aws

개발자 안내서

AWS App Runner

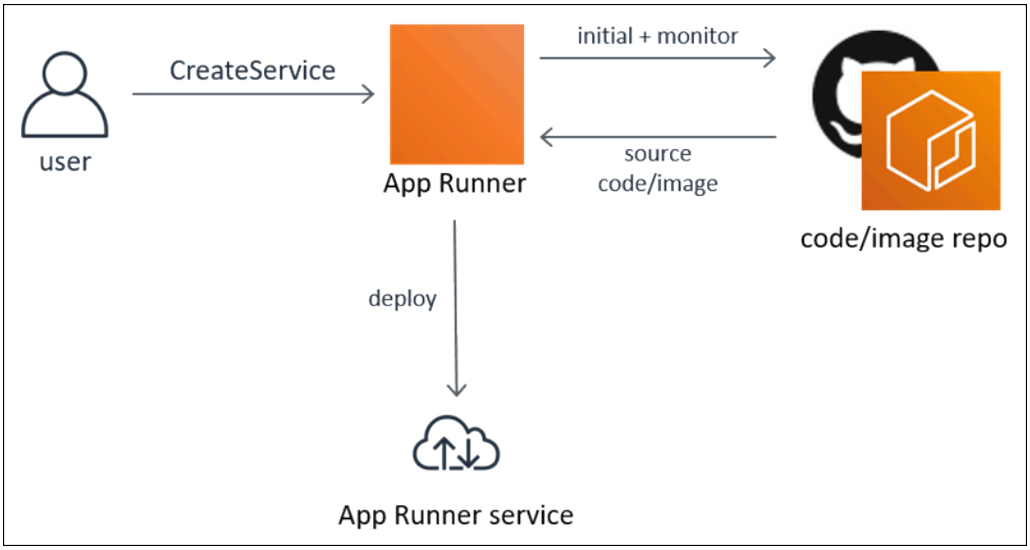
AWS App Runner: 개발자 안내서

Table of Contents

무엇입니까 AWS App Runner?

AWS App Runner 소스 코드 또는 컨테이너 이미지를 클라우드의 확장 가능하고 안전한 웹 애플리케이션으로 직접 배포할 수 있는 빠르고 간단하며 비용 효율적인 방법을 제공하는 AWS 서비스입니다. AWS 새로운 기술을 배우거나, 사용할 컴퓨팅 서비스를 결정하거나, AWS 리소스를 프로비저닝하고 구성하는 방법을 몰라도 됩니다.

App Runner는 코드 또는 이미지 리포지토리에 직접 연결됩니다. 완전 관리형 운영, 고성능, 확장성, 보안을 갖춘 자동 통합 및 전달 파이프라인을 제공합니다.



앱 러너는 누구를 위한 제품인가요?

개발자인 경우 App Runner를 사용하여 코드나 이미지 리포지토리의 새 버전을 배포하는 프로세스를 간소화할 수 있습니다.

운영팀의 경우 App Runner를 사용하면 커밋을 코드 리포지토리에 푸시하거나 새 컨테이너 이미지 버전을 이미지 리포지토리에 푸시할 때마다 자동으로 배포할 수 있습니다.

App Runner에 액세스하기

다음 인터페이스 중 하나를 사용하여 App Runner 서비스 배포를 정의하고 구성할 수 있습니다.

- App Runner 콘솔 - App Runner 서비스를 관리하기 위한 웹 인터페이스를 제공합니다.
- 앱 러너 API - 앱 러너 작업을 수행하기 위한 RESTful API를 제공합니다. 자세한 내용은 AWS App Runner API 참조를 참조하세요.

- AWS 명령줄 인터페이스 (AWS CLI) — Amazon VPC를 비롯한 다양한 AWS 서비스에 대한 명령을 제공하며 Windows, macOS 및 Linux에서 지원됩니다. 자세한 정보는 AWS Command Line Interface을 참조하세요.
- AWS SDK — 언어별 API를 제공하고 서명 계산, 요청 재시도 처리, 오류 처리와 같은 많은 연결 세부 정보를 관리합니다. 자세한 정보는 AWS SDK를 참조하세요.

앱 러너 요금

App Runner는 애플리케이션을 실행하는 비용 효율적인 방법을 제공합니다. App Runner 서비스가 소비하는 리소스에 대해서만 비용을 지불하면 됩니다. 요청 트래픽이 적으면 서비스 규모를 줄여 컴퓨팅 인스턴스 수를 줄일 수 있습니다. 프로비저닝된 인스턴스의 최소 및 최대 수, 인스턴스가 처리하는 최대 부하 등 확장성 설정을 제어할 수 있습니다.

App Runner 자동 확장에 대한 자세한 내용은 을 참조하십시오. the section called “Auto Scaling”

요금 정보는 AWS App Runner 요금을 참조하세요.

다음에 있는 것

다음 주제에서 App Runner를 시작하는 방법을 알아보세요.

- 설정— App Runner를 사용하기 위한 사전 필수 단계를 완료하세요.
- 시작하기— 첫 번째 애플리케이션을 App Runner에 배포합니다.

앱 러너 설정

신규 AWS 고객인 경우 사용을 시작하기 전에 이 페이지에 나열된 설정 사전 요구 사항을 완료하십시오. AWS App Runner

이러한 설정 절차에는 AWS Identity and Access Management (IAM) 서비스를 사용합니다. IAM에 대한 자세한 내용은 다음 참조 자료를 참조하세요.

- AWS Identity and Access Management (IAM)
- IAM 사용 설명서

가입해 보세요 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

프로그래밍 방식 액세스 권한 부여

사용자가 AWS 외부 사용자와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		• AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

다음에 있는 것

필수 단계를 완료했습니다. App Runner에 첫 번째 애플리케이션을 배포하려면 을 참조하십시오. 시작하기

앱 러너 시작하기

AWS App Runner 기존 컨테이너 이미지 또는 소스 코드를 에서 실행 중인 웹 AWS 서비스로 직접 전환하는 빠르고 간단하며 비용 효율적인 방법을 제공하는 서비스입니다. AWS 클라우드

이 자습서에서는 App Runner 서비스에 애플리케이션을 배포하는 AWS App Runner 데 사용할 수 있는 방법을 다룹니다. 소스 코드 및 배포, 서비스 빌드, 서비스 런타임 구성을 안내합니다. 또한 코드 버전을 배포하고, 구성을 변경하고, 로그를 보는 방법도 보여줍니다. 마지막으로 자습서에서는 자습서의 절차를 따르면서 만든 리소스를 정리하는 방법을 보여줍니다.

주제

필수 조건

자습서를 시작하기 전에 다음 작업을 수행하세요.

1. 의 설정 단계를 완료하십시오설정.
2. 리포지토리를 사용할지 아니면 Bitbucket GitHub 리포지토리를 사용할지 결정하세요.
   - Bitbucket을 사용하려면 Bitbucket 계정이 아직 없다면 먼저 Bitbucket 계정을 만드세요. Bitbucket을 처음 사용하는 경우 Bitbucket 클라우드 설명서에서 Bitbucket 시작하기를 참조하십시오.
   - 아직 계정이 없다면 GitHub계정을 만들어 사용하세요 GitHub. 처음 사용하는 경우 GitHub GitHub 문서에서 시작하기를 참조하세요. GitHub

> **Note**
>
> 계정에서 여러 저장소 제공자에 대한 연결을 만들 수 있습니다. 따라서 Bitbucket GitHub 리포지토리와 Bitbucket 리포지토리 모두에서 배포하는 방법을 살펴보려면 이 절차를 반

복하면 됩니다. 다음 번에는 새 App Runner 서비스를 만들고 다른 저장소 공급자를 위한 새 계정 연결을 만드십시오.

3. 리포지토리 제공자 계정에서 리포지토리를 생성하세요. 이 자습서에서는 리포지토리 이름을 사용합니다`python-hello`. 다음 예제에 지정된 이름과 내용을 사용하여 리포지토리의 루트 디렉터리에 파일을 생성합니다.

`python-hello`예제 리포지토리의 파일

Example requirements.txt

```
pyramid==2.0
```

Example server.py

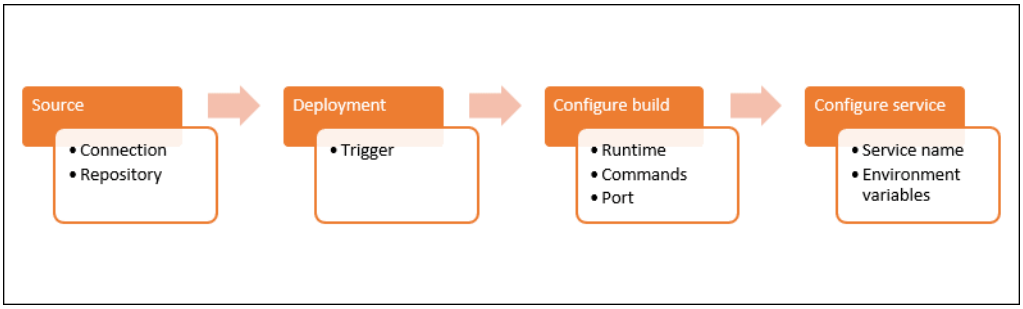
```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

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()
```

1단계: 앱 러너 서비스 만들기

이 단계에서는 GitHub Bitbucket이나 Bitbucket에 만든 예제 소스 코드 리포지토리를 기반으로 App Runner 서비스를 만듭니다. the section called “필수 조건” 이 예제에는 간단한 Python 웹 사이트가 포함되어 있습니다. 서비스를 생성하기 위해 수행하는 주요 단계는 다음과 같습니다.

1. 소스 코드를 구성하세요.
2. 소스 배포를 구성합니다.
3. 애플리케이션 빌드를 구성합니다.
4. 서비스를 구성하세요.
5. 검토 및 확인.

다음 다이어그램은 App Runner 서비스를 만드는 단계를 요약한 것입니다.

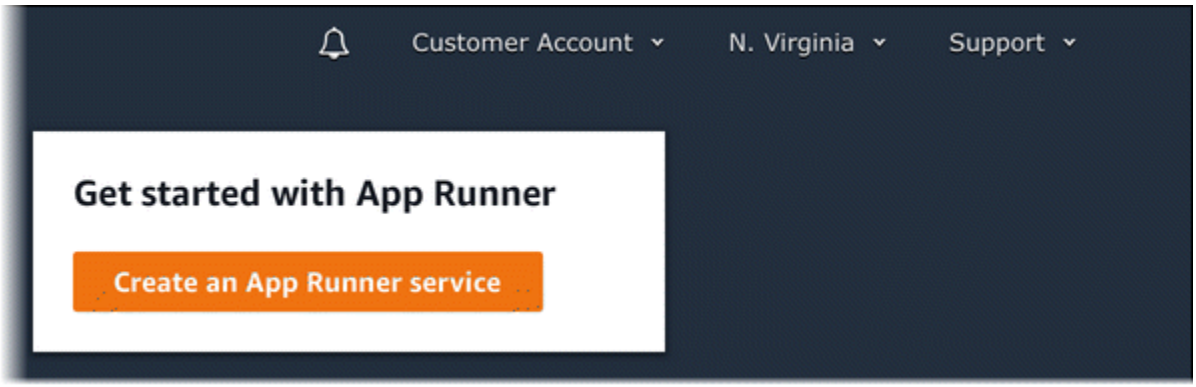


소스 코드 리포지토리를 기반으로 App Runner 서비스를 만들려면

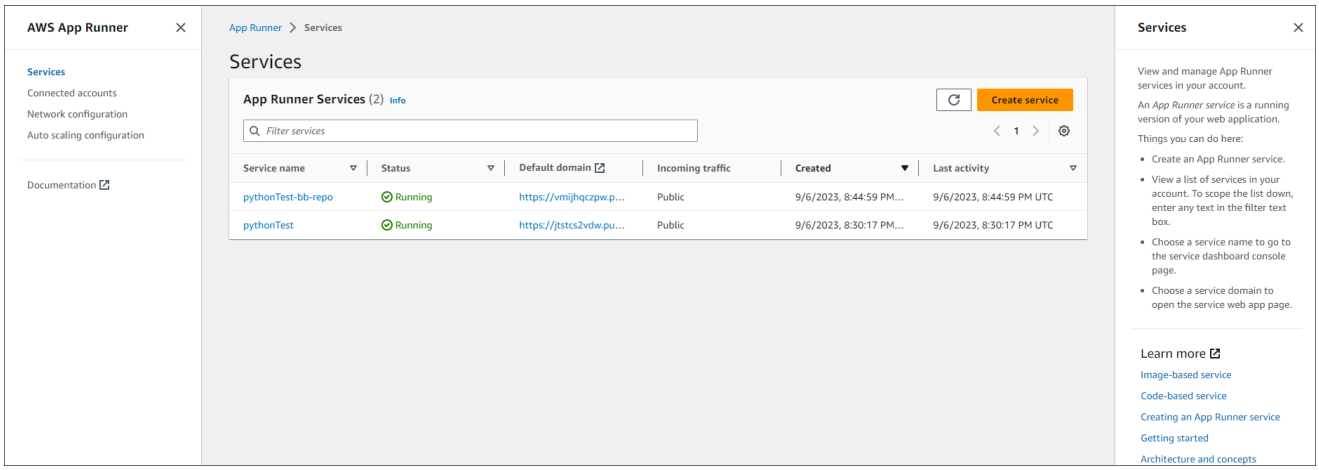
1. 소스 코드를 구성하세요.

   a. App Runner 콘솔을 열고 지역 목록에서 원하는 항목을 선택합니다. AWS 리전

   b. 에 아직 App Runner 서비스가 AWS 계정 없는 경우 콘솔 홈페이지가 표시됩니다. 앱 러너 서비스 만들기를 선택합니다.



에 기존 서비스가 AWS 계정 있는 경우 서비스 목록이 있는 서비스 페이지가 표시됩니다. 서비스 생성을 선택합니다.



c. 소스 및 배포 페이지의 소스 섹션에서 리포지토리 유형으로 소스 코드 리포지토리를 선택합니다.

d. 제공자 유형을 선택합니다. 둘 중 하나를 GitHub선택하거나 Bitbucket을 선택합니다.

e. 다음으로 새로 추가를 선택합니다. 메시지가 표시되면 사용자 GitHub 또는 Bitbucket 자격 증명을 제공하십시오.

f. 이전에 선택한 공급자 유형에 따라 다음 단계 세트를 선택합니다.

> **Note**
>
> GitHub 계정에 사용할 GitHub AWS 커넥터를 설치하는 다음 단계는 일회성 단계입니다. 연결을 재사용하여 이 계정의 리포지토리를 기반으로 여러 App Runner 서비스를 생성할 수 있습니다. 기존 연결이 있는 경우 연결을 선택하고 저장소 선택으로 건너뛰십시오.
>
> Bitbucket 계정의 AWS 커넥터에도 동일하게 적용됩니다. 앱 러너 서비스의 소스 코드 리포지토리로 GitHub Bitbucket과 Bitbucket을 모두 사용하는 경우 공급자당 AWS Connector를 하나씩 설치해야 합니다. 그런 다음 각 커넥터를 재사용하여 더 많은 App Runner 서비스를 생성할 수 있습니다.

- 의 GitHub경우 다음 단계를 따르세요.

  i. 다음 화면에서 연결 이름을 입력합니다.

  ii. App GitHub Runner를 처음 사용하는 경우 다른 앱 설치를 선택하십시오.

iii. AWS 커넥터 대상 GitHub 대화 상자에서 메시지가 표시되면 GitHub 계정 이름을 선택합니다.

iv. AWS 커넥터를 승인하라는 메시지가 표시되면 [AWS 연결 권한 부여] 를 GitHub 선택합니다.

v. AWS 커넥터 설치 대상 GitHub 대화 상자에서 설치를 선택합니다.

계정 이름은 선택한 GitHub 계정/조직으로 표시됩니다. 이제 계정에서 저장소를 선택할 수 있습니다.

vi. 리포지토리의 경우, 생성한 예제 리포지토리를 선택합니다`python-hello`. Branch의 경우 리포지토리의 기본 브랜치 이름 (예: main) 을 선택합니다.

vii. 소스 디렉터리는 기본값으로 그대로 두십시오. 디렉터리의 기본값은 리포지토리 루트입니다. 이전 사전 요구 사항 단계에서 리포지토리 루트 디렉터리에 소스 코드를 저장했습니다.

- Bitbucket의 경우 다음 단계를 따르십시오.

i. 다음 화면에서 연결 이름을 입력합니다.

ii. 앱 러너와 함께 Bitbucket을 처음 사용하는 경우 다른 앱 설치를 선택하십시오.

iii. 액세스AWS CodeStar 요청 대화 상자에서 작업 공간을 선택하고 Bitbucket 통합을 AWS CodeStar 위한 액세스 권한을 부여할 수 있습니다. 작업 공간을 선택한 다음 액세스 허용을 선택합니다.

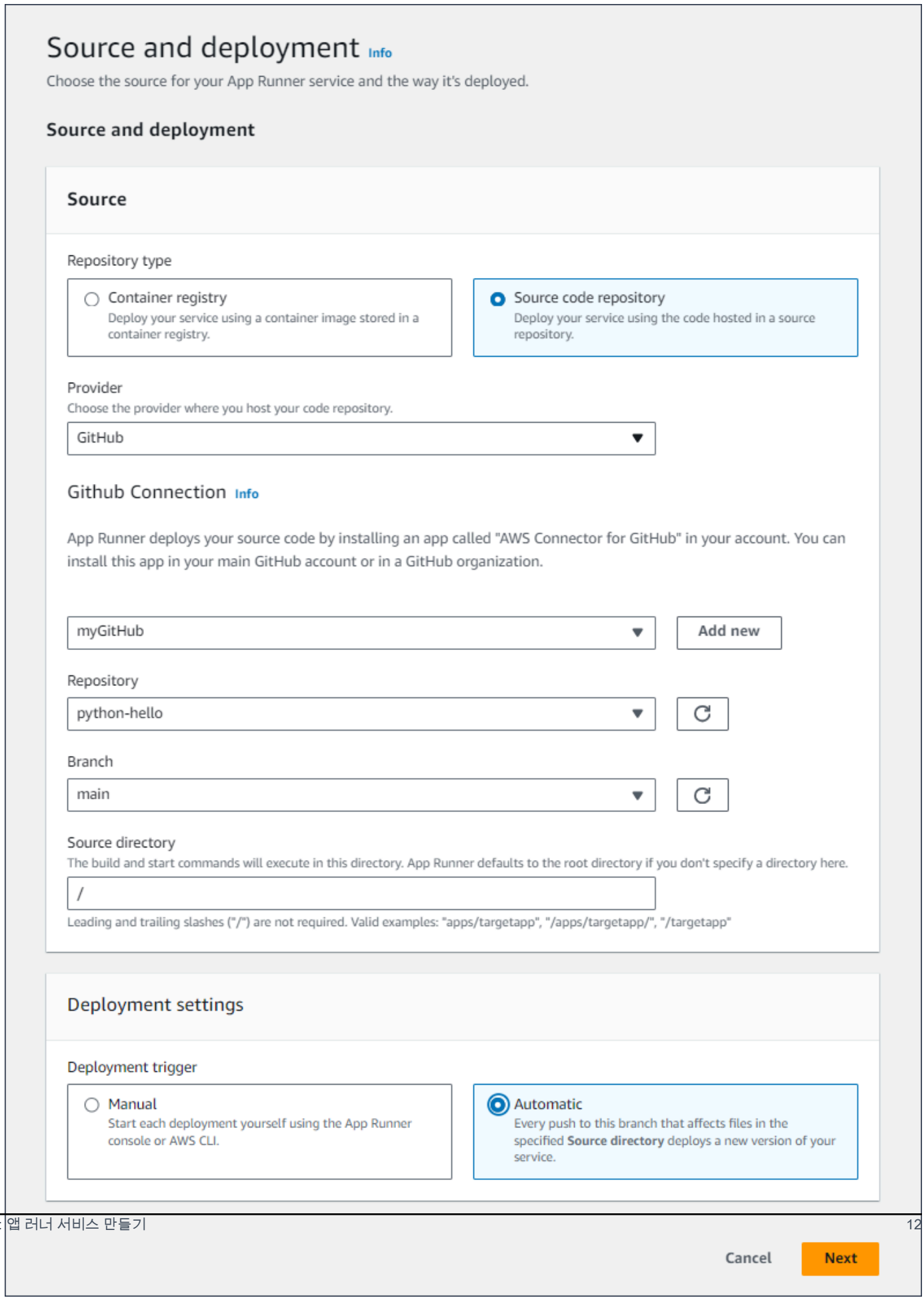
iv. 이제 AWS 콘솔로 리디렉션됩니다. Bitbucket 응용 프로그램이 올바른 Bitbucket 작업 영역으로 설정되었는지 확인하고 다음을 선택합니다.

v. 리포지토리의 경우, 생성한 예제 리포지토리를 선택합니다. `python-hello` Branch의 경우 리포지토리의 기본 브랜치 이름 (예: main) 을 선택합니다.

vi. 소스 디렉터리는 기본값으로 그대로 두십시오. 디렉터리의 기본값은 리포지토리 루트입니다. 이전 사전 요구 사항 단계에서 리포지토리 루트 디렉터리에 소스 코드를 저장했습니다.

