie.

Eine dienstverknüpfte Rolle für App Runner bearbeiten

App Runner erlaubt es Ihnen nicht, die AWSService RoleForAppRunner dienstverknüpfte Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer dienstverknüpften Rolle für App Runner

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch Ihre serviceverknüpfte Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

Bereinigen einer serviceverknüpften Rolle

Bevor mit IAM eine serviceverknüpfte Rolle löschen können, müssen Sie zunächst alle von der Rolle verwendeten Ressourcen löschen.

In App Runner bedeutet dies, dass Sie alle App Runner-Dienste in Ihrem Konto löschen. Informationen zum Löschen von App Runner-Diensten finden Sie unterthe section called “Löschung”.

> **Note**
>
> Wenn der App Runner-Dienst die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Manuelles Löschen der -serviceverknüpften Rolle

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die mit dem AWSService RoleForAppRunner Dienst verknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

Unterstützte Regionen für dienstverknüpfte App Runner-Rollen

App Runner unterstützt die Verwendung von dienstbezogenen Rollen in allen Regionen, in denen der Dienst verfügbar ist. Weitere Informationen finden Sie unter AWS App Runner -Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Rollen für Netzwerke verwenden

AWS App Runner verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine dienstverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, die direkt mit App Runner verknüpft ist. Dienstbezogene Rollen sind von App Runner vordefiniert und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstverknüpfte Rolle erleichtert die Einrichtung von App Runner, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. App Runner definiert die Berechtigungen seiner dienstbezogenen Rollen, und sofern nicht anders definiert, kann nur App Runner seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre App Runner-Ressourcen geschützt, da Sie die Zugriffsberechtigung für die Ressourcen nicht versehentlich entfernen können.

Informationen zu anderen Services, die serviceverknüpften Rollen unterstützen, finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Mit dem Dienst verknüpfte Rollenberechtigungen für App Runner

App Runner verwendet die angegebene dienstverknüpfte Rolle. AWSServiceRoleForAppRunnerNetworking

Mit dieser Rolle kann App Runner die folgenden Aufgaben ausführen:

- Hängen Sie eine VPC an Ihren App Runner-Dienst an und verwalten Sie Netzwerkschnittstellen.

Die mit dem AWSService RoleForAppRunnerNetworking Dienst verknüpfte Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `networking.apprunner.amazonaws.com`

Die genannte Richtlinie für Rollenberechtigungen AppRunnerNetworkingServiceRolePolicy enthält alle Berechtigungen, die App Runner benötigt, um Aktionen in Ihrem Namen auszuführen.

AppRunnerNetworkingServiceRolePolicy

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateNetworkInterface",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "AWSAppRunnerManaged"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
```

```
        },
        "StringLike": {
          "aws:RequestTag/AWSAppRunnerManaged": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DeleteNetworkInterface",
      "Resource": "*",
      "Condition": {
        "Null": {
          "ec2:ResourceTag/AWSAppRunnerManaged": "false"
        }
      }
    }
  ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigungen im IAM-Benutzerhandbuch.

Eine dienstbezogene Rolle für App Runner erstellen

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie einen VPC-Connector in der AWS Management Console, der oder der AWS API erstellen AWS CLI, erstellt App Runner die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie einen VPC-Connector erstellen, erstellt App Runner die serviceverknüpfte Rolle erneut für Sie.

Eine dienstverknüpfte Rolle für App Runner bearbeiten

App Runner erlaubt es Ihnen nicht, die AWSService RoleForAppRunnerNetworking dienstverknüpfte Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer dienstverknüpften Rolle für App Runner

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch Ihre serviceverknüpfte Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

Bereinigen einer serviceverknüpften Rolle

Bevor mit IAM eine serviceverknüpfte Rolle löschen können, müssen Sie zunächst alle von der Rolle verwendeten Ressourcen löschen.

In App Runner bedeutet dies, dass Sie die VPC-Connectors von allen App Runner-Diensten in Ihrem Konto trennen und die VPC-Connectors löschen. Weitere Informationen finden Sie unter the section called “Ausgehender Verkehr”.