2. 배포 구성: 배포 설정 섹션에서 자동을 선택한 후 다음을 선택합니다.

> **Note**
>
> 자동 배포를 사용하면 리포지토리 소스 디렉터리에 새로 커밋할 때마다 새 버전의 서비스가 자동으로 배포됩니다.


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. 애플리케이션 빌드를 구성합니다.

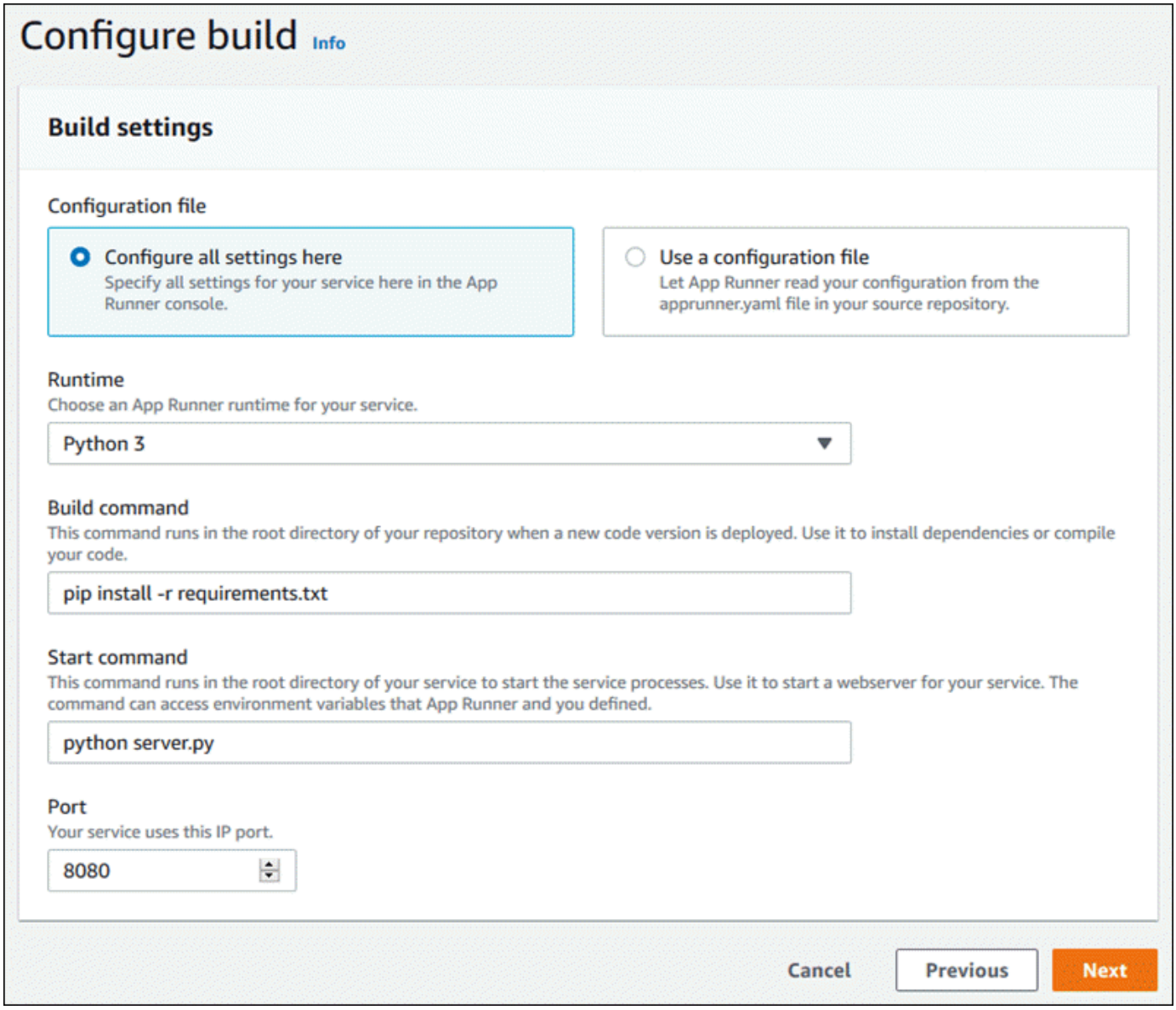
   a. 빌드 구성 페이지의 구성 파일에서 여기에서 모든 설정 구성을 선택합니다.

   b. 다음 빌드 설정을 제공하세요.

      - 런타임 — Python 3을 선택합니다.
      - 빌드 명령 — 엔터**`pip install -r requirements.txt`**.
      - 시작 명령 — 입력 **`python server.py`**
      - 포트 — 엔터**`8080`**.

   c. 다음을 선택합니다.

> **Note**
>
> Python 3 런타임은 기본 Python 3 이미지와 예제 Python 코드를 사용하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지의 컨테이너 인스턴스를 실행하는 서비스를 시작합니다.



4. 서비스를 구성하세요.

   a. 서비스 구성 페이지의 서비스 설정 섹션에서 서비스 이름을 입력합니다.

   b. 환경 변수에서 환경 변수 추가를 선택합니다. 환경 변수에 다음 값을 입력합니다.

      - 소스 — 일반 텍스트를 선택합니다.
      - 환경 변수 이름 — **NAME**
      - 환경 변수 값 — 모든 이름 (예: 이름)

> **Note**
>
> 예제 애플리케이션은 이 환경 변수에 설정된 이름을 읽고 해당 웹 페이지에 이름을 표시합니다.

c. 다음을 선택합니다.

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.

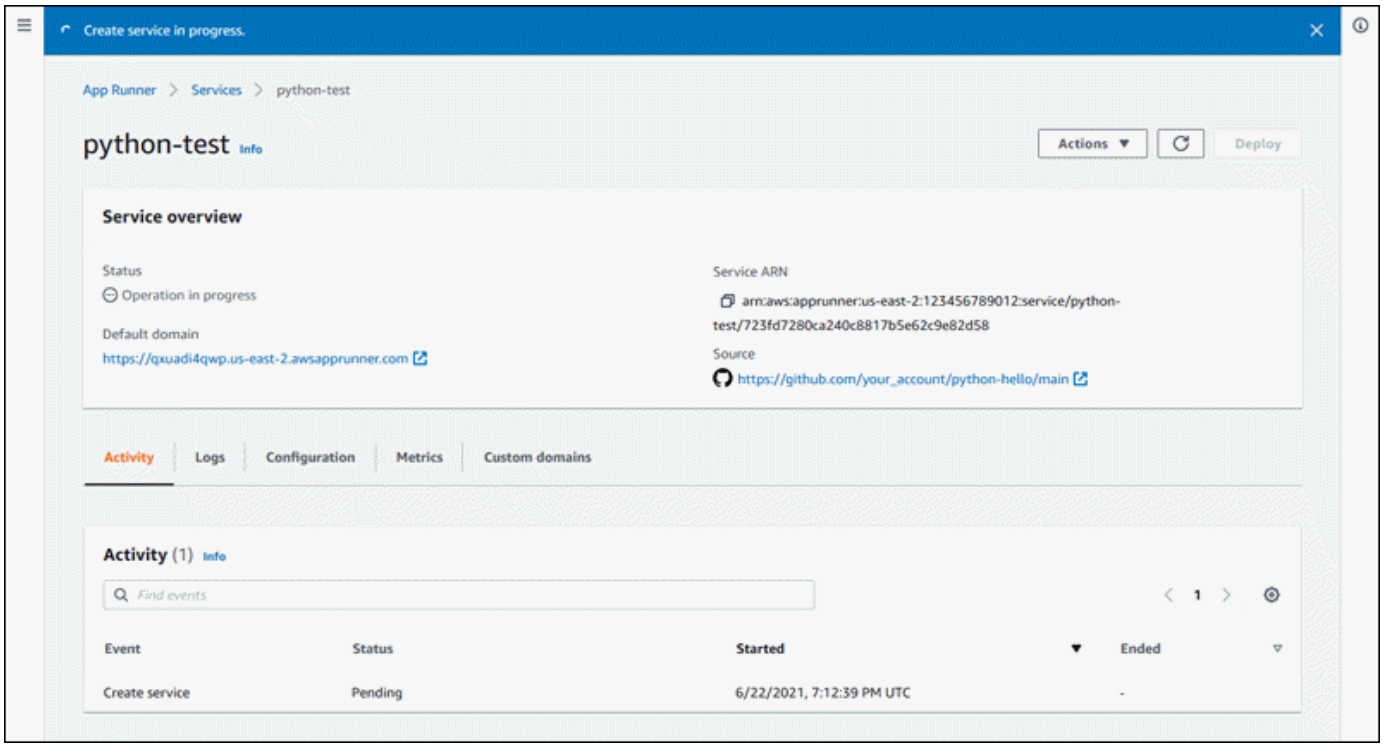
▼ Tags Info

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

Tags — *optional*

5. 검토 및 생성 페이지에서 입력한 모든 세부 정보를 확인한 다음 생성 및 배포를 선택합니다.

   서비스가 성공적으로 생성되면 콘솔에 새 서비스의 서비스 개요와 함께 서비스 대시보드가 표시됩니다.



6. 서비스가 실행 중인지 확인하세요.

   a. 서비스 대시보드 페이지에서 서비스 상태가 Running (실행 중) 이 될 때까지 기다립니다.

   b. 기본 도메인 값을 선택합니다. 기본 도메인 값은 서비스 웹사이트의 URL입니다.

> **Note**
>
> 앱 러너 애플리케이션의 보안을 강화하기 위해*.awsapprunner.com 도메인은 공개 접미사 목록 (PSL) 에 등록되어 있습니다. 보안 강화를 위해 App Runner 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

```
# #### ### ## #### #####. #####, ### #####!
```

2단계: 서비스 코드 변경

이 단계에서는 리포지토리 소스 디렉터리의 코드를 변경합니다. App Runner CI/CD 기능은 변경 내용을 자동으로 빌드하여 서비스에 배포합니다.

서비스 코드를 변경하려면

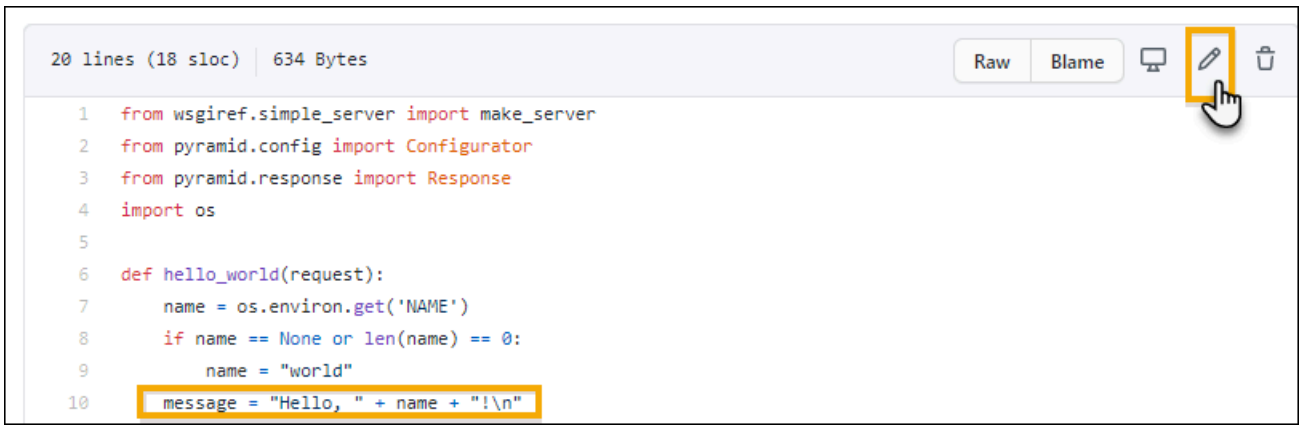
1. 예제 저장소로 이동합니다.
2. 이름이 지정된 파일을 `server.py` 편집합니다.
3. 변수에 `message` 지정된 표현식에서 텍스트를 `Hello` 로 변경합니다`Good morning`.
4. 변경 내용을 저장한 후 저장소에 커밋합니다.
5. 다음 단계는 GitHub 리포지토리의 서비스 코드 변경을 보여줍니다.

   a. 예제 GitHub 리포지토리로 이동합니다.

   b. 파일 이름을 `server.py` 선택하여 해당 파일을 탐색합니다.

   c. 이 파일 편집 (연필 아이콘) 을 선택합니다.

   d. 변수에 `message` 지정된 표현식에서 텍스트를 `Hello` 로 변경합니다`Good morning`.



e. 변경 사항 커밋을 선택합니다.

6. 새 커밋이 App Runner 서비스에 배포되기 시작합니다. 서비스 대시보드 페이지에서 서비스 상태가 작업 진행 중으로 변경됩니다.

   배포가 종료될 때까지 기다리세요. 서비스 대시보드 페이지에서 서비스 상태가 다시 Running (실행 중) 으로 변경되어야 합니다.

7. 배포가 성공했는지 확인하세요. 서비스의 웹 페이지가 표시되는 브라우저 탭을 새로 고치세요.

   이제 페이지에 다음과 같은 수정된 메시지가 표시됩니다. 안녕하세요, ***### ##!***

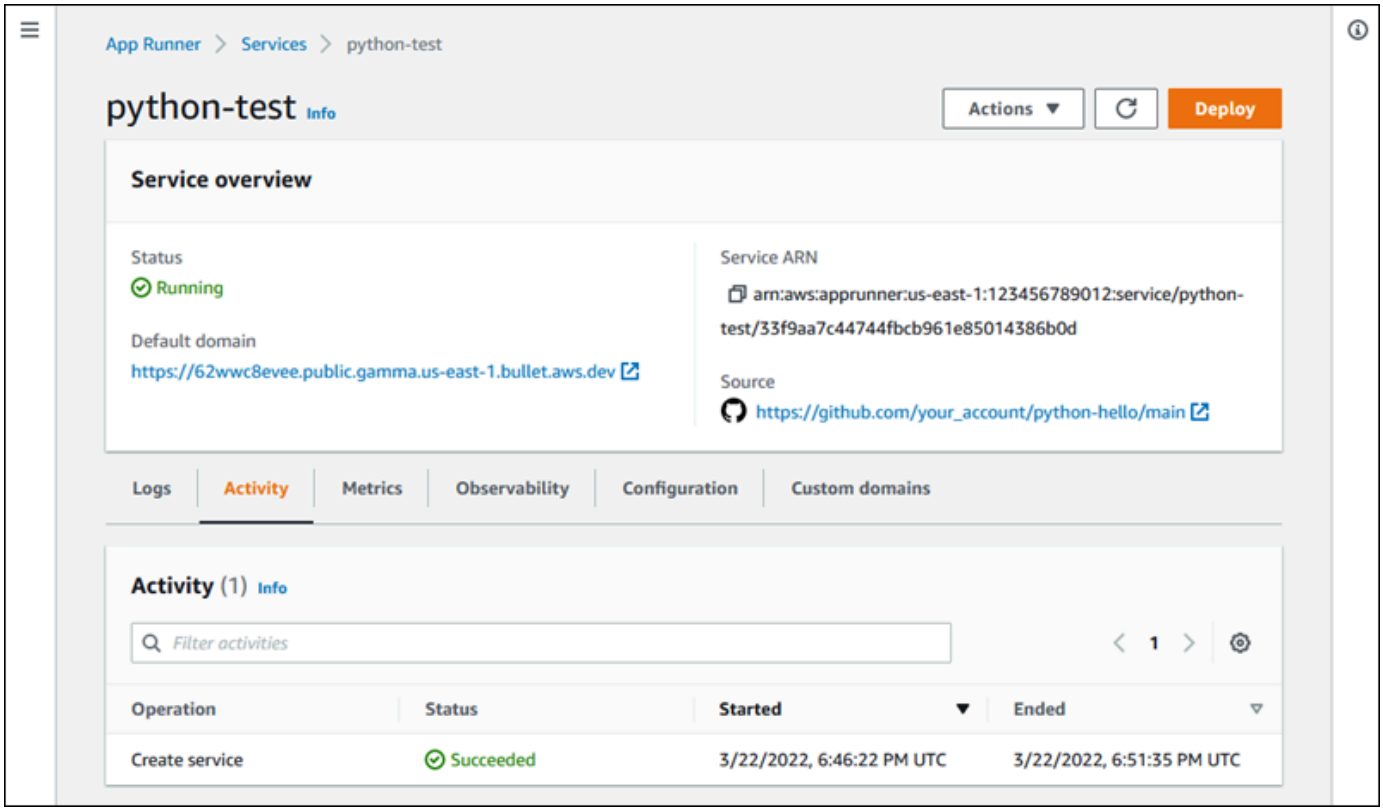
3단계: 구성 변경

이 단계에서는 서비스 구성 변경을 시연하기 위해 **NAME** 환경 변수 값을 변경합니다.

환경 변수 값을 변경하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

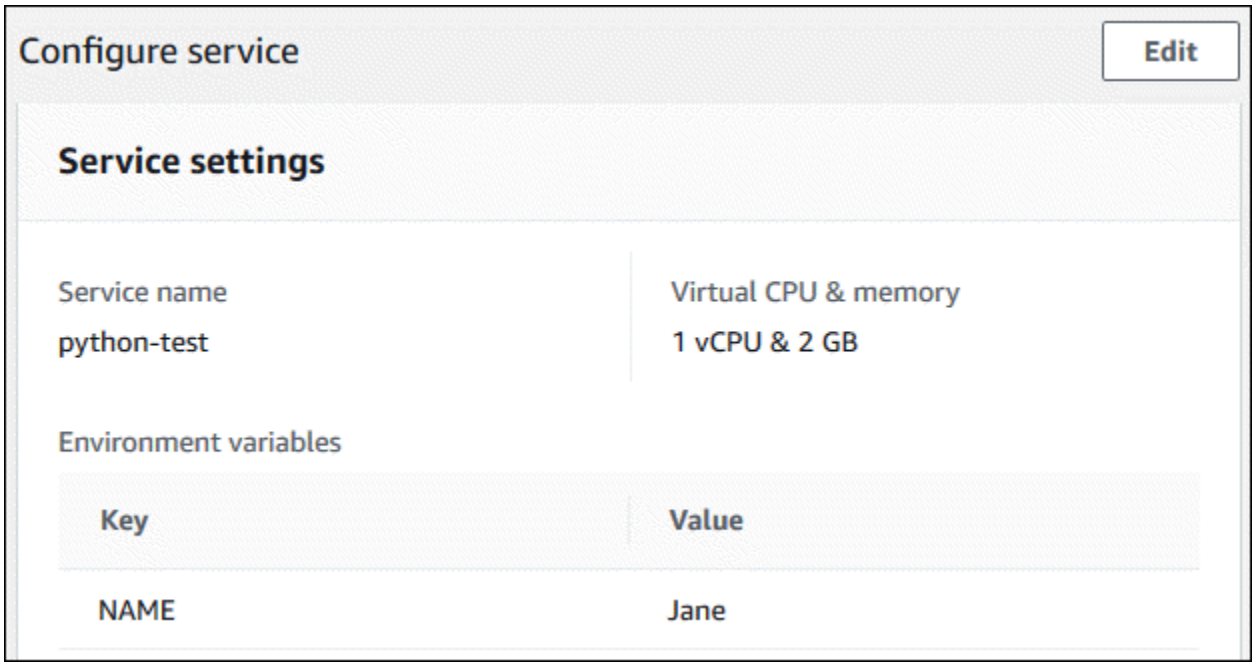
   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 구성 탭을 선택합니다.

   콘솔에는 서비스 구성 설정이 여러 섹션으로 표시됩니다.

4. 서비스 구성 섹션에서 편집을 선택합니다.



5. 키가 있는 환경 변수의 **NAME** 경우 값을 다른 이름으로 변경합니다.

6. [Apply changes]를 선택합니다.

   App Runner가 업데이트 프로세스를 시작합니다. 서비스 대시보드 페이지에서 서비스 상태가 작업 진행 중으로 변경됩니다.

7. 업데이트가 종료될 때까지 기다리세요. 서비스 대시보드 페이지에서 서비스 상태가 다시 Running (실행 중) 으로 변경되어야 합니다.

8. 업데이트가 성공했는지 확인하세요. 서비스의 웹 페이지가 표시되는 브라우저 탭을 새로 고침하세요.

   이제 페이지에 수정된 이름이 표시됩니다. 안녕하세요, **# ##**!

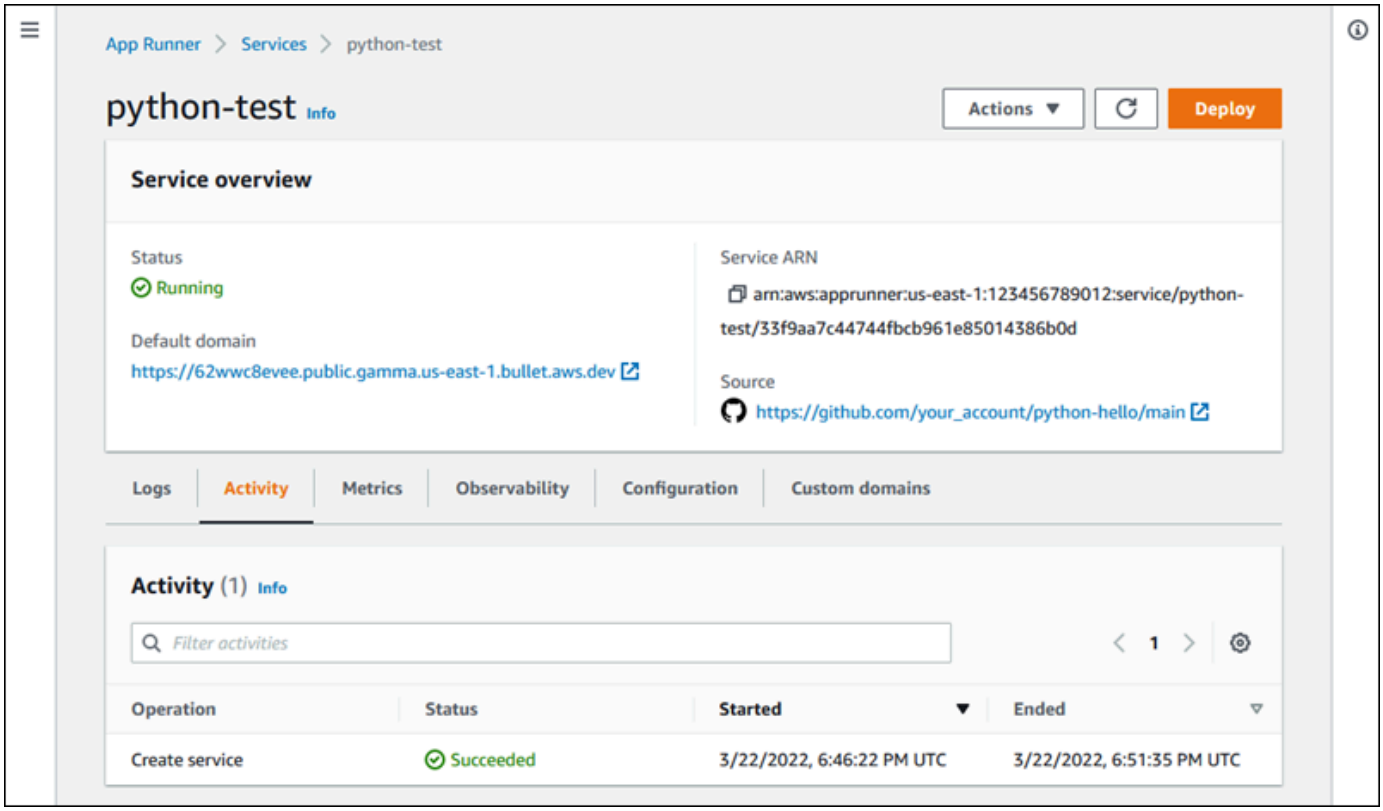
4단계: 서비스 로그 보기

이 단계에서는 App Runner 콘솔을 사용하여 App Runner 서비스에 대한 로그를 확인합니다. App Runner는 CloudWatch 로그를 Amazon Logs (CloudWatch Logs) 로 스트리밍하여 서비스 대시보드에 표시합니다. App Runner 로그에 대한 자세한 내용은 을 참조하십시오. the section called “로그 (로그) CloudWatch ”

서비스 로그를 보려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

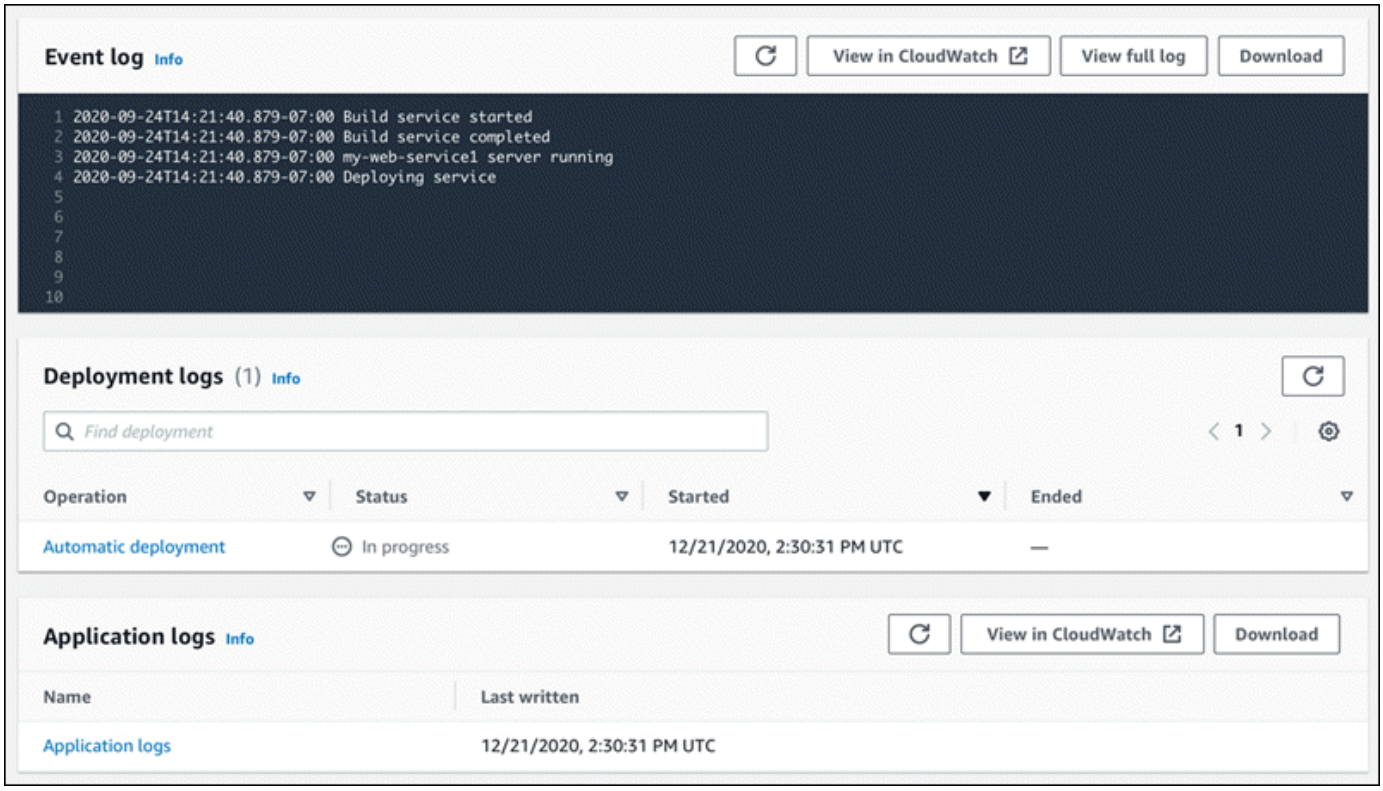
   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 로그 탭을 선택합니다.

   콘솔에는 여러 섹션에 몇 가지 유형의 로그가 표시됩니다.

   - 이벤트 로그 - App Runner 서비스의 라이프사이클 내 활동입니다. 콘솔에는 최신 이벤트가 표시됩니다.
   - 배포 로그 - App Runner 서비스에 대한 소스 리포지토리 배포입니다. 콘솔에는 각 배포에 대한 별도의 로그 스트림이 표시됩니다.
   - 애플리케이션 로그 - App Runner 서비스에 배포된 웹 애플리케이션의 출력입니다. 콘솔은 실행 중인 모든 인스턴스의 출력을 단일 로그 스트림으로 결합합니다.



4. 특정 배포를 찾으려면 검색어를 입력하여 배포 로그 목록의 범위를 좁히십시오. 표에 나타나는 모든 값을 검색할 수 있습니다.
5. 로그 내용을 보려면 전체 로그 보기 (이벤트 로그) 또는 로그 스트림 이름 (배포 및 애플리케이션 로그) 을 선택합니다.
6. [Download] 를 선택하여 로그를 다운로드합니다. 배포 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.
7. View CloudWatch in을 선택하여 CloudWatch 콘솔을 열고 콘솔의 모든 기능을 사용하여 App Runner 서비스 로그를 탐색하세요. 배포 로그 스트림의 경우 먼저 로그 스트림을 선택하십시오.

> **Note**
>
> CloudWatch 콘솔은 결합된 애플리케이션 로그 대신 특정 인스턴스의 애플리케이션 로그를 보려는 경우에 특히 유용합니다.

5단계: 정리

이제 App Runner 서비스를 만들고, 로그를 보고, 일부 변경하는 방법을 배웠습니다. 이 단계에서는 서비스를 삭제하여 더 이상 필요하지 않은 리소스를 제거합니다.

서비스를 삭제하려면

1. 서비스 대시보드 페이지에서 작업을 선택한 다음 서비스 삭제를 선택합니다.
2. 확인 대화 상자에서 요청된 텍스트를 입력한 다음 삭제를 선택합니다.

   결과: 콘솔이 서비스 페이지로 이동합니다. 방금 삭제한 서비스는 삭제 상태로 표시됩니다. 잠시 후 목록에서 사라집니다.

이 자습서의 일부로 만든 GitHub 및 Bitbucket 연결을 삭제하는 것도 고려해 보십시오. 자세한 정보는 the section called “연결”을 참조하세요.

다음에 있는 것

첫 번째 App Runner 서비스를 배포했으니 이제 다음 항목에서 자세히 알아보세요.

- 아키텍처 및 개념— App Runner와 관련된 아키텍처, 주요 개념 및 AWS 리소스
- 이미지 기반 서비스 및 코드 기반 서비스 — App Runner가 배포할 수 있는 두 가지 유형의 애플리케이션 소스.
- 앱 러너용 개발— App Runner에 배포하기 위해 애플리케이션 코드를 개발하거나 마이그레이션할 때 알아야 할 사항.
- 앱 러너 콘솔— App Runner 콘솔을 사용하여 서비스를 관리하고 모니터링할 수 있습니다.
- 서비스 관리— App Runner 서비스의 수명 주기를 관리합니다.
- 관찰성— 지표를 모니터링하고, 로그를 읽고, 이벤트를 처리하고, 서비스 작업 호출을 추적하고, HTTP 호출과 같은 애플리케이션 이벤트를 추적하여 App Runner 서비스 운영에 대한 가시성을 확보할 수 있습니다.
- 앱 러너 구성 파일— App Runner 서비스의 빌드 및 런타임 동작에 대한 옵션을 지정하는 구성 기반 방법입니다.
- 앱 러너 API— App Runner API (애플리케이션 프로그래밍 인터페이스) 를 사용하여 App Runner 리소스를 만들고, 읽고, 업데이트하고, 삭제합니다.
- 보안— App Runner AWS 및 기타 서비스를 사용하는 동안 클라우드 보안을 보장하는 다양한 방법.

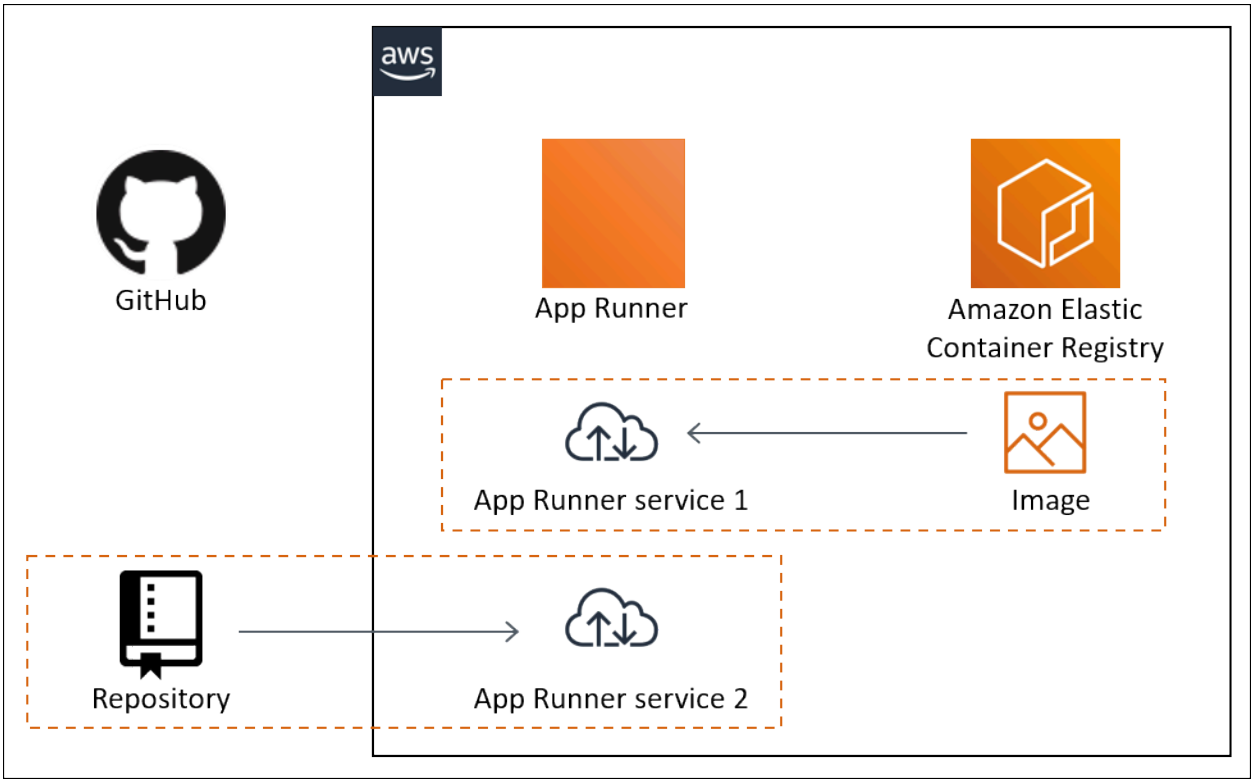
앱 러너 아키텍처 및 개념

AWS App Runner 저장소에서 소스 코드나 소스 이미지를 가져와서 에서 실행 중인 웹 서비스를 만들고 유지 관리합니다 AWS 클라우드. 일반적으로 서비스를 만들려면 App Runner 작업 CreateService, 을 하나만 호출하면 됩니다.

소스 이미지 리포지토리를 사용하면 App Runner가 웹 서비스를 실행하기 위해 배포할 수 있는 ready-to-use 컨테이너 이미지를 제공합니다. 소스 코드 리포지토리를 사용하여 웹 서비스를 빌드하고 실행하기 위한 코드와 지침을 제공하고 특정 런타임 환경을 대상으로 합니다. App Runner는 여러 프로그래밍 플랫폼을 지원하며, 각 플랫폼에는 플랫폼 메이저 버전에 대한 하나 이상의 관리형 런타임이 있습니다.

현재 App Runner는 Bitbucket 또는 GitHub리포지토리에서 소스 코드를 검색하거나, 사용자의 Amazon Elastic Container Registry (Amazon ECR) 에서 소스 이미지를 검색할 수 있습니다. AWS 계정

다음 다이어그램은 App Runner 서비스 아키텍처의 개요를 보여줍니다. 다이어그램에는 두 가지 예제 서비스가 있습니다. 하나는 Amazon ECR에서 GitHub 소스 코드를 배포하고 다른 하나는 Amazon ECR에서 소스 이미지를 배포합니다. Bitbucket 리포지토리에도 동일한 흐름이 적용됩니다.