> **Note**
>
> Wenn der App Runner-Dienst die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Manuelles Löschen der serviceverknüpften Rolle

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die mit dem AWSService RoleForAppRunnerNetworking Dienst verknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

Unterstützte Regionen für dienstverknüpfte App Runner-Rollen

App Runner unterstützt die Verwendung von dienstbezogenen Rollen in allen Regionen, in denen der Dienst verfügbar ist. Weitere Informationen finden Sie unter AWS App Runner -Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

AWS verwaltete Richtlinien für AWS App Runner

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige

Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

App Runner aktualisiert AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für App Runner an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Abonnieren Sie den RSS-Feed auf der Seite mit dem App Runner-Dokumentenverlauf, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderung	Beschreibung	Datum
AWSAppRunnerReadOnlyAccess – Neue Richtlinie	App Runner hat eine neue Richtlinie hinzugefügt, die es Benutzern ermöglicht, Details zu App Runner-Ressourcen aufzulisten und anzuzeigen.	24. Februar 2022
AWSAppRunnerFullAccess – Aktualisierung auf eine bestehende Richtlinie	App Runner hat die Ressourcenliste für die `iam:CreateServiceLinkedRole` Aktion aktualisiert, um die Erstellung einer `AWSServiceRoleForA`	8. Februar 2022

Änderung	Beschreibung	Datum
	`ppRunnerNetworking` dienstbez ogenen Rolle zu ermöglichen.	
AppRunnerNetworkingServiceRolePolicy – Neue Richtlinie	App Runner hat eine neue Richtlinie hinzugefügt, die es App Runner ermöglich t, Aufrufe an Amazon Virtual Private Cloud zu tätigen, um eine VPC an Ihren App Runner-Service anzuhängen und Netzwerkschnittstellen im Namen der App Runner-Dienste zu verwalten. Die Richtlinie wird in der `AWSServic eRoleForAppRunnerNetworking` serviceverknüpften Rolle verwendet.	8. Februar 2022
AWSAppRunnerFullAccess – Neue Richtlinie	App Runner hat eine neue Richtlinie hinzugefügt, die es Benutzern ermöglicht, alle App Runner-Aktionen durchzuführen.	10. Januar 2022
AppRunnerServiceRolePolicy – Neue Richtlinie	App Runner hat eine neue Richtlinie hinzugefügt, die es App Runner ermöglich t, Amazon CloudWatch Logs und Amazon CloudWatch Events im Namen der App Runner-Dienste aufzurufen. Die Richtlini e wird in der `AWSServiceRoleForA ppRunner` serviceverknüpften Rolle verwendet.	1. März 2021
AWSAppRunnerServicePolicyFo rECRAccess – Neue Richtlinie	App Runner hat eine neue Richtlinie hinzugefügt, die App Runner den Zugriff auf Amazon Elastic Container Registry (Amazon ECR) -Images in Ihrem Konto ermöglicht.	1. März 2021
App Runner hat begonnen, Änderungen zu verfolgen	App Runner hat damit begonnen, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	1. März 2021

Fehlerbehebung bei Identität und Zugriff auf App Runner

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS App Runner und IAM auftreten können.

Weitere Sicherheitsthemen zu App Runner finden Sie unterSicherheit.

Themen

- Ich bin nicht autorisiert, eine Aktion in App Runner durchzuführen
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine App Runner-Ressourcen ermöglichen

Ich bin nicht autorisiert, eine Aktion in App Runner durchzuführen

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion auszuführen, wenden Sie sich an Ihren Administrator, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihre AWS Anmeldeinformationen zur Verfügung gestellt hat.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um Details zu einem App Runner-Dienst anzuzeigen, aber keine Berechtigungen hat. `apprunner:DescribeService`

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

In diesem Fall bittet Mary ihren Administrator, ihre Richtlinien zu aktualisieren, damit sie mithilfe der `apprunner:DescribeService` Aktion auf die *`my-example-service`* Ressource zugreifen kann.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine App Runner-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob App Runner diese Funktionen unterstützt, finden Sie unter. So funktioniert App Runner mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Zugriff auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Protokollierung und Überwachung in App Runner

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihres AWS App Runner Dienstes. Durch die Erfassung von Überwachungsdaten aus allen Teilen Ihrer AWS Lösung können Sie einen Fehler leichter debuggen, falls einer auftritt. App Runner lässt sich in mehrere AWS Tools integrieren, mit denen Sie Ihre App Runner-Dienste überwachen und auf potenzielle Vorfälle reagieren können.

CloudWatch Amazon-Alarme

Mit CloudWatch Amazon-Alarmen können Sie eine Servicemetrik über einen von Ihnen festgelegten Zeitraum beobachten. Wenn die Metrik für eine bestimmte Anzahl von Zeiträumen einen bestimmten Schwellenwert überschreitet, erhalten Sie eine Benachrichtigung.

App Runner sammelt eine Vielzahl von Metriken über den Service als Ganzes und die Instanzen (Skalierungseinheiten), auf denen Ihr Webservice ausgeführt wird. Weitere Informationen finden Sie unter Metriken (CloudWatch).

Anwendungsprotokolle

App Runner sammelt die Ausgabe Ihres Anwendungscodes und streamt sie an Amazon CloudWatch Logs. Was in dieser Ausgabe enthalten ist, liegt bei Ihnen. Sie könnten beispielsweise detaillierte Aufzeichnungen von Anfragen an Ihren Webservice hinzufügen.

Diese Protokollaufzeichnungen könnten sich bei Sicherheits- und Zugriffsprüfungen als nützlich erweisen. Weitere Informationen finden Sie unter Protokolle (CloudWatch Protokolle).

AWS CloudTrail Aktionsprotokolle

App Runner ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in App Runner ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe für App Runner als Ereignisse. Sie können die neuesten Ereignisse in der CloudTrail Konsole anzeigen und einen Trail erstellen, um die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon Simple Storage Service (Amazon S3) -Bucket zu ermöglichen. Weitere Informationen finden Sie unter API-Aktionen (CloudTrail).

Konformitätsprüfung für App Runner

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS App Runner Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Compliance und Governance im Bereich Sicherheit – In diesen Anleitungen für die Lösungsimplementierung werden Überlegungen zur Architektur behandelt. Außerdem werden Schritte für die Bereitstellung von Sicherheits- und Compliance-Features beschrieben.
- Referenz für berechtigte HIPAA-Services – Listet berechtigte HIPAA-Services auf. Nicht alle AWS-Services sind HIPAA-fähig.
- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.

- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerelementreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Resilienz in App Runner

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

AWS App Runner verwaltet und automatisiert die Nutzung der AWS globalen Infrastruktur in Ihrem Namen. Wenn Sie App Runner verwenden, profitieren Sie von den verfügbaren Verfügbarkeits- und Fehlertoleranzmechanismen. AWS

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Infrastruktursicherheit in AWS App Runner

Als verwalteter Service AWS App Runner ist er durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf App Runner zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

App Runner mit VPC-Endpunkten verwenden

Ihre AWS Anwendung integriert möglicherweise AWS App Runner Dienste mit anderen AWS-Services , die in einer VPC von Amazon Virtual Private Cloud (Amazon VPC) ausgeführt werden. Teile Ihrer Anwendung stellen möglicherweise Anfragen von der VPC aus an App Runner. Dies können Sie beispielsweise für die kontinuierliche Bereitstellung AWS CodePipeline in Ihrem App Runner-Dienst verwenden. Eine Möglichkeit, die Sicherheit Ihrer Anwendung zu verbessern, besteht darin, diese App Runner-Anfragen (und Anfragen an andere AWS-Services) über einen VPC-Endpunkt zu senden.

Mithilfe eines VPC-Endpunkts können Sie Ihre VPC privat mit unterstützten AWS-Services und VPC-Endpunktdiensten verbinden, die von unterstützt werden. AWS PrivateLink Sie benötigen kein Internet-Gateway, kein NAT-Gerät, keine VPN-Verbindung oder Verbindung. AWS Direct Connect

Ressourcen in Ihrer VPC verwenden keine öffentlichen IP-Adressen, um mit App Runner-Ressourcen zu interagieren. Der Datenverkehr zwischen Ihrer VPC und App Runner verlässt das Amazon-Netzwerk nicht. Weitere Informationen zu VPC-Endpunkten finden Sie im Handbuch unter VPC-Endpoints.AWS PrivateLink