앱 러너 개념

다음은 App Runner에서 실행되는 웹 서비스와 관련된 주요 개념입니다.

- App Runner 서비스 — App Runner가 소스 코드 리포지토리 또는 컨테이너 이미지를 기반으로 애플리케이션을 배포하고 관리하는 데 사용하는 AWS 리소스입니다. App Runner 서비스는 애플리케이션의 실행 버전입니다. 서비스 생성에 대한 자세한 내용은 을 참조하십시오the section called “생성”.
- 소스 유형 - App Runner 서비스를 배포하기 위해 제공하는 소스 리포지토리의 유형 (소스 코드 또는 소스 이미지).
- 리포지토리 제공자 — 애플리케이션 소스 (예: Bitbucket 또는 Amazon ECR) 를 포함하는 리포지토리 서비스입니다. GitHub
- 앱 러너 연결 — App Runner가 리포지토리 공급자 계정 (예: 계정 또는 조직) 에 액세스할 수 있게 해주는 AWS 리소스입니다. GitHub 연결에 대한 자세한 내용은 the section called “연결” 단원을 참조하십시오.
- 런타임 - 소스 코드 리포지토리를 배포하기 위한 기본 이미지입니다. App Runner는 다양한 프로그래밍 플랫폼 및 버전에 맞는 다양한 관리형 런타임을 제공합니다. 자세한 정보는 코드 기반 서비스을 참조하세요.
- 배포 — 소스 리포지토리의 버전 (코드 또는 이미지) 을 App Runner 서비스에 적용하는 작업입니다. 서비스에 대한 첫 번째 배포는 서비스 생성의 일부로 이루어집니다. 다음 두 가지 방법 중 하나로 나중에 배포할 수 있습니다.
  - 자동 배포 — CI/CD 기능. 리포지토리에 나타나는 대로 애플리케이션의 각 버전을 자동으로 빌드 (소스 코드용) 하여 배포하도록 App Runner 서비스를 구성할 수 있습니다. 소스 코드 리포지토리의 새 커밋이나 소스 이미지 리포지토리의 새 이미지 버전일 수 있습니다.
  - 수동 배포 — 사용자가 명시적으로 시작하는 App Runner 서비스에 대한 배포입니다.
- 커스텀 도메인 — App Runner 서비스에 연결하는 도메인입니다. 웹 애플리케이션 사용자는 기본 App Runner 하위 도메인 대신 이 도메인을 사용하여 웹 서비스에 액세스할 수 있습니다. 자세한 정보는 the section called “사용자 지정 도메인 이름”을 참조하세요.

> **Note**
>
> 앱 러너 애플리케이션의 보안을 강화하기 위해*.awsapprunner.com 도메인은 공개 접미사 목록 (PSL) 에 등록되어 있습니다. 보안 강화를 위해 App Runner 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을

보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

- 유지 관리 — App Runner가 App Runner 서비스를 실행하는 인프라에서 가끔 수행하는 활동입니다. 유지 관리가 진행 중이면 몇 분 동안 서비스 상태가 일시적으로 `OPERATION_IN_PROGRESS` (콘솔에서 작업 진행 중) 으로 변경됩니다. 이 기간 동안에는 서비스에 대한 작업 (예: 배포, 구성 업데이트, 일시 중지/재개 또는 삭제) 이 차단됩니다. 몇 분 후 서비스 상태가 로 돌아오면 작업을 다시 시도하세요. `RUNNING`

  > **Note**
  >
  > 작업이 실패한다고 해서 App Runner 서비스가 다운된 것은 아닙니다. 애플리케이션이 활성 상태이고 요청을 계속 처리합니다. 서비스에 다운타임이 발생할 가능성은 거의 없습니다.

  특히 App Runner는 서비스를 호스팅하는 기본 하드웨어에서 문제가 발견되면 서비스를 마이그레이션합니다. 서비스 다운타임을 방지하기 위해 App Runner는 서비스를 새로운 인스턴스 세트에 배포하고 트래픽을 인스턴스 세트로 이동합니다 (청록색 배포). 때때로 요금이 일시적으로 약간 인상될 수 있습니다.

앱 러너 지원 구성

App Runner 서비스를 구성할 때 서비스에 할당할 가상 CPU 및 메모리 구성을 지정합니다. 선택한 컴퓨팅 구성을 기준으로 비용을 지불합니다. 요금에 대한 자세한 내용은 AWS Resource Groups 요금을 참조하십시오.

다음 표에는 App Runner가 지원하는 vCPU 및 메모리 구성에 대한 정보가 나와 있습니다.

CPU	메모리
0.25 vCPU	0.5GB
0.25 vCPU	1GB
0.5 vCPU	1GB
1 vCPU	2GB

CPU	메모리
1 vCPU	3GB
1 vCPU	4GB
2 vCPU	4GB
2 vCPU	6 기가바이트
4 vCPU	8GB
4 vCPU	10GB
4 vCPU	12기가바이트

앱 러너 리소스

App Runner를 사용하면 앱 러너에서 몇 가지 유형의 리소스를 만들고 관리할 수 있습니다. AWS 계정 이러한 리소스는 코드에 액세스하고 서비스를 관리하는 데 사용됩니다.

다음 표에는 이러한 리소스의 개요가 나와 있습니다.

리소스 이름	설명
Service	실행 중인 애플리케이션 버전을 나타냅니다. 이 가이드의 나머지 부분에서는 대부분 서비스 유형, 관리, 구성 및 모니터링에 대해 설명합니다. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:service/` *`service-name`* `[/`*`service-id`* `]`
Connection	App Runner 서비스에 타사 제공업체에 저장된 비공개 리포지토리에 대한 액세스를 제공합니다. 여러 서비스에서 공유하기 위한 별도의 리소스로 존재합니다. 연결에 대한 자세한 내용은 the section called “연결” 단원을 참조하십시오. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:connection/` *`connection-name`* `[/`*`connection-id`* `]`

리소스 이름	설명
AutoScalingConfiguration	App Runner 서비스에 애플리케이션의 자동 크기 조정을 제어하는 설정을 제공합니다. 여러 서비스에서 공유하기 위한 별도의 리소스로 존재합니다. 자동 스케일링에 대한 자세한 내용은 the section called “Auto Scaling” 섹션을 참조하세요. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:autoscalingconfiguration/` *`config-name`* `[/`*`config-revision`* `[/`*`config-id`* `]]`
ObservabilityConfiguration	App Runner 서비스를 위한 추가 애플리케이션 옵저버빌리티 기능을 구성합니다. 여러 서비스에서 공유하기 위한 별도의 리소스로 존재합니다. 옵저버빌리티 구성에 대한 자세한 내용은 을 참조하십시오the section called “옵저버빌리티 구성”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:observabilityconfiguration/` *`config-name`* `[/`*`config-revision`* `[/`*`config-id`* `]]`
VpcConnector	앱 러너 서비스의 VPC 설정을 구성합니다. 여러 서비스에서 공유하기 위한 별도의 리소스로 존재합니다. VPC 기능에 대한 자세한 내용은 을 참조하십시오. the section called “발신 트래픽” ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:vpcconnector/` *`connector-name`* `[/`*`connector-revision`* `[/`*`connector-id`* `]]`
VpcIngressConnection	수신 트래픽을 구성하는 데 사용되는 AWS App Runner 리소스입니다. VPC 인터페이스 엔드포인트와 앱 러너 서비스를 연결하여 Amazon VPC 내에서만 앱 러너 서비스에 액세스할 수 있도록 합니다. VPC의 기능에 대한 자세한 내용은 IngressConnection 을 참조하십시오. the section called “프라이빗 엔드포인트 활성화” ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:vpcingressconnection/` *`vpc-ingress-connection-name`* `[/`*`connector-id`* `]]`

앱 러너 리소스 할당량

AWS 각 계정의 리소스 사용량에 대해 계정에 일부 할당량 (한도라고도 함) 을 부과합니다. AWS AWS 리전다음 표에는 App Runner 리소스와 관련된 할당량이 나열되어 있습니다. 할당량은 AWS App Runner 엔드포인트와 의 할당량에도 나열되어 있습니다. AWS 일반 참조

리소스 쿼터		설명	기본값	조정 가능한가요?
Services		계정에서 각 서비스에 대해 생성할 수 있는 최대 서비스 수입니다 AWS 리전.	30	✓ 예
Connections		계정에서 각 연결에 대해 만들 수 있는 최대 연결 수입니다 AWS 리전. 여러 서비스에서 단일 연결을 사용할 수 있습니다.	10	✓ 예
Auto scaling configurations	이름	계정에서 생성한 Auto Scaling 구성에서 각각에 대해 가질 수 있는 최대 고유 이름 수입니다 AWS 리전. 여러 서비스에서 하나의 Auto scaling 구성을 사용할 수 있습니다.	10	✓ 예
	이름별 수정 횟수	계정에서 각 AWS 리전 고유 이름에 대해 생성할 수 있는 최대 Auto Scaling 구성 수정 개수입니다. 여러 서비스에서 단일 Auto Scaling 구성 버전을 사용할 수 있습니다.	5	× 아니요
Observability configurations	이름	계정에서 각 AWS 리전항목에 대해 생성한 옵저버빌리티 구성에서 가질 수 있는 최대 고유 이름 수입니다. 여러 서비스에서 하나의 관찰성 구성을 사용할 수 있습니다.	10	✓ 예
	이름별 수정 횟수	계정에서 각 AWS 리전 고유 이름에 대해 생성할 수 있는 옵저버빌리티 구성 수정 개정의 최대 수입니다. 여러 서비스에서 단일 통합 가시성 구성 수정 버전을 사용할 수 있습니다.	10	× 아니요

리소스 쿼터	설명	기본값	조정 가능한가요?
VPC connectors	계정에서 각 VPC 커넥터를 만들 수 있는 최대 VPC 커넥터 수입니다. AWS 리전여러 서비스에서 하나의 VPC 커넥터를 사용할 수 있습니다.	10	✓ 예
VPC Ingress Connection	계정에서 생성할 수 있는 각 VPC 인그레스 연결의 최대 수입니다. AWS 리전단일 VPC 인그레스 연결을 사용하여 여러 App Runner 서비스에 액세스할 수 있습니다.	1	× 아니요

대부분의 할당량은 조정 가능하며 할당량 증가를 요청할 수 있습니다. 자세한 내용은 Service Quotas 사용 설명서에서 할당량 증가 요청을 참조하세요.

소스 이미지 기반 App Runner 서비스

를 AWS App Runner 사용하여 근본적으로 다른 두 가지 유형의 서비스 소스, 즉 소스 코드와 소스 이미지를 기반으로 서비스를 만들고 관리할 수 있습니다. 소스 유형에 관계없이 App Runner는 서비스의 시작, 실행, 규모 조정, 부하 분산을 담당합니다. App Runner의 CI/CD 기능을 사용하여 소스 이미지 또는 코드의 변경 사항을 추적할 수 있습니다. App Runner는 변경 사항을 발견하면 소스 코드용으로 자동으로 빌드하여 새 버전을 App Runner 서비스에 배포합니다.

이 장에서는 소스 이미지를 기반으로 하는 서비스에 대해 설명합니다. 소스 코드 기반 서비스에 대한 자세한 내용은 을 참조하십시오코드 기반 서비스.

소스 이미지는 이미지 리포지토리에 저장된 공개 또는 비공개 컨테이너 이미지입니다. App Runner를 이미지로 가리키면 App Runner가 이 이미지를 기반으로 컨테이너를 실행하는 서비스를 시작합니다. 빌드 단계는 필요하지 않습니다. 차라리 ready-to-deploy 이미지를 제공하면 됩니다.

이미지 리포지토리 제공자

App Runner는 다음 이미지 저장소 제공업체를 지원합니다.

- Amazon Elastic 컨테이너 레지스트리 (Amazon ECR) — 전용 이미지를 저장합니다. AWS 계정
- Amazon Elastic 컨테이너 레지스트리 퍼블릭 (Amazon ECR 퍼블릭) — 공개적으로 읽을 수 있는 이미지를 저장합니다.

공급자 사용 사례

- 계정에서 Amazon ECR에 저장된 이미지 사용 AWS
- 다른 AWS 계정에서 Amazon ECR에 저장된 이미지 사용
- Amazon ECR 퍼블릭에 저장된 이미지 사용

계정에서 Amazon ECR에 저장된 이미지 사용 AWS

Amazon ECR은 이미지를 리포지토리에 저장합니다. 프라이빗 리포지토리와 퍼블릭 리포지토리가 있습니다. 비공개 리포지토리에서 App Runner 서비스에 이미지를 배포하려면 App Runner가 Amazon ECR에서 이미지를 읽을 수 있는 권한이 필요합니다. 앱 러너에 해당 권한을 부여하려면 앱 러너에 액세스 역할을 제공해야 합니다. 이 역할은 필요한 Amazon ECR 작업 권한을 가진 AWS Identity and Access Management (IAM) 역할입니다. App Runner 콘솔을 사용하여 서비스를 생성할 때 계정의 기

존 역할을 선택할 수 있습니다. 또는 IAM 콘솔을 사용하여 새 사용자 지정 역할을 생성할 수 있습니다. 또는 App Runner 콘솔에서 관리형 정책을 기반으로 역할을 생성하도록 선택할 수 있습니다.

App Runner API 또는 를 사용하는 AWS CLI경우 2단계 프로세스를 완료합니다. 먼저 IAM 콘솔을 사용하여 액세스 역할을 생성합니다. App Runner에서 제공하는 관리형 정책을 사용하거나 사용자 지정 권한을 입력할 수 있습니다. 그런 다음 CreateServiceAPI 작업을 사용하여 서비스를 생성하는 동안 액세스 역할을 제공합니다.

App Runner 서비스 생성에 대한 자세한 내용은 을 참조하십시오the section called “생성”.

다른 AWS 계정에서 Amazon ECR에 저장된 이미지 사용

App Runner 서비스를 생성할 때 서비스가 속한 계정이 아닌 다른 AWS 계정에 속하는 Amazon ECR 리포지토리에 저장된 이미지를 사용할 수 있습니다. 교차 계정 이미지를 사용할 때는 이전 섹션에서 동일 계정 이미지에 대해 나열한 사항 외에도 염두에 두어야 할 몇 가지 추가 고려 사항이 있습니다.

- 교차 계정 저장소에는 정책이 연결되어 있어야 합니다. 리포지토리 정책은 액세스 역할에 리포지토리의 이미지를 읽을 수 있는 권한을 제공합니다. 이를 위해 다음 정책을 사용하십시오. 액세스 *access-role-arn* 역할의 Amazon 리소스 이름 (ARN) 으로 대체하십시오.

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

  Amazon ECR 리포지토리에 리포지토리 정책을 연결하는 방법에 대한 자세한 내용은 Amazon Elastic Container 레지스트리 사용 설명서의 리포지토리 정책 설명 설정을 참조하십시오.
- App Runner는 서비스가 속해 있는 계정과 다른 계정의 Amazon ECR 이미지 자동 배포를 지원하지 않습니다.

Amazon ECR 퍼블릭에 저장된 이미지 사용

Amazon ECR 퍼블릭은 공개적으로 읽을 수 있는 이미지를 저장합니다. Amazon ECR과 Amazon ECR Public 간의 주요 차이점은 다음과 같으며, 앱 러너 서비스와 관련하여 알아두어야 합니다.

- Amazon ECR 퍼블릭 이미지는 공개적으로 읽을 수 있습니다. Amazon ECR Public 이미지를 기반으로 서비스를 생성할 때는 액세스 역할을 제공할 필요가 없습니다. 리포지토리에는 정책을 첨부할 필요가 없습니다.
- 앱 러너는 Amazon ECR 퍼블릭 이미지에 대한 자동 (연속) 배포를 지원하지 않습니다.

Amazon ECR 퍼블릭에서 직접 서비스 시작

Amazon ECR Public Gallery에서 호스팅되는 호환 가능한 웹 애플리케이션의 컨테이너 이미지를 App Runner에서 실행되는 웹 서비스로 직접 시작할 수 있습니다. 갤러리를 탐색할 때 갤러리 페이지에서 Launch with App Runner를 찾아 이미지를 찾아보십시오. 이 옵션이 있는 이미지는 App Runner와 호환됩니다. 갤러리에 대한 자세한 내용은 Amazon ECR 퍼블릭 사용 설명서의 Amazon ECR 퍼블릭 갤러리 사용을 참조하십시오.

갤러리 이미지를 앱 러너 서비스로 시작하려면

1. 이미지의 갤러리 페이지에서 App Runner로 실행을 선택합니다.

   결과: App Runner 콘솔이 새 브라우저 탭에서 열립니다. 콘솔에는 필요한 새 서비스 세부 정보가 대부분 미리 채워진 서비스 만들기 마법사가 표시됩니다.

2. 콘솔에 표시된 지역이 아닌 다른 AWS 지역에서 서비스를 생성하려면 콘솔 헤더에 표시된 지역을 선택합니다. 그런 다음 다른 지역을 선택합니다.

3. 포트에는 이미지 애플리케이션이 수신하는 포트 번호를 입력합니다. 일반적으로 이미지의 갤러리 페이지에서 찾을 수 있습니다.
4. 필요한 경우 기타 구성 세부 정보를 변경할 수 있습니다.
5. [다음] 을 선택하고 설정을 검토한 다음 [만들기 및 배포] 를 선택합니다.

이미지 예제

App Runner 팀은 Amazon ECR 퍼블릭 갤러리에서 hello-app-runner예제 이미지를 유지 관리합니다. 이 예제를 사용하여 이미지 기반 App Runner 서비스 생성을 시작할 수 있습니다. 자세한 내용은 을 참조하십시오. hello-app-runner

소스 코드 기반 앱 러너 서비스

를 AWS App Runner 사용하여 근본적으로 다른 두 가지 유형의 서비스 소스, 즉 소스 코드와 소스 이미지를 기반으로 서비스를 만들고 관리할 수 있습니다. 소스 유형에 관계없이 App Runner는 서비스의 시작, 실행, 확장, 부하 분산을 담당합니다. App Runner의 CI/CD 기능을 사용하여 소스 이미지 또는 코드의 변경 사항을 추적할 수 있습니다. App Runner는 변경 사항을 발견하면 소스 코드용으로 자동으로 빌드하여 새 버전을 App Runner 서비스에 배포합니다.

이 장에서는 소스 코드를 기반으로 하는 서비스에 대해 설명합니다. 소스 이미지 기반 서비스에 대한 자세한 내용은 을 참조하십시오이미지 기반 서비스.

소스 코드는 App Runner가 자동으로 빌드하고 배포하는 애플리케이션 코드입니다. App Runner를 코드 리포지토리의 소스 디렉터리로 가리키고 프로그래밍 플랫폼 버전에 해당하는 적절한 런타임을 선택합니다. App Runner는 런타임의 기본 이미지와 애플리케이션 코드를 기반으로 이미지를 빌드합니다. 그런 다음 이 이미지를 기반으로 컨테이너를 실행하는 서비스를 시작합니다.

App Runner는 편리한 플랫폼별 관리형 런타임을 제공합니다. 각 런타임은 소스 코드에서 컨테이너 이미지를 빌드하고 이미지에 언어 런타임 종속성을 추가합니다. Dockerfile과 같은 컨테이너 구성 및 빌드 지침을 제공할 필요가 없습니다.

이 장의 하위 항목에서는 App Runner가 지원하는 다양한 플랫폼, 즉 다양한 프로그래밍 환경 및 버전에 대한 관리형 런타임을 제공하는 관리형 플랫폼에 대해 설명합니다.

주제

소스 코드 리포지토리 제공자

App Runner는 소스 코드 리포지토리에서 소스 코드를 읽어 배포합니다. App Runner는 Bitbucket이라는 두 개의 소스 코드 저장소 제공자를 지원합니다. GitHub

소스 코드 리포지토리 공급자를 통해 배포

소스 코드 리포지토리에서 App Runner 서비스에 소스 코드를 배포하기 위해 App Runner는 해당 서비스에 대한 연결을 설정합니다. App Runner 콘솔을 사용하여 서비스를 만들 때는 App Runner에서 소스 코드를 배포할 수 있도록 연결 세부 정보와 소스 디렉터리를 제공합니다.

연결

서비스 생성 절차의 일부로 연결 세부 정보를 제공합니다. App Runner API 또는 를 사용하는 경우 연결은 별도의 리소스입니다. AWS CLI먼저 CreateConnectionAPI 작업을 사용하여 연결을 생성합니다. 그런 다음 CreateServiceAPI 작업을 사용하여 서비스를 생성하는 동안 연결의 ARN을 제공합니다.

소스 디렉터리

서비스를 생성할 때 소스 디렉토리도 제공합니다. 기본적으로 App Runner는 저장소의 루트 디렉터리를 소스 디렉터리로 사용합니다. 소스 디렉터리는 애플리케이션의 소스 코드 및 구성 파일을 저장하는 소스 코드 리포지토리 내 위치입니다. 빌드 및 시작 명령은 소스 디렉터리에서도 실행됩니다. App Runner API 또는 AWS CLI 를 사용하여 서비스를 만들거나 업데이트할 때는 CreateService및 UpdateServiceAPI 작업에 소스 디렉터리를 제공합니다. 자세한 내용은 이어지는 소스 디렉터리 단원을 참조하십시오.

App Runner 서비스 생성에 대한 자세한 내용은 을 참조하십시오. the section called “생성” App Runner 연결에 대한 자세한 내용은 을 참조하십시오. the section called “연결”

소스 디렉터리

App Runner 서비스를 생성할 때 리포지토리 및 브랜치와 함께 소스 디렉터리를 제공할 수 있습니다. 소스 디렉터리 필드의 값을 애플리케이션의 소스 코드 및 구성 파일을 저장하는 리포지토리 디렉터리 경로로 설정합니다. App Runner는 사용자가 제공한 소스 디렉터리 경로에서 빌드 및 시작 명령을 실행합니다.

루트 리포지토리 디렉토리의 소스 디렉토리 경로 값을 절대값으로 입력합니다. 값을 지정하지 않으면 리포지토리 최상위 디렉터리 (리포지토리 루트 디렉터리라고도 함) 가 기본값으로 사용됩니다.

최상위 저장소 디렉터리 외에 다른 소스 디렉터리 경로를 제공할 수도 있습니다. 이는 모노레포 리포지토리 아키텍처를 지원합니다. 즉, 여러 애플리케이션의 소스 코드가 하나의 리포지토리에 저장됩니다. 단일 저장소에서 여러 App Runner 서비스를 만들고 지원하려면 각 서비스를 생성할 때 다른 소스 디렉토리를 지정하십시오.

> **Note**
>
> 여러 App Runner 서비스에 동일한 소스 디렉터리를 지정하는 경우 두 서비스 모두 개별적으로 배포되고 운영됩니다.

`apprunner.yaml`구성 파일을 사용하여 서비스 매개변수를 정의하기로 선택한 경우 해당 파일을 저장소의 소스 디렉토리 폴더에 배치하십시오.

배포 트리거 옵션이 자동으로 설정된 경우 원본 디렉터리에서 변경 내용을 적용하면 자동 배포가 트리거됩니다. 원본 디렉터리 경로의 변경 사항만 자동 배포를 트리거합니다. 소스 디렉터리의 위치가 자동 배포의 범위에 어떤 영향을 미치는지 이해하는 것이 중요합니다. 자세한 내용은 의 자동 배포를 참조하십시오. 배포 방법

> **Note**
>
> App Runner 서비스에서 PHP 관리 런타임을 사용하고 기본 루트 리포지토리가 아닌 다른 소스 디렉터리를 지정하려는 경우 올바른 PHP 런타임 버전을 사용하는 것이 중요합니다. 자세한 정보는 PHP 플랫폼 사용을 참조하세요.

앱 러너 관리형 플랫폼

App Runner 관리형 플랫폼은 다양한 프로그래밍 환경에 대한 관리형 런타임을 제공합니다. 각 관리형 런타임을 통해 프로그래밍 언어 또는 런타임 환경 버전을 기반으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. 관리형 런타임을 사용하는 경우 App Runner는 관리형 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며, 언어 런타임 패키지는 물론 몇 가지 도구 및 널리 사용되는 종속성 패키지를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 구축합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는

App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

App Runner는 모든 배포 또는 서비스 업데이트 시 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

관리형 런타임 버전 및 앱 러너 빌드

이제 App Runner에서 애플리케이션을 위한 업데이트된 빌드 프로세스를 제공합니다. 현재는 2023년 12월 29일에 마지막으로 릴리스된 관리형 런타임 Python 3.11 및 Node.js 18에서 실행되는 서비스에 대한 새 빌드를 호출합니다. 이렇게 수정된 빌드 프로세스는 더 빠르고 효율적입니다. 또한 애플리케이션을 실행하는 데 필요한 소스 코드, 빌드 아티팩트 및 런타임만 포함하는 더 작은 크기로 최종 이미지를 생성합니다.

새 빌드 프로세스를 수정된 App Runner 빌드라고 하고 원래 빌드 프로세스를 원본 App Runner 빌드라고 합니다. 이전 버전의 런타임 플랫폼에 대한 주요 변경을 방지하기 위해 App Runner는 수정된 빌드를 특정 런타임 버전 (일반적으로 새로 릴리스되는 메이저 릴리스) 에만 적용합니다.

수정된 빌드가 특정 사용 사례에 이전 버전과 호환되도록 하고 애플리케이션 빌드를 보다 유연하게 구성할 수 있도록 `apprunner.yaml` 구성 파일에 새 구성요소를 도입했습니다. 이는 선택적 `pre-run`매개변수입니다. 다음 섹션에서 빌드에 대한 기타 유용한 정보와 함께 이 매개변수를 사용하는 경우를 설명합니다.

다음 표에는 특정 관리형 런타임 버전에 적용되는 App Runner 빌드 버전이 나와 있습니다. 현재 런타임에 대해 계속 알려드리기 위해 이 문서를 계속 업데이트할 예정입니다.

플랫폼	오리지널 빌드	수정된 빌드
Python – 출시 정보	• Python 3.8 • Python 3.7	• 파이썬 3.11 (!)
Node.js – 출시 정보	• Node.js 16 • Node.js 14 • Node.js 12	• Node.js 18
Corretto — 릴리스 정보	• Corretto 11	

플랫폼	오리지널 빌드	수정된 빌드
	• Corretto 8	
.NET – 릴리스 정보	• .NET 6	
PHP – 릴리스 정보	• PHP 8.1	
Ruby – 출시 정보	• 루비 3.1	
Go – 출시 정보	• Go 1	

Important

Python 3.11 — Python 3.11 관리 런타임을 사용하는 서비스의 빌드 구성에 대한 구체적인 권장 사항이 있습니다. 자세한 내용은 Python 플랫폼 주제를 참조하십시오특정 런타임 버전에 대한 콜아웃.

App Runner 빌드 및 마이그레이션에 대해 자세히 알아보기

수정된 빌드를 사용하는 최신 런타임으로 애플리케이션을 마이그레이션할 때 빌드 구성을 약간 수정해야 할 수 있습니다.

마이그레이션 고려 사항에 대한 컨텍스트를 제공하기 위해 먼저 원본 App Runner 빌드와 수정된 빌드의 상위 수준 프로세스를 설명하겠습니다. 이어서 일부 구성 업데이트가 필요할 수 있는 서비스의 특정 속성을 설명하는 섹션을 살펴보겠습니다.

오리지널 App Runner 빌드

오리지널 App Runner 애플리케이션 빌드 프로세스는 서비스를 활용합니다. AWS CodeBuild 초기 단계는 서비스에서 큐레이션한 이미지를 기반으로 합니다. CodeBuild 해당하는 App Runner 관리 런타임 이미지를 기본 이미지로 사용하는 Docker 빌드 프로세스가 이어집니다.

일반적인 단계는 다음과 같습니다.

1. CodeBuild큐레이션된 이미지에서 `pre-build` 명령을 실행합니다.

   `pre-build`명령은 선택 사항입니다. `apprunner.yaml`구성 파일에서만 지정할 수 있습니다.

2. 이전 단계의 동일한 CodeBuild 이미지에서 를 사용하여 `build` 명령을 실행합니다.

   `build`명령이 필요합니다. 앱 러너 콘솔, 앱 러너 API 또는 `apprunner.yaml` 구성 파일에서 지정할 수 있습니다.

3. Docker 빌드를 실행하여 특정 플랫폼 및 런타임 버전에 대한 App Runner 관리형 런타임 이미지를 기반으로 이미지를 생성합니다.
4. 2단계에서 생성한 이미지에서 `/app` 디렉토리를 복사합니다. 대상은 3단계에서 생성한 App Runner 관리 런타임 이미지를 기반으로 하는 이미지입니다.
5. 생성된 App Runner 관리 런타임 이미지에서 `build` 명령을 다시 실행합니다. 빌드 명령을 다시 실행하여 4단계에서 복사한 `/app` 디렉터리의 소스 코드로부터 빌드 아티팩트를 생성합니다. 나중에 App Runner에서 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행할 것입니다.

   `build`명령은 필수입니다. 앱 러너 콘솔, 앱 러너 API 또는 `apprunner.yaml` 구성 파일에서 지정할 수 있습니다.

6. 2단계의 CodeBuild 이미지에서 `post-build` 명령을 실행합니다.

   `post-build`명령은 선택 사항입니다. `apprunner.yaml`구성 파일에서만 지정할 수 있습니다.

빌드가 완료되면 App Runner는 5단계에서 생성된 App Runner 관리 런타임 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

수정된 앱 러너 빌드

수정된 빌드 프로세스는 이전 섹션에서 설명한 원래 빌드 프로세스보다 더 빠르고 효율적입니다. 이전 버전 빌드에서 발생하는 빌드 명령의 중복을 제거합니다. 또한 애플리케이션을 실행하는 데 필요한 소스 코드, 빌드 아티팩트 및 런타임만 포함하는 더 작은 크기로 최종 이미지를 생성합니다.

이 빌드 프로세스는 Docker 다단계 빌드를 사용합니다. 일반적인 프로세스 단계는 다음과 같습니다.

1. 빌드 단계 - App Runner 빌드 이미지를 기반으로 `pre-build` 실행하고 `build` 명령을 실행하는 docker 빌드 프로세스를 시작합니다.
   a. 애플리케이션 소스 코드를 디렉터리에 복사합니다. `/app`

   > **Note**
   >
   > 이 `/app` 디렉토리는 Docker 빌드의 모든 단계에서 작업 디렉토리로 지정됩니다.

   b. `pre-build` 명령 실행

pre-build명령은 선택 사항입니다. apprunner.yaml구성 파일에서만 지정할 수 있습니다.

c. build명령을 실행합니다.

build명령이 필요합니다. 앱 러너 콘솔, 앱 러너 API 또는 apprunner.yaml 구성 파일에서 지정할 수 있습니다.

2. 패키징 단계 — App Runner 실행 이미지를 기반으로 최종 고객 컨테이너 이미지를 생성합니다.

a. 이전 빌드 단계의 **/app** 디렉터리를 새 Run 이미지로 복사합니다. 여기에는 애플리케이션 소스 코드와 이전 단계의 빌드 아티팩트가 포함됩니다.

b. pre-run명령을 실행합니다. build명령을 사용하여 /app 디렉터리 외부에서 런타임 이미지를 수정해야 하는 경우 apprunner.yaml 구성 파일의 이 세그먼트에 동일하거나 필요한 명령을 추가하십시오.

이는 수정된 App Runner 빌드를 지원하기 위해 도입된 새 매개변수입니다.

pre-run명령은 선택 사항입니다. apprunner.yaml구성 파일에서만 지정할 수 있습니다.

> **참고**
>
> - pre-run명령은 수정된 빌드에서만 지원됩니다. 서비스에서 원본 빌드를 사용하는 런타임 버전을 사용하는 경우에는 구성 파일에 추가하지 마세요.
> - 명령을 사용하여 /app 디렉터리 외부의 내용을 수정할 필요가 없는 경우에는 build pre-run 명령을 지정할 필요가 없습니다.

3. 빌드 후 단계 — 이 단계는 빌드 단계부터 다시 시작되어 명령을 실행합니다post-build.

a. 디렉터리 내에서 post-build 명령을 /app 실행합니다.

post-build명령은 선택 사항입니다. apprunner.yaml구성 파일에서만 지정할 수 있습니다.

빌드가 완료되면 App Runner는 Run 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

> **Note**
>
> 빌드 프로세스를 구성할 apprunner.yaml 때 Run 섹션의 env 항목에 현혹되지 마세요. 2단계 (b) 에서 참조한 pre-run 명령 매개변수가 Run 섹션에 있더라도 Run 섹션의 env 매개변수를 사용하여 빌드를 구성하지 마세요. pre-run명령은 구성 파일의 Build 섹션에 정의된 env 변수만 참조합니다. 자세한 내용은 App Runner 구성 파일 장을 참조하십시오실행 섹션.

마이그레이션 고려를 위한 서비스 요구 사항

애플리케이션 환경에 이 두 가지 요구 사항 중 하나가 있는 경우 `pre-run` 명령을 추가하여 빌드 구성을 수정해야 합니다.

- `build`명령을 사용하여 `/app` 디렉터리 외부의 내용을 수정해야 하는 경우.
- `build`명령을 두 번 실행하여 필요한 환경을 만들어야 하는 경우 이는 매우 특이한 요구 사항입니다. 대부분의 빌드에서는 이 작업을 수행하지 않습니다.

`/app`디렉터리 외부의 수정

- 수정된 App Runner 빌드에서는 애플리케이션에 디렉터리 외부의 종속성이 없는 것으로 가정합니다. `/app`
- `apprunner.yaml`파일, App Runner API 또는 App Runner 콘솔과 함께 제공하는 명령은 디렉터리에 빌드 아티팩트를 생성해야 합니다. `/app`
- `pre-build`,`build`, `post-build` 명령을 수정하여 모든 빌드 아티팩트가 디렉터리에 있도록 할 수 있습니다. `/app`
- 애플리케이션에서 서비스를 위해 생성된 이미지를 추가로 수정하기 위해 빌드가 필요한 경우 `/app` 디렉토리 외부에서 에서 새 `pre-run` 명령을 사용할 수 있습니다. `apprunner.yaml` 자세한 정보는 구성 파일을 사용하여 App Runner 서비스 옵션 설정을 참조하세요.

`build`명령을 두 번 실행합니다.

- 원래 App Runner 빌드는 `build` 명령을 두 번 실행합니다. 처음에는 2단계에서, 그 다음에는 5단계에서 다시 실행합니다. 개정된 App Runner 빌드는 이러한 중복성을 해결하고 명령을 한 번만 실행합니다. `build` 애플리케이션에 `build` 명령을 두 번 실행해야 하는 특별한 요구 사항이 있는 경우 수정된 App Runner 빌드는 매개변수를 사용하여 동일한 명령을 지정하고 다시 실행할 수 있는 옵션을 제공합니다. `pre-run` 이렇게 하면 동일한 이중 빌드 동작이 유지됩니다.

Python 플랫폼 사용

AWS App Runner Python 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 통해 Python 버전을 기반으로 하는 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Python 런타임을 사용하는 경우 App Runner는 관리되는 Python 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며 Python 버전의 런타임 패키지와 일부 도구 및 널리 사용되는 종속성 패키지를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이

션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한 Python 런타임 이름과 버전은 을 참조하십시오the section called “출시 정보”.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

Python 런타임의 버전 구문: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

예: `3.8.5`

다음 예제는 버전 잠금을 보여줍니다.

- `3.8`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `3.8.5`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

- Python 런타임 구성
- 특정 런타임 버전에 대한 쿨아웃
- Python 런타임 예제
- Python 런타임 릴리스 정보

Python 런타임 구성

관리형 런타임을 선택할 때는 최소한 명령을 빌드하고 실행하도록 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.

- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.
- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

특정 런타임 버전에 대한 콜아웃

> **Note**
>
> 이제 App Runner는 Python 3.11 및 Node.js 18의 런타임 버전을 기반으로 애플리케이션의 업데이트된 빌드 프로세스를 실행합니다. 애플리케이션이 이러한 런타임 버전 중 하나에서 실행되는 경우 수정된 빌드 관리형 런타임 버전 및 앱 러너 빌드 프로세스에 대한 자세한 내용은 을 참조하십시오. 다른 모든 런타임 버전을 사용하는 애플리케이션은 영향을 받지 않으며 원래 빌드 프로세스를 계속 사용합니다.

Python 3.11 (개정된 앱 러너 빌드)

관리되는 Python 3.11 런타임에 대해 apprunner.yaml의 다음 설정을 사용하십시오.

- 상단 섹션의 키를 다음과 같이 설정합니다. `runtime` `python311`

  Example

  ```
  runtime: python311
  ```

- `pip3`대신 `pip` 를 사용하여 종속성을 설치하십시오.
- 대신 `python3` 인터프리터를 사용하세요. `python`
- `pip3`설치 프로그램을 명령으로 `pre-run` 실행합니다. Python은 디렉토리 외부에 종속성을 설치합니다. `/app` App Runner는 Python 3.11용으로 수정된 App Runner 빌드를 실행하므로 `apprunner.yaml` 파일의 Build 섹션에 있는 명령을 통해 `/app` 디렉토리 외부에 설치된 모든 것은 손실됩니다. 자세한 정보는 수정된 앱 러너 빌드을 참조하세요.

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 -
```

자세한 내용은 이 항목 뒷부분의 Python 3.11용 확장 구성 파일 예제도 참조하십시오.

Python 런타임 예제

다음 예제는 Python 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다. 마지막 예는 Python 런타임 서비스에 배포할 수 있는 전체 Python 응용 프로그램의 소스 코드입니다.

> **Note**
>
> *# #### #### ### ### 3.7.7# 3.11###.* 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Python 런타임 버전은 을 참조하십시오the section called “출시 정보”.

최소 Python 구성 파일

이 예제는 Python 관리 런타임에서 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오. the section called “구성 파일 예제”

파이썬 3.11은 `pip3` 및 `python3` 명령을 사용합니다. 자세한 내용은 이 항목 뒷부분의 Python 3.11용 확장 구성 파일 예제를 참조하십시오.

Example apprunner.yaml

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

```
run:
  command: python app.py
```

확장된 Python 구성 파일

이 예제는 Python 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *3.7.7###*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Python 런타임 버전은 을 참조하십시오the section called “출시 정보”.
> 파이썬 3.11은 pip3 및 python3 명령을 사용합니다. 자세한 내용은 이 항목 뒷부분의 Python 3.11용 확장 구성 파일 예제를 참조하십시오.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-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"
```

확장된 파이썬 구성 파일 — 파이썬 3.11 (수정된 빌드 사용)

이 예제는 에서 Python 3.11 관리 런타임과 함께 모든 구성 키를 사용하는 방법을 보여줍니다. `apprunner.yaml` 이 Python 버전은 수정된 App Runner 빌드를 사용하므로 이 예제에는 `pre-run` 섹션이 포함되어 있습니다.