> **Note**
>
> Standardmäßig wird die Webanwendung in Ihrem App Runner-Dienst in einer VPC ausgeführt, die App Runner bereitstellt und konfiguriert. Diese VPC ist öffentlich. Das bedeutet, dass sie mit dem Internet verbunden ist. Sie können Ihre Anwendung optional einer benutzerdefinierten VPC zuordnen. Weitere Informationen finden Sie unter the section called “Ausgehender Verkehr”.
> Sie können Ihre Dienste so konfigurieren, dass sie auf das Internet zugreifen AWS APIs, auch wenn Ihr Dienst mit einer VPC verbunden ist. Anweisungen zum Aktivieren des öffentlichen Internetzugangs für ausgehenden VPC-Verkehr finden Sie unter. the section called “Überlegungen bei der Auswahl eines Subnetzes ”
> App Runner unterstützt nicht die Erstellung eines VPC-Endpunkts für Ihre Anwendung.

Einen VPC-Endpunkt für App Runner einrichten

Um den Schnittstellen-VPC-Endpunkt für den App Runner-Dienst in Ihrer VPC zu erstellen, folgen Sie dem Verfahren Schnittstellen-Endpunkt erstellen im AWS PrivateLink Handbuch. Wählen Sie für Servicename `com.amazonaws.region.apprunner` aus.

Überlegungen zum Datenschutz in VPC-Netzwerken

> **Important**
>
> Die Verwendung eines VPC-Endpunkts für App Runner stellt nicht sicher, dass der gesamte Datenverkehr von Ihrer VPC vom Internet ferngehalten wird. Die VPC ist möglicherweise öffentlich. Darüber hinaus verwenden einige Teile Ihrer Lösung möglicherweise keine VPC-Endpunkte für AWS API-Aufrufe. Sie AWS-Services könnten beispielsweise andere Dienste

über ihre öffentlichen Endpunkte aufrufen. Wenn für die Lösung in Ihrer VPC Datenschutz erforderlich ist, lesen Sie diesen Abschnitt.

Beachten Sie Folgendes, um die Vertraulichkeit des Netzwerkverkehrs in Ihrer VPC zu gewährleisten:

- DNS-Namen aktivieren — Teile Ihrer Anwendung senden möglicherweise weiterhin Anfragen über das Internet über den `apprunner.region.amazonaws.com` öffentlichen Endpunkt an App Runner. Wenn Ihre VPC mit Internetzugang konfiguriert ist, sind diese Anfragen erfolgreich, ohne dass Sie etwas davon erfahren. Sie können dies verhindern, indem Sie sicherstellen, dass die Option DNS-Name aktivieren aktiviert ist, wenn Sie den Endpunkt erstellen. Standardmäßig ist er auf „true“ gesetzt. Hierdurch wird ein DNS-Eintrag in Ihrer VPC hinzugefügt, der den Endpunkt für den öffentlichen Service dem VPC-Schnittstellenendpunkt zuordnet.
- VPC-Endpunkte für zusätzliche Dienste konfigurieren — Ihre Lösung sendet möglicherweise Anfragen an andere. AWS-Services AWS CodePipeline Könnte beispielsweise Anfragen senden an. AWS CodeBuild Konfigurieren Sie VPC-Endpunkte für diese Dienste und aktivieren Sie DNS-Namen auf diesen Endpunkten.
- Eine private VPC konfigurieren — Wenn möglich (wenn Ihre Lösung überhaupt keinen Internetzugang benötigt), richten Sie Ihre VPC als privat ein, was bedeutet, dass sie keine Internetverbindung hat. Dadurch wird sichergestellt, dass ein fehlender VPC-Endpunkt einen sichtbaren Fehler verursacht, sodass Sie den fehlenden Endpunkt hinzufügen können.

Verwenden von Endpunktrichtlinien zur Steuerung des Zugriffs mit VPC-Endpunkten

VPC-Endpunktrichtlinien werden für App Runner unterstützt. Standardmäßig ist der vollständige Zugriff auf App Runner über den Schnittstellenendpunkt zulässig. VPC-Endpunktrichtlinien können verwendet werden, um zu kontrollieren, welche AWS-Prinzipale auf den App Runner-Endpunkt zugreifen können. Alternativ können Sie den Endpunkt-Netzwerkschnittstellen eine Sicherheitsgruppe zuordnen, um den Datenverkehr zu App Runner über den Schnittstellenendpunkt zu steuern.