이 `pre-run` 매개변수는 수정된 App Runner 빌드에서만 지원됩니다. 애플리케이션이 원본 App Runner 빌드에서 지원하는 런타임 버전을 사용하는 경우 구성 파일에 이 매개변수를 삽입하지 마세요. 자세한 정보는 관리형 런타임 버전 및 앱 러너 빌드을 참조하세요.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *`3.11###`*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Python 런타임 버전은 을 참조하십시오the section called “출시 정보”.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-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"
```

완전한 Python 애플리케이션 소스

이 예제에서는 Python 런타임 서비스에 배포할 수 있는 전체 Python 응용 프로그램의 소스 코드를 보여줍니다.

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 aprunner.yaml

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

Python 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 Python 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 개정된 앱 러너 빌드

실행 시간 이름	마이너 버전	포함된 패키지
파이썬 3.11 (파이썬311)	3.11.9	SQLite 3.45.3
	3.11.8	SQLite 3.45.2
	3.11.8	SQLite 3.45.1
	3.11.7	SQLite 3.44.2

참고

- Python 3.11 — Python 3.11 관리 런타임을 사용하는 서비스의 빌드 구성에 대한 구체적인 권장 사항이 있습니다. 자세한 내용은 Python 플랫폼 주제를 참조하십시오특정 런타임 버전에 대한 콜아웃.
- App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

지원되는 런타임 버전 — 오리지널 App Runner 빌드

실행 시간 이름	마이너 버전	포함된 패키지
파이썬 3 (파이썬 3)	3.8.16	SQLite 3.45.3
	3.8.16	SQLite 3.45.2
	3.8.16	SQLite 3.45.1
	3.8.16	SQLite 3.44.2
	3.8.16	SQLite 3.43.2
	3.8.16	SQLite 3.43.1
	3.8.16	SQLite 3.43.0
	3.8.16	SQLite 3.42.0
	3.8.16	SQLite 3.41.2
	3.8.16	SQLite 3.40.1
	3.8.16	SQLite 3.40.0
	3.8.15	SQLite 3.40.0

실행 시간 이름	마이너 버전	포함된 패키지
	3.8.5	SQLite 3.39.4
	3.7.16	SQLite 3.45.3
	3.7.16	SQLite 3.45.2
	3.7.16	SQLite 3.45.1
	3.7.16	SQLite 3.44.2
	3.7.16	SQLite 3.43.2
	3.7.16	SQLite 3.43.1
	3.7.16	SQLite 3.43.0
	3.7.16	SQLite 3.42.0
	3.7.16	SQLite 3.41.2
	3.7.16	SQLite 3.40.1
	3.7.16	SQLite 3.40.0
	3.7.15	SQLite 3.40.0
	3.7.10	SQLite 3.40.0

Node.js 플랫폼 사용

AWS App Runner Node.js 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 통해 Node.js 버전을 기반으로 하는 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Node.js 런타임을 사용하는 경우 App Runner는 관리되는 Node.js 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며 Node.js 버전의 런타임 패키지와 일부 도구를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한 Node.js 런타임 이름 및 버전은 을 참조하십시오the section called “출시 정보”.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

Node.js 런타임의 버전 구문: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

예: `12.21.0`

다음 예제는 버전 잠금을 보여줍니다.

- `12.21`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `12.21.0`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

- Node.js 런타임 구성
- 특정 런타임 버전에 대한 콜아웃
- Node.js 런타임 예제
- Node.js 런타임 릴리스 정보

Node.js 런타임 구성

관리형 런타임을 선택할 때는 최소한 명령을 빌드하고 실행하도록 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.

- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

특히 Node.js 런타임을 사용하면 소스 리포지토리의 루트에 이름이 지정된 `package.json` JSON 파일을 사용하여 빌드와 런타임을 구성할 수도 있습니다. 이 파일을 사용하여 Node.js 엔진 버전, 종속성 패키지 및 다양한 명령 (명령줄 응용 프로그램) 을 구성할 수 있습니다. npm 또는 yarn과 같은 패키지 관리자는 이 파일을 명령의 입력으로 해석합니다.

예:

- npm install`dependencies`및 `devDependencies` 노드로 정의된 패키지를 설치합니다. `package.json`
- npm start또는 의 `scripts/start` 노드에서 `package.json` 정의한 명령을 npm run start 실행합니다.

다음은 예 `package.json` 파일입니다.

package.json

```
{
  "name": "node-js-getting-started",
  "version": "0.3.0",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "12.21.0"
  },
  "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"
  }
}
```

에 대한 자세한 내용은 npm *package.json* Docs 웹 사이트에서 package.json 파일 만들기를 참조하십시오.

팁

- package.json파일에 start 명령이 정의되어 있는 경우 다음 예와 같이 App Runner 구성 파일에서 run 명령으로 사용할 수 있습니다.

  Example

  package.json

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

  aprunner.yaml

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

- 개발 npm install 환경에서 실행하면 npm이 파일을 생성합니다. package-lock.json 이 파일에는 npm이 방금 설치한 패키지 버전의 스냅샷이 들어 있습니다. 이후 npm은 종속 항목을 설치할 때 이와 동일한 버전을 사용합니다. yarn을 설치하면 파일이 생성됩니다. yarn.lock 이러한 파일을 소스 코드 리포지토리에 커밋하여 애플리케이션이 개발 및 테스트에 사용한 종속성 버전과 함께 설치되었는지 확인하세요.
- 또한 App Runner 구성 파일을 사용하여 Node.js 버전과 시작 명령을 구성할 수 있습니다. 이렇게 하면 이러한 정의가 에 있는 정의보다 우선합니다. package.json 의 node package.json 버전과 App Runner 구성 파일의 runtime-version 값이 충돌하면 App Runner 빌드 단계가 실패합니다.

특정 런타임 버전에 대한 콜아웃

Node.js 18 (개정된 앱 러너 빌드)

이제 App Runner는 Python 3.11 및 Node.js 18의 런타임 버전을 기반으로 애플리케이션의 업데이트된 빌드 프로세스를 실행합니다. 애플리케이션이 이러한 런타임 버전 중 하나에서 실행되는 경우 수정된 빌드 관리형 런타임 버전 및 앱 러너 빌드 프로세스에 대한 자세한 내용은 을 참조하십시오. 다른 모든 런타임 버전을 사용하는 애플리케이션은 영향을 받지 않으며 원래 빌드 프로세스를 계속 사용합니다.

Node.js 런타임 예제

다음 예제는 Node.js 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다.

> **Note**
>
> *# #### #### ### ### 12.21.0# 18.19.0###.* 사용하려는 버전으로 바꿀 수 있습니다. 지원되는 최신 Node.js 런타임 버전은 을 참조하십시오the section called “출시 정보”.

최소 Node.js 구성 파일

이 예제는 Node.js 관리 런타임과 함께 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오. the section called “구성 파일 예제”

Example apprunner.yaml

```
version: 1.0
runtime: nodejs12
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

확장된 Node.js 구성 파일

이 예제에서는 Node.js 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *12.21.0###*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Node.js 런타임 버전은 을 참조하십시오the section called “출시 정보”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs12
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: 12.21.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

확장된 Node.js 구성 파일 — Node.js 18 (수정된 빌드 사용)

이 예제에서는 Node.js 관리 런타임과 함께 모든 구성 키를 에서 사용하는 방법을 보여줍니다`apprunner.yaml`. 이 Node.js 버전에서는 수정된 App Runner 빌드를 사용하므로 이 예제에는 `pre-run` 섹션이 포함되어 있습니다.

이 `pre-run` 파라미터는 수정된 App Runner 빌드에서만 지원됩니다. 애플리케이션이 원본 App Runner 빌드에서 지원하는 런타임 버전을 사용하는 경우 구성 파일에 이 매개변수를 삽입하지 마세요. 자세한 정보는 관리형 런타임 버전 및 앱 러너 빌드을 참조하세요.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *18.19.0###*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Node.js 런타임 버전은 을 참조하십시오the section called “출시 정보”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs18
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: 18.19.0
  pre-run:
    - node copy-global-files.js
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

그런트가 포함된 Node.js 앱

이 예제에서는 Grunt로 개발한 Node.js 애플리케이션을 구성하는 방법을 보여줍니다. Grunt는 명령줄 JavaScript 작업 실행기입니다. 반복적인 작업을 실행하고 프로세스 자동화를 관리하여 인적 오류를 줄입니다. Grunt 및 Grunt 플러그인은 npm을 사용하여 설치 및 관리됩니다. 소스 리포지토리의 루트에 `Gruntfile.js` 파일을 포함시켜 Grunt를 구성합니다.

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 aprunner.yaml

> **Note**
>
> *# #### #### ### ### 12.21.0###.* 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Node.js 런타임 버전은 을 참조하십시오the section called “출시 정보”.

```
version: 1.0
runtime: nodejs12
build:
  commands:
    pre-build:
      - npm install grunt grunt-cli
      - npm install --only=dev
      - npm run build
    build:
      - npm install --production
run:
  runtime-version: 12.21.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Node.js 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 Node.js 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 개정된 앱 러너 빌드

실행 시간 이름	마이너 버전	포함된 패키지
노드 18 (노드 18)	18.20.2	nodejs (npm 포함), 원사
	18.19.1	nodejs (npm 포함), 원사
	18.19.0	nodejs (npm 포함), 원사

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

지원되는 런타임 버전 — 오리지널 App Runner 빌드

실행 시간 이름	마이너 버전	포함된 패키지
노드 16 (노드 16)	16.20.2	nodejs (npm 포함), 원사
	16.20.1	nodejs (npm 포함), 원사
	16.20.0	nodejs (npm 포함), 원사
	16.19.1	nodejs (npm 포함), 원사
	16.19.0	nodejs (npm 포함), 원사
	16.18.1	nodejs (npm 포함), 원사
	16.17.1	nodejs (npm 포함), 원사
	16.17.0	nodejs (npm 포함), 원사
노드 14 (노드 14)	14.21.3	nodejs (npm 포함), 원사
	14.21.2	nodejs (npm 포함), 원사
	14.21.1	nodejs (npm 포함), 원사
	14.20.1	nodejs (npm 포함), 원사
	14.19.0	nodejs (npm 포함), 원사
노드 12 (노드 12)	12.22.12	nodejs (npm 포함), 원사
	12.21.0	nodejs (npm 포함), 원사

Java 플랫폼 사용

AWS App Runner Java 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 사용하면 Java 버전 기반 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Java 런타임을 사용하는 경우 App Runner는 관리형 Java 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며 Java 버전용 런타임 패키지와 일부 도구를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

현재 지원되는 모든 자바 런타임은 Amazon Corretto를 기반으로 합니다. 유효한 Java 런타임 이름 및 버전은 을 참조하십시오. the section called “릴리스 정보”

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

Amazon Corretto 런타임의 버전 구문:

런타임	구문	예
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`

다음 예는 버전 잠금을 보여줍니다.

- `11.0.13`— Open JDK 업데이트 버전을 잠급니다. 앱 러너는 오픈 JDK 및 Amazon Corretto 하위 레벨 빌드만 업데이트합니다.
- `11.0.13.08.1`— 특정 버전으로 잠글 수 있습니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

Java 런타임 구성

관리형 런타임을 선택할 때는 최소한 명령을 빌드하고 실행하도록 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.
- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

자바 런타임 예제

다음 예제는 Java 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다. 마지막 예는 Corretto 11 런타임 서비스에 배포할 수 있는 전체 Java 응용 프로그램의 소스 코드입니다.

> **Note**
>
> *`# #### #### ### ### 11.0.13.08.1###.`* 사용하려는 버전으로 바꿀 수 있습니다. 지원되는 최신 Java 런타임 버전은 을 참조하십시오the section called “릴리스 정보”.

미니멀 코레토 11 구성 파일

이 예제는 Corretto 11 관리형 런타임과 함께 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오.

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 .
```

확장된 Corretto 11 구성 파일

이 예제에서는 Corretto 11 관리형 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> *# ### ### ### ### 11.0.13.08.1###.* 사용하려는 버전으로 바꿀 수 있습니다. 지원되는 최신 Java 런타임 버전은 을 참조하십시오the section called “릴리스 정보”.

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"
```

완전한 Corretto 11 애플리케이션 소스

이 예제는 Corretto 11 런타임 서비스에 배포할 수 있는 전체 Java 응용 프로그램의 소스 코드를 보여줍니다.

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

```
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>
```

자바 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 Java 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 오리지널 App Runner 빌드

실행 시간 이름	마이너 버전	포함된 패키지
코레토 11 (코레토 11)	11.0.23.9.1	아마존 코레토 11 메이븐 3.9.6, 그라들 6.9.4
	11.0.22.7.1	아마존 코레토 11 메이븐 3.9.6, 그라들 6.9.4
	11.0.21.9.1	아마존 코레토 11 메이븐 3.9.6, 그라들 6.9.4
	11.0.21.9.1	아마존 코레토 11 메이븐 3.9.5, 그라들 6.9.4
	11.0.20.9.1	아마존 코레토 11 메이븐 3.9.4, 그라들 6.9.4
	11.0.20.8.1	아마존 코레토 11 메이븐 3.9.3, 그라들 6.9.4
	11.0.19.7.1	아마존 코레토 11 메이븐 3.9.3, 그라들 6.9.4
	11.0.19.7.1	아마존 코레토 11 메이븐 3.9.2, 그라들 6.9.4
	11.0.18.10.1	아마존 코레토 11 메이븐 3.9.1, 그라들 6.9.4
	11.0.18.10.1	아마존 코레토 11 메이븐 3.9.0, 그라들 6.9.4
	11.0.18.10.1	아마존 코레토 11 메이븐 3.8.7, 그라들 6.9.3
	11.0.17.8.1	아마존 코레토 11 메이븐 3.8.6, 그라들 6.9.3

실행 시간 이름	마이너 버전	포함된 패키지
	11.0.16.9.1	아마존 코레토 11 메이븐 3.8.6, 그라들 6.9.2
	11.0.13.08.1	아마존 코레토 11, 메이븐 3.6.3, 그라들 6.5
코레토 8 (코레토 8)	8.412.08.1	아마존 코레토 8, 메이븐 3.9.6, 그라들 6.9.4
	8.402.08.1	아마존 코레토 8, 메이븐 3.9.6, 그라들 6.9.4
	8.392.08.1	아마존 코레토 8, 메이븐 3.9.6, 그라들 6.9.4
	8.392.08.1	아마존 코레토 8, 메이븐 3.9.5, 그라들 6.9.4
	8.382.05.1	아마존 코레토 8, 메이븐 3.9.4, 그라들 6.9.4
	8.382.05.1	아마존 코레토 8, 메이븐 3.9.3, 그라들 6.9.4
	8.372.07.1	아마존 코레토 8, 메이븐 3.9.3, 그라들 6.9.4
	8.372.07.1	아마존 코레토 8, 메이븐 3.9.2, 그라들 6.9.4
	8.362.08.1	아마존 코레토 8, 메이븐 3.9.1, 그라들 6.9.4
	8.362.08.1	아마존 코레토 8, 메이븐 3.9.0, 그라들 6.9.4
	8.362.08.1	아마존 코레토 8 메이븐 3.8.7, 그라들 6.9.3

실행 시간 이름	마이너 버전	포함된 패키지
	8.352.08.1	아마존 코레토 8 메이븐 3.8.6, 그라들 6.9.3
	8.342.07.4	아마존 코레토 8 메이븐 3.8.6, 그라들 6.9.2
	8.312.07.1	아마존 코레토 8, 메이븐 3.6.3, 그라들 6.5

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

.NET 플랫폼 사용

AWS App Runner .NET 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 사용하면.NET 버전 기반 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. .NET 런타임을 사용하는 경우 App Runner는 관리형 .NET 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며, .NET 버전의 런타임 패키지와 일부 도구 및 널리 사용되는 종속성 패키지를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 구축합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한.NET 런타임 이름 및 버전은 을 참조하십시오. the section called “릴리스 정보”

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-`

`version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

.NET 런타임의 버전 구문: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

예: `6.0.9`

다음 예제는 버전 잠금을 보여줍니다.

- `6.0`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `6.0.9`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

- .NET 런타임 구성
- .NET 런타임 예제
- .NET 6 런타임 릴리스 정보

.NET 런타임 구성

관리형 런타임을 선택할 때는 최소한 빌드 및 실행 명령도 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.
- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

.NET 런타임 예제

다음 예제는.NET 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다. 마지막 예는 .NET 런타임 서비스에 배포할 수 있는 완전한.NET 애플리케이션의 소스 코드입니다.

> **Note**
>
> *# #### #### ### ### 6.0.9###.* 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 .NET 런타임 버전은 을 참조하십시오the section called “릴리스 정보”.

최소 .NET 구성 파일

이 예제는.NET 관리 런타임과 함께 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오. the section called “구성 파일 예제”

Example apprunner.yaml

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

확장된.NET 구성 파일

이 예제는.NET 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *6.0.9###*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 .NET 런타임 버전은 을 참조하십시오the section called “릴리스 정보”.

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"
```

완전한 .NET 애플리케이션 소스

이 예제는 .NET 런타임 서비스에 배포할 수 있는 완전한.NET 애플리케이션의 소스 코드를 보여줍니다.

> **Note**
>
> - 다음 명령을 실행하여 간단한 .NET 6 웹 앱을 생성합니다. `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - 생성된 .NET 6 웹 앱에 추가합니다. `apprunner.yaml`

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"
```

.NET 6 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 .NET 6 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 오리지널 앱 러너 빌드

실행 시간 이름	마이너 버전	포함된 패키지
닷넷 6 (닷넷6)	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

실행 시간 이름	마이너 버전	포함된 패키지
	6.0.10	.NET SDK 6.0.402
6.0.9	.NET SDK 6.0.401	

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

PHP 플랫폼 사용

AWS App Runner PHP 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 사용하여 PHP 버전을 기반으로 하는 웹 애플리케이션으로 컨테이너를 빌드하고 실행할 수 있습니다. PHP 런타임을 사용하는 경우 App Runner는 관리형 PHP 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며 PHP 버전용 런타임 패키지와 일부 도구를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한 PHP 런타임 이름 및 버전은 을 참조하십시오. the section called “릴리스 정보”

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

PHP 런타임의 버전 구문: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

예: `8.1.10`

버전 잠금의 예는 다음과 같습니다.

- `8.1`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `8.1.10`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

> **Important**
>
> App Runner 서비스의 코드 리포지토리 소스 디렉터리를 기본 리포지토리 루트 디렉터리가 아닌 다른 위치에 지정하려면 PHP 관리형 런타임 버전이 PHP 이상이어야 합니다. `8.1.22` 이전 버전의 PHP 런타임 버전은 기본 루트 소스 디렉터리만 사용할 `8.1.22` 수 있습니다.

주제

PHP 런타임 구성

관리형 런타임을 선택할 때는 최소한 빌드 및 실행 명령도 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.
- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

호환성

다음 웹 서버 중 하나를 사용하여 PHP 플랫폼에서 App Runner 서비스를 실행할 수 있습니다.

- Apache HTTP Server
- NGINX

Apache HTTP ServerPHP-FPM과도 NGINX 호환됩니다. 다음 중 하나를 NGINX사용하여 Apache HTTP ServerAND를 시작할 수 있습니다.

- 수퍼바이저 - 실행에 대한 자세한 내용은 수퍼바이저 실행을 참조하십시오. supervisord
- 시작 스크립트

Apache HTTP Server 또는 NGINX를 사용하여 PHP 플랫폼에서 App Runner 서비스를 구성하는 방법에 대한 예는 을 참조하십시오. the section called “완전한 PHP 애플리케이션 소스”

파일 구조

웹 서버 `root` 디렉터리 아래의 `public` 폴더에 `index.php` 설치해야 합니다.

> **Note**
>
> `startup.sh`또는 `supervisord.conf` 파일은 웹 서버의 루트 디렉터리에 저장하는 것이 좋습니다. `start`명령이 `startup.sh` 또는 `supervisord.conf` 파일이 저장된 위치를 가리키는지 확인하십시오.

다음은 를 사용하는 경우 파일 구조의 예입니다 supervisord.

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

다음은 시작 스크립트를 사용하는 경우의 파일 구조 예제입니다.

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

이러한 파일 구조는 App Runner 서비스용으로 지정된 코드 리포지토리 소스 디렉터리에 저장하는 것이 좋습니다.

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

> **Important**
>
> App Runner 서비스의 코드 리포지토리 소스 디렉터리를 기본 리포지토리 루트 디렉터리가 아닌 다른 위치에 지정하려면 PHP 관리 런타임 버전이 PHP 이상이어야 합니다. `8.1.22` 이전 버전의 PHP 런타임 버전은 기본 루트 소스 디렉터리만 사용할 `8.1.22` 수 있습니다.
> App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 버전 잠금을 지정하지 않은 한, 서비스는 기본적으로 최신 런타임을 사용합니다.

PHP 런타임 예제

다음은 PHP 서비스를 빌드하고 실행하는 데 사용되는 App Runner 구성 파일의 예시입니다.

최소 PHP 구성 파일

다음 예제는 PHP 관리 런타임과 함께 사용할 수 있는 최소 구성 파일입니다. 최소 구성 파일에 대한 자세한 내용은 을 참조하십시오the section called “구성 파일 예제”.

Example apprunner.yaml

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

확장된 PHP 구성 파일

다음 예제에서는 PHP 관리 런타임과 함께 모든 구성 키를 사용합니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *8.1.10###*. 사용하려는 버전으로 바꿀 수 있습니다. 지원되는 최신 PHP 런타임 버전은 을 참조하십시오the section called “릴리스 정보”.

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"
```

완전한 PHP 애플리케이션 소스

다음은 또는 를 사용하여 PHP 런타임 서비스에 배포하는 데 사용할 수 있는 PHP 응용 프로그램 소스 코드의 예입니다. Apache HTTP ServerNGINX 이 예제에서는 기본 파일 구조를 사용한다고 가정합니다.

를 사용하여 Apache HTTP Server PHP 플랫폼 실행 supervisord

Example 파일 구조

> **Note**
>
> - `supervisord.conf`파일은 저장소의 어느 곳에나 저장할 수 있습니다. `start`명령이 `supervisord.conf` 파일이 저장된 위치를 가리키는지 확인하십시오.
> - `root`디렉터리 아래의 `public` 폴더에 `index.php` 설치해야 합니다.

```
/
## 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 aprunner.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>
```

다음을 사용하여 Apache HTTP Server PHP 플랫폼 실행 startup script

Example 파일 구조

> **Note**
>
> - `startup.sh`파일은 저장소의 어느 곳에나 저장할 수 있습니다. `start`명령이 `startup.sh` 파일이 저장된 위치를 가리키는지 확인하십시오.
> - `root`디렉터리 아래의 `public` 폴더에 `index.php` 설치해야 합니다.

```
/
## 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**
>
> - `startup.sh`파일을 Git 리포지토리에 커밋하기 전에 실행 파일로 저장해야 합니다. 파일에 대한 실행 권한을 설정하는 `chmod +x startup.sh` 데 사용합니다. `startup.sh`
> - 파일을 실행 `startup.sh` 파일로 저장하지 않는 경우 `apprunner.yaml` 파일의 `chmod +x startup.sh build` 명령으로 를 입력합니다.

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>
```

다음을 사용하여 NGINX PHP 플랫폼 실행 supervisord

Example 파일 구조

> **Note**
>
> - `supervisord.conf`파일은 저장소의 어느 곳에나 저장할 수 있습니다. `start`명령이 `supervisord.conf` 파일이 저장된 위치를 가리키는지 확인하십시오.
> - `root`디렉터리 아래의 `public` 폴더에 `index.php` 설치해야 합니다.

```
/
## 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 aprunner.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>
```

다음을 사용하여 NGINX PHP 플랫폼 실행 startup script

Example 파일 구조

> **Note**
>
> - `startup.sh`파일은 저장소의 어느 곳에나 저장할 수 있습니다. `start`명령이 `startup.sh` 파일이 저장된 위치를 가리키는지 확인하십시오.
> - `root`디렉터리 아래의 `public` 폴더에 `index.php` 설치해야 합니다.

```
/
## 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**
>
> - `startup.sh`파일을 Git 리포지토리에 커밋하기 전에 실행 파일로 저장해야 합니다. 파일에 대한 실행 권한을 설정하는 `chmod +x startup.sh` 데 사용합니다. `startup.sh`

- 파일을 실행 `startup.sh` 파일로 저장하지 않는 경우 `apprunner.yaml` 파일의 `chmod +x startup.sh` `build` 명령으로 를 입력합니다.

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>
```

PHP 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 PHP 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 오리지널 앱 러너 빌드

실행 시간 이름	마이너 버전
PHP 8.1 (php81)	8.1.28
	8.1.27

실행 시간 이름	마이너 버전
	8.1.26
	8.1.24
	8.1.22
	8.1.21
	8.1.20
	8.1.19
	8.1.17
	8.1.16
	8.1.14
	8.1.13
	8.1.12
8.1.10	

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

Ruby 플랫폼 사용

AWS App Runner Ruby 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 통해 Ruby 버전을 기반으로 하는 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Ruby 런타임을 사용하는 경우 App Runner는 관리되는 Ruby 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며 Ruby 버전의 런타임 패키지와 일부 도구를 포함합니다. App Runner는 이 관

리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한 Ruby 런타임 이름 및 버전은 을 참조하십시오. the section called “출시 정보”

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

Ruby 런타임의 버전 구문: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

예: `3.1.2`

다음 예제는 버전 잠금을 보여줍니다.

- `3.1`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `3.1.2`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

- Ruby 런타임 구성
- Ruby 런타임 예제
- 루비 런타임 릴리스 정보

Ruby 런타임 구성

관리형 런타임을 선택할 때는 최소한 빌드 및 실행 명령도 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.

- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

Ruby 런타임 예제

다음 예제는 Ruby 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다.

최소 Ruby 구성 파일

이 예제는 Ruby 관리 런타임에서 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오. the section called “구성 파일 예제”

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
```

확장된 루비 설정 파일

이 예제는 Ruby 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *3.1.2###*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Ruby 런타임 버전은 을 참조하십시오the section called “출시 정보”.

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"
```

전체 루비 애플리케이션 소스

이 예제는 Ruby 런타임 서비스에 배포할 수 있는 전체 Ruby 애플리케이션의 소스 코드를 보여줍니다.

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 aprunner.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
```

루비 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 Ruby 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 오리지널 App Runner 빌드

실행 시간 이름	마이너 버전	포함된 패키지
루비 3.1 (루비31)	3.1.4	SQLite 3.45.3
	3.1.4	SQLite 3.45.2
	3.1.4	SQLite 3.45.1
	3.1.4	SQLite 3.44.2
	3.1.4	SQLite 3.43.2
	3.1.4	SQLite 3.43.1

실행 시간 이름	마이너 버전	포함된 패키지
	3.1.4	SQLite 3.43.0
	3.1.4	SQLite 3.42.0
	3.1.4	SQLite 3.41.2
	3.1.3	SQLite 3.41.0
	3.1.3	SQLite 3.40.0
	3.1.2	SQLite 3.39.4

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

Go 플랫폼 사용

AWS App Runner Go 플랫폼은 관리형 런타임을 제공합니다. 각 런타임을 통해 Go 버전을 기반으로 하는 웹 애플리케이션으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Go 런타임을 사용하는 경우 App Runner는 관리되는 Go 런타임 이미지로 시작합니다. 이 이미지는 Amazon Linux Docker 이미지를 기반으로 하며, Go 버전용 런타임 패키지와 일부 도구를 포함합니다. App Runner는 이 관리형 런타임 이미지를 기본 이미지로 사용하고 애플리케이션 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음 이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

App Runner 콘솔 또는 API 작업을 사용하여 서비스를 만들 때 App Runner 서비스의 런타임을 지정합니다. CreateService 런타임을 소스 코드의 일부로 지정할 수도 있습니다. 코드 리포지토리에 포함하는 App Runner 구성 파일에서 `runtime` 키워드를 사용하십시오. 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

유효한 Go 런타임 이름 및 버전은 을 참조하십시오the section called “출시 정보”.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 애플리케이션에 특정 버전의 관리형 런타임이 필요한 경우 App Runner 구성 파일의 `runtime-`

`version` 키워드를 사용하여 지정할 수 있습니다. 메이저 또는 마이너 버전을 포함하여 모든 수준의 버전으로 잠글 수 있습니다. App Runner는 서비스 런타임을 하위 수준으로만 업데이트합니다.

Go 런타임의 버전 구문: *major*`[.`*minor*`[.`*patch*`]]`

예: `1.18.7`

다음 예제는 버전 잠금을 보여줍니다.

- `1.18`— 메이저 버전과 마이너 버전을 잠급니다. App Runner는 패치 버전만 업데이트합니다.
- `1.18.7`— 특정 패치 버전으로 잠급니다. App Runner는 런타임 버전을 업데이트하지 않습니다.

주제

Go 런타임 구성

관리형 런타임을 선택할 때는 최소한 명령을 빌드하고 실행하도록 구성해야 합니다. App Runner 서비스를 만들거나 업데이트할 때 구성합니다. 다음 방법 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

- App Runner 콘솔 사용 — 생성 프로세스 또는 구성 탭의 빌드 구성 섹션에서 명령을 지정합니다.
- 앱 러너 API 사용 — CreateService또는 UpdateServiceAPI 작업을 호출합니다. CodeConfigurationValues데이터 유형의 `BuildCommand` 및 `StartCommand` 멤버를 사용하여 명령을 지정합니다.
- 구성 파일 사용 - 최대 3번의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일 제공은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때는 App Runner가 구성 설정을 만들 때 직접 가져오는지 아니면 구성 파일에서 가져오는지 지정합니다.

Go 런타임 예제

다음 예제는 Go 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여줍니다.

최소 Go 구성 파일

이 예제는 Go 관리 런타임과 함께 사용할 수 있는 최소 구성 파일을 보여줍니다. App Runner가 최소 구성 파일을 사용하여 가정하는 내용은 을 참조하십시오. the section called “구성 파일 예제”

Example apprunner.yaml

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

확장된 Go 구성 파일

이 예제에서는 Go 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여줍니다.

> **Note**
>
> 이 예제에 사용된 런타임 버전은 *`1.18.7###`*. 사용하려는 버전으로 교체할 수 있습니다. 지원되는 최신 Go 런타임 버전은 을 참조하십시오the section called “출시 정보”.

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"
```

완전한 Go 애플리케이션 소스

이 예제는 Go 런타임 서비스에 배포할 수 있는 전체 Go 애플리케이션의 소스 코드를 보여줍니다.