Integration mit dem Schnittstellenendpunkt

App Runner unterstützt AWS PrivateLink, bietet private Konnektivität zu App Runner und verhindert, dass der Datenverkehr über das Internet zugänglich ist. Damit Ihre Anwendung Anfragen an App Runner senden kann AWS PrivateLink, konfigurieren Sie einen VPC-Endpunkttyp, den sogenannten

Schnittstellenendpunkt. Weitere Informationen finden Sie im Handbuch unter Interface VPC endpoints (AWS PrivateLink).AWS PrivateLink

Konfiguration und Schwachstellenanalyse in App Runner

AWS und unsere Kunden sind gemeinsam dafür verantwortlich, ein hohes Maß an Sicherheit und Konformität der Softwarekomponenten zu erreichen. Weitere Informationen finden Sie im Modell der AWS gemeinsamen Verantwortung.

Patchen Sie Container-Images

Das Patchen des Container-Images liegt im Rahmen des gemeinsamen Sicherheitsmodells in der Verantwortung des Kunden. Der Image-Eigentümer ist dafür verantwortlich, das Container-Image zu aktualisieren und regelmäßig zu patchen. Wir empfehlen, einen Routineplan für die Überprüfung und Aktualisierung Ihrer Container-Images aufzustellen. Weitere Informationen zum Scannen Ihrer Images auf Sicherheitslücken finden Sie in der AWS App Runner-Dokumentation

Weitere Sicherheitsthemen zu App Runner finden Sie unterSicherheit.

Bewährte Sicherheitsmethoden für App Runner

AWS App Runner bietet mehrere Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Weitere Sicherheitsthemen von App Runner finden Sie unterSicherheit.

Bewährte Methoden für vorbeugende Sicherheitsmaßnahmen

Vorbeugende Sicherheitskontrollen versuchen, Vorfälle zu verhindern, bevor sie auftreten.

Implementieren des Zugriffs mit geringsten Berechtigungen

App Runner bietet AWS Identity and Access Management (IAM) verwaltete Richtlinien für IAM-Benutzer und die Zugriffsrolle. Diese verwalteten Richtlinien spezifizieren alle Berechtigungen, die für den korrekten Betrieb Ihres App Runner-Dienstes erforderlich sein könnten.

Ihre Anwendung benötigt möglicherweise nicht alle Berechtigungen in unseren verwalteten Richtlinien. Sie können sie anpassen und nur die Berechtigungen gewähren, die für Ihre Benutzer und Ihren App Runner-Dienst zur Ausführung ihrer Aufgaben erforderlich sind. Dies ist besonders relevant für Benutzerrichtlinien, in denen unterschiedliche Benutzerrollen möglicherweise unterschiedliche Berechtigungsanforderungen haben. Die Implementierung der geringstmöglichen Zugriffsrechte ist eine grundlegende Voraussetzung zum Reduzieren des Sicherheitsrisikos und der Auswirkungen, die aufgrund von Fehlern oder böswilligen Absichten entstehen könnten.

Ihre Images auf Schwachstellen scannen

Sie können die Amazon ECRs verwenden APIs , um Softwareschwachstellen in Ihren Container-Images zu identifizieren. Weitere Informationen finden Sie in der Amazon ECR-Dokumentation.

Bewährte Methoden für aufdeckende Sicherheitsmaßnahmen

Aufdeckende Sicherheitskontrollen identifizieren Sicherheitsverstöße, nachdem sie aufgetreten sind. Sie können Ihnen dabei helfen, potenzielle Sicherheitsbedrohungen oder -vorfälle zu erkennen.

Implementieren der Überwachung

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Sicherheit, Verfügbarkeit und Leistung Ihrer App Runner-Lösungen. AWS bietet verschiedene Tools und Dienste, mit denen Sie Ihre AWS Dienste überwachen können.

Es folgen einige Beispiele für zu überwachende Elemente:

- CloudWatch Amazon-Metriken für App Runner — Richten Sie Alarme für wichtige App Runner-Metriken und für die benutzerdefinierten Metriken Ihrer Anwendung ein. Details hierzu finden Sie unter Metriken (CloudWatch).
- AWS CloudTrail Einträge — Verfolge Aktionen, die sich auf die Verfügbarkeit auswirken könnten, wie `PauseService` oder`DeleteConnection`. Details hierzu finden Sie unter API-Aktionen (CloudTrail).

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.