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
```

Go 1 런타임 릴리스 정보

이 항목에는 App Runner가 지원하는 Go 런타임 버전에 대한 전체 세부 정보가 나열되어 있습니다.

지원되는 런타임 버전 — 오리지널 앱 러너 빌드

실행 시간 이름	마이너 버전	
고 1 (go1)	1.18.10	
	1.18.9	
	1.18.8	
	1.18.7	

> **Note**
>
> App Runner는 최근에 출시된 특정 주요 런타임에 대해 수정된 빌드 프로세스를 제공합니다. 따라서 이 문서의 특정 섹션에서 수정된 App Runner 빌드와 원본 App Runner 빌드에 대한 참조를 볼 수 있습니다. 자세한 내용은 관리형 런타임 버전 및 앱 러너 빌드 단원을 참조하세요.

앱 러너용 애플리케이션 코드 개발

이 장에서는 배포할 응용 프로그램 코드를 개발하거나 마이그레이션할 때 고려해야 하는 런타임 정보와 개발 지침에 대해 설명합니다. AWS App Runner

런타임 정보

컨테이너 이미지를 제공하든 App Runner에서 빌드하든 관계없이 App Runner는 컨테이너 인스턴스에서 애플리케이션 코드를 실행합니다. 다음은 컨테이너 인스턴스 런타임 환경의 몇 가지 주요 측면입니다.

- 프레임워크 지원 — App Runner는 웹 애플리케이션을 구현하는 모든 이미지를 지원합니다. 어떤 프로그래밍 언어를 선택하든, 어떤 웹 애플리케이션 서버나 프레임워크를 사용하든 상관없이 사용할 수 있습니다. 편의를 위해 다양한 프로그래밍 플랫폼에 대한 플랫폼별 관리 런타임을 제공하여 애플리케이션 빌드 프로세스와 추상 이미지 생성을 간소화합니다.
- 웹 요청 — App Runner는 컨테이너 인스턴스에 HTTP 1.0 및 HTTP 1.1에 대한 지원을 제공합니다. 서비스 구성에 대한 자세한 내용은 을 참조하십시오the section called “구성”. HTTPS 보안 트래픽 처리를 구현하지 않아도 됩니다. App Runner는 들어오는 모든 HTTP 요청을 해당 HTTPS 엔드포인트로 리디렉션합니다. HTTP 웹 요청을 리디렉션할 수 있도록 설정하지 않아도 됩니다. App Runner는 요청을 애플리케이션 컨테이너 인스턴스로 전달하기 전에 TLS를 종료합니다.

  > **Note**
  >
  > - HTTP 요청에는 총 120초의 요청 제한 시간이 있습니다. 120초에는 애플리케이션이 본문을 포함하여 요청을 읽고 HTTP 응답 작성을 완료하는 데 걸리는 시간이 포함됩니다.
  > - 요청 읽기 및 응답 제한 시간은 사용하는 애플리케이션에 따라 다릅니다. 이러한 애플리케이션에는 자체 내부 시간 제한이 있을 수 있습니다. 예를 들어 Python용 HTTP 서버인 Gunicorn에는 30초의 기본 시간 제한 시간이 있습니다. 이러한 경우 애플리케이션의 제한 시간이 App Runner 120초 제한 시간보다 우선합니다.
  > - App Runner는 완전관리형 서비스이므로 TLS 종료를 자동으로 관리하므로 TLS 암호 제품군이나 기타 매개변수를 구성할 필요가 없습니다.

- 스테이트리스 앱 — 현재 App Runner는 스테이트풀 앱을 지원하지 않습니다. 따라서 App Runner는 단일 수신 웹 요청을 처리하는 기간 이후에도 상태 지속성을 보장하지 않습니다.
- 스토리지 — App Runner는 들어오는 트래픽 양에 따라 App Runner 애플리케이션의 인스턴스를 자동으로 늘리거나 줄입니다. App Runner 애플리케이션에 대한 Auto Scaling 옵션을 구성할 수 있습

니다. 웹 요청을 처리하는 현재 활성 인스턴스 수는 들어오는 트래픽 볼륨을 기반으로 하므로 App Runner는 파일이 단일 요청 처리 이후에도 계속 유지될 수 있다고 보장할 수 없습니다. 따라서 App Runner는 컨테이너 인스턴스의 파일 시스템을 임시 저장소로 구현하므로 파일이 일시적입니다. 예를 들어 App Runner 서비스를 일시 중지했다가 다시 시작해도 파일이 유지되지 않습니다.

App Runner는 3GB의 임시 스토리지를 제공하며 3GB의 임시 스토리지 중 일부를 인스턴스에서 가져와 압축하고 압축하지 않은 컨테이너 이미지를 저장하는 데 사용합니다. 나머지 임시 스토리지는 App Runner 서비스에서 사용할 수 있습니다. 하지만 이 스토리지는 스테이트리스 (Stateless) 특성 때문에 영구 스토리지는 아닙니다.

> **Note**
>
> 스토리지 파일이 여러 요청에 걸쳐 지속되는 시나리오가 있을 수 있습니다. 예를 들어, 다음 요청이 동일한 인스턴스에 도착하더라도 스토리지 파일은 계속 유지됩니다. 요청 간에 스토리지 파일이 계속 유지되는 것은 특정 상황에서 유용할 수 있습니다. 예를 들어 요청을 처리할 때 향후 요청에서 필요할 경우 애플리케이션에서 다운로드하는 파일을 캐시할 수 있습니다. 이렇게 하면 향후 요청 처리 속도가 빨라질 수 있지만 속도 향상을 보장할 수는 없습니다. 코드는 이전 요청에서 다운로드한 파일이 여전히 존재한다고 가정해서는 안 됩니다. 처리량이 높고 지연 시간이 짧은 인메모리 데이터 스토어를 사용하여 캐싱을 보장하려면 Amazon과 같은 서비스를 사용하십시오. ElastiCache

- 환경 변수 - 기본적으로 App Runner는 컨테이너 인스턴스에서 `PORT` 환경 변수를 사용할 수 있도록 합니다. 포트 정보로 변수 값을 구성하고 사용자 지정 환경 변수 및 값을 추가할 수 있습니다. AWS Secrets Manager또는 AWS Systems Manager 파라미터 스토어에 저장된 민감한 데이터를 환경 변수로 참조할 수도 있습니다. 환경 변수 생성에 대한 자세한 내용은 을 참조하십시오참조 환경 변수.
- 인스턴스 역할 - 애플리케이션 코드가 서비스 API 또는 AWS SDK 중 하나를 사용하여 AWS 서비스를 호출하는 경우 AWS Identity and Access Management (IAM) 을 사용하여 인스턴스 역할을 생성하십시오. 그런 다음 앱을 만들 때 App Runner 서비스에 연결하세요. 코드에 필요한 모든 AWS 서비스 작업 권한을 인스턴스 역할에 포함하세요. 자세한 정보는 the section called “인스턴스 역할”을 참조하세요.

코드 개발 가이드라인

App Runner 웹 애플리케이션용 코드를 개발할 때는 이 가이드라인을 고려하세요.

- 상태 비저장 코드 설계 - App Runner 서비스에 배포하는 웹 애플리케이션을 상태 비저장 모드로 설계하십시오. 코드에서는 수신되는 단일 웹 요청을 처리하는 기간 이후로는 어떤 상태도 지속되지 않는다고 가정해야 합니다.
- 임시 파일 삭제 - 파일을 만들면 파일 시스템에 저장되며 서비스 스토리지 할당의 일부를 차지합니다. out-of-storage 오류를 방지하려면 임시 파일을 장기간 보관하지 마세요. 파일 캐싱을 결정할 때는 스토리지 크기와 요청 처리 속도의 균형을 맞추세요.
- 인스턴스 시작 — App Runner는 5분의 인스턴스 시작 시간을 제공합니다. 인스턴스는 구성된 수신 포트에서 요청을 수신하고 시작 후 5분 이내에 정상 상태여야 합니다. 시작 시간 동안 앱 러너 인스턴스에는 vCPU 구성에 따라 가상 CPU (vCPU) 가 할당됩니다. 사용 가능한 vCPU 구성에 대한 자세한 내용은 을 참조하십시오. the section called “앱 러너 지원 구성”

  인스턴스가 성공적으로 시작되면 유휴 상태가 되어 요청을 기다립니다. 인스턴스 시작 시간을 기준으로 비용을 지불하며, 인스턴스 시작당 최소 요금은 1분입니다. 요금에 대한 자세한 정보는 AWS App Runner 요금을 참조하세요.

앱 러너 콘솔 사용

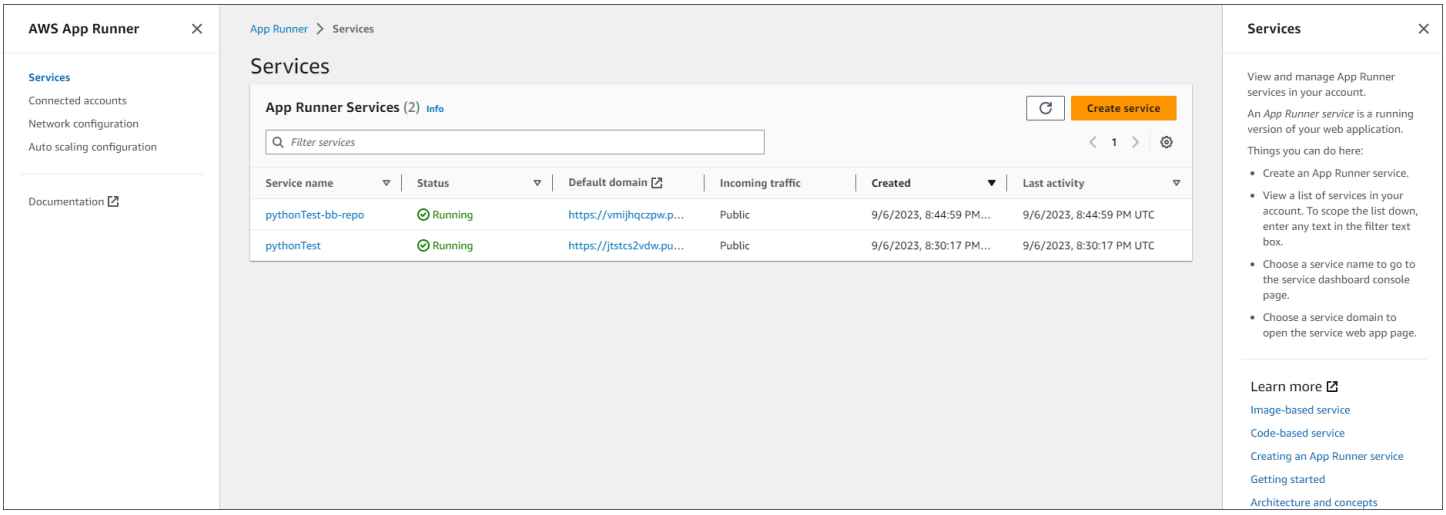
AWS App Runner 콘솔을 사용하여 App Runner 서비스와 관련 리소스 (예: 연결된 계정) 를 만들고, 관리하고, 모니터링할 수 있습니다. 기존 서비스를 보고, 새 서비스를 만들고, 서비스를 구성할 수 있습니다. App Runner 서비스의 상태를 볼 수 있을 뿐만 아니라 로그를 보고, 활동을 모니터링하고, 지표를 추적할 수 있습니다. 또한 서비스 웹 사이트 또는 소스 리포지토리로 이동할 수 있습니다.

다음 섹션에서는 콘솔의 레이아웃과 기능을 설명하고 관련 정보를 안내합니다.

전체 콘솔 레이아웃

App Runner 콘솔에는 세 가지 영역이 있습니다. 왼쪽에서 오른쪽으로:

- 탐색 창 - 축소하거나 확장할 수 있는 측면 창입니다. 이를 사용하여 사용하려는 최상위 콘솔 페이지를 선택할 수 있습니다.
- 콘텐츠 창 — 콘솔 페이지의 주요 부분입니다. 이를 사용하여 정보를 보고 작업을 수행할 수 있습니다.
- 도움말 창 — 자세한 내용을 볼 수 있는 측면 창입니다. 펼치면 현재 페이지에 대한 도움말을 볼 수 있습니다. 또는 콘솔 페이지에서 정보 링크를 선택하여 상황에 맞는 도움을 받을 수도 있습니다.



서비스 페이지

서비스 페이지에는 계정의 App Runner 서비스가 나열됩니다. 필터 텍스트 상자를 사용하여 목록의 범위를 좁힐 수 있습니다.

서비스 페이지로 이동하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택합니다.

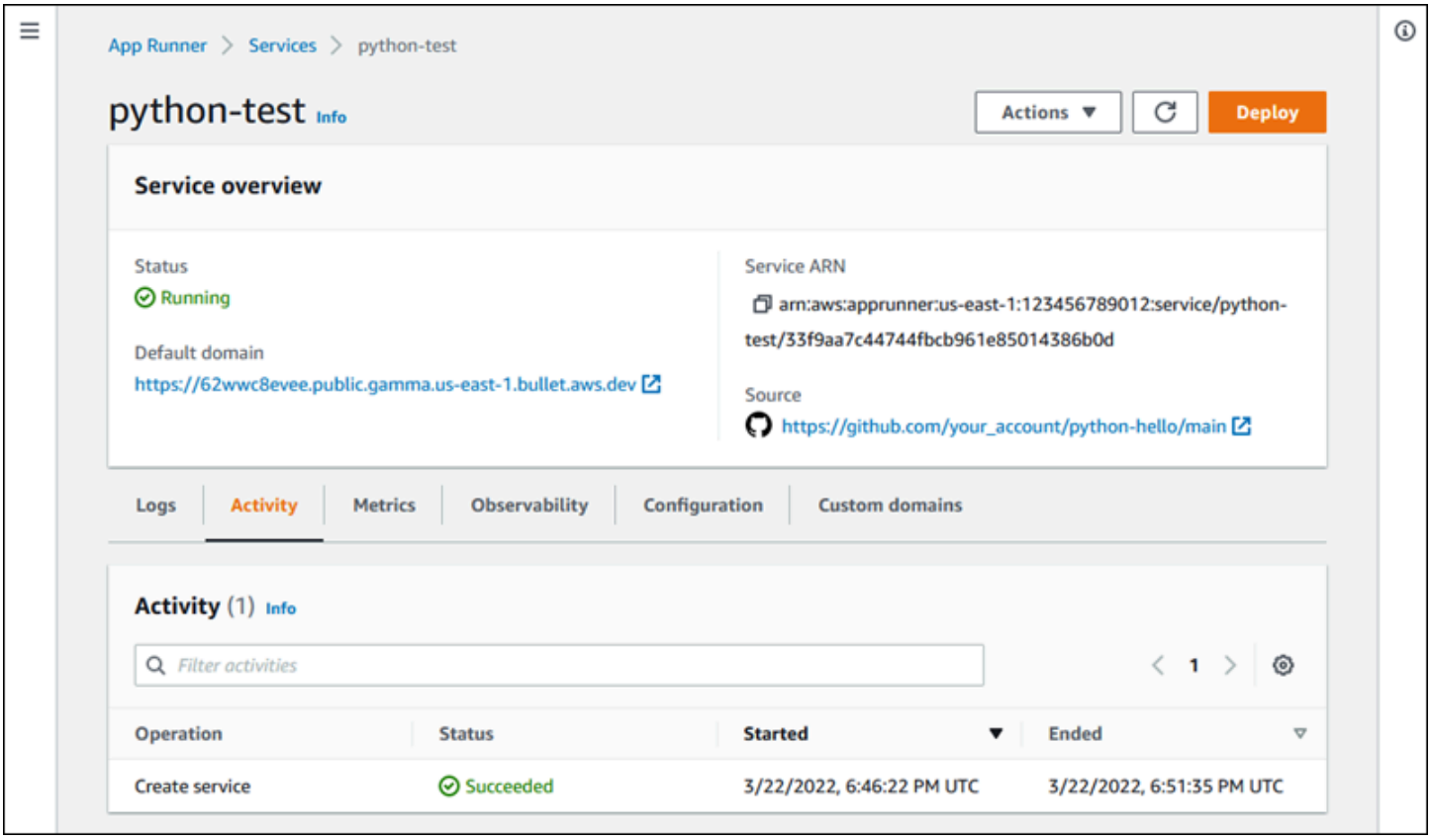
여기에서 수행할 수 있는 작업:

- 앱 러너 서비스를 만드세요. 자세한 정보는 the section called “생성”을 참조하세요.
- 서비스 이름을 선택하면 서비스 대시보드 콘솔 페이지로 이동합니다.
- 서비스 도메인을 선택하여 서비스 웹 앱 페이지를 엽니다.

서비스 대시보드 페이지

서비스 대시보드 페이지에서 App Runner 서비스에 대한 정보를 보고 관리할 수 있습니다. 페이지 상단에서 서비스 이름을 볼 수 있습니다.

서비스 대시보드로 이동하려면 서비스 페이지 (이전 섹션 참조) 로 이동한 다음 App Runner 서비스를 선택합니다.



서비스 개요 섹션에서는 App Runner 서비스 및 애플리케이션에 대한 기본 세부 정보를 제공합니다. 여기서 수행할 수 있는 작업은 다음과 같습니다.

- 상태, 상태, ARN과 같은 서비스 세부 정보를 볼 수 있습니다.
- Default 도메인으로 이동합니다. Default 도메인은 App Runner가 서비스에서 실행되는 웹 애플리케이션에 제공하는 도메인입니다. App Runner가 소유한 도메인의 하위 `awsapprunner.com` 도메인입니다.
- 서비스에 배포된 소스 리포지토리로 이동합니다.
- 서비스에 소스 리포지토리 배포를 시작합니다.
- 서비스를 일시 중지, 재개, 삭제합니다.

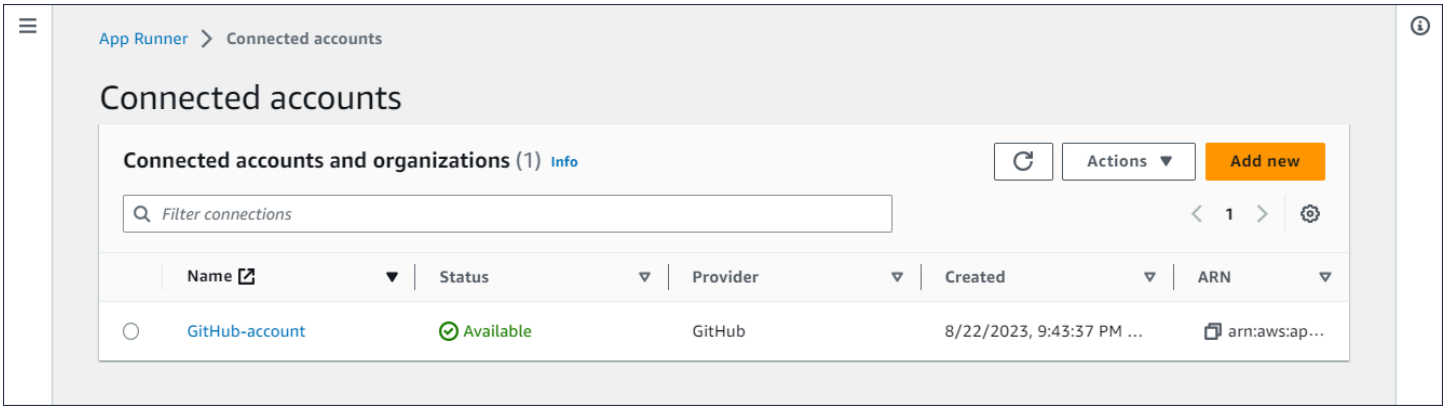
서비스 개요 아래의 탭은 서비스 관리 및 가시성을 위한 것입니다.

연결된 계정 페이지

연결된 계정 페이지에는 계정의 소스 코드 리포지토리 공급자에 대한 App Runner 연결이 나열됩니다. 필터 텍스트 상자를 사용하여 목록의 범위를 좁힐 수 있습니다. 연결된 계정에 대한 자세한 내용은 을 참조하십시오the section called “연결”.

연결된 계정 페이지로 이동하려면

1. App Runner 콘솔을 열고 지역 목록에서 사용자 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 연결된 계정을 선택합니다.



여기서 할 수 있는 일:

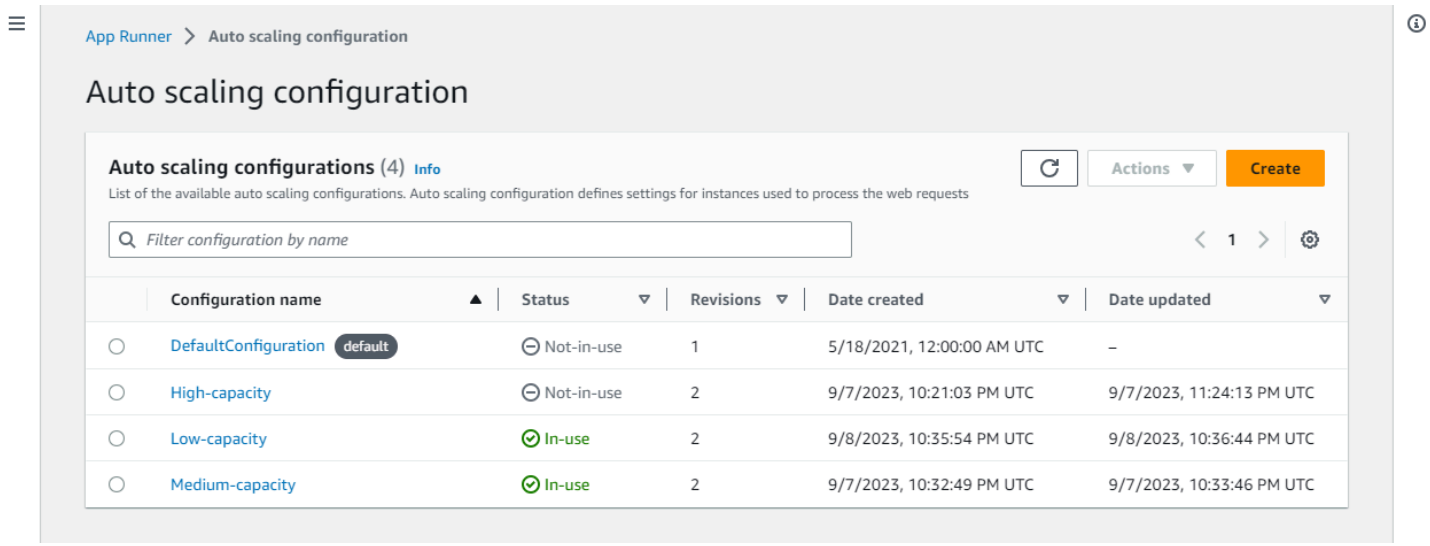
- 계정의 리포지토리 공급자 연결 목록을 볼 수 있습니다. 목록의 범위를 좁히려면 필터 텍스트 상자에 텍스트를 입력합니다.
- 관련 공급자 계정 또는 조직으로 이동하려면 연결 이름을 선택합니다.
- 연결을 선택하여 (서비스 생성의 일환으로) 방금 설정한 연결의 핸드셰이크를 완료하거나 연결을 삭제합니다.

Auto Scaling 구성 페이지

Auto Scaling 구성 페이지에는 계정에서 설정한 Auto Scaling 구성이 나열됩니다. 몇 가지 매개변수를 구성하여 Auto Scaling 동작을 조정하고 나중에 하나 이상의 App Runner 서비스에 할당할 수 있는 다양한 구성으로 저장할 수 있습니다. 필터 텍스트 상자를 사용하여 목록의 범위를 좁힐 수 있습니다. Auto Scaling 구성에 대한 자세한 내용은 을 참조하십시오서비스의 Auto Scaling 관리.

Auto Scaling 구성 페이지로 이동하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 Auto Scaling 구성을 선택합니다.



여기서 할 수 있는 일들:

- 계정의 기존 Auto Scaling 구성 목록을 확인하십시오.
- 새 Auto Scaling 구성을 생성하거나 기존 구성을 수정하십시오.
- Auto Scaling 구성을 새로 생성하는 서비스의 기본값으로 설정합니다.

- 구성을 삭제합니다.
- 구성 이름을 선택하면 Auto Scaling 수정 패널로 이동하여 수정 내용을 관리할 수 있습니다.

App Runner 서비스 관리

이 장에서는 AWS App Runner 서비스를 관리하는 방법을 설명합니다. 이 장에서는 서비스 생성, 구성 및 삭제, 서비스에 새 애플리케이션 버전 배포, 서비스 일시 중지 및 재개를 통한 웹 서비스 가용성 제어 등 서비스 수명 주기를 관리하는 방법을 알아봅니다. 또한 연결 및 Auto Scaling과 같은 서비스의 다른 측면을 관리하는 방법도 알아봅니다.

주제

앱 러너 서비스 만들기

AWS App Runner 컨테이너 이미지 또는 소스 코드 리포지토리에서 자동으로 확장되는 실행 중인 웹 서비스로의 전환을 자동화합니다. 필요한 몇 가지 설정만 지정하여 App Runner를 소스 이미지나 코드로 안내합니다. App Runner는 필요한 경우 애플리케이션을 빌드하고, 컴퓨팅 리소스를 프로비저닝하고, 컴퓨팅 리소스에서 실행되도록 애플리케이션을 배포합니다.

서비스를 생성하면 App Runner가 서비스 리소스를 생성합니다. 경우에 따라 연결 리소스를 제공해야 할 수도 있습니다. App Runner 콘솔을 사용하는 경우 콘솔은 암시적으로 연결 리소스를 생성합니다. App Runner 리소스 유형에 대한 자세한 내용은 을 참조하십시오. the section called “앱 러너 리소스” 이러한 리소스 유형에는 각 계정과 관련된 할당량이 있습니다. AWS 리전자세한 정보는 the section called “앱 러너 리소스 할당량”을 참조하세요.

소스 유형 및 공급자에 따라 서비스를 만드는 절차에는 미묘한 차이가 있습니다. 이 항목에서는 상황에 적합한 방법을 따를 수 있도록 이러한 소스 유형을 만드는 여러 절차를 설명합니다. 코드 예제를 사용하여 기본 절차를 시작하는 방법은 을 참조하십시오. 시작하기

필수 조건

App Runner 서비스를 생성하기 전에 다음 작업을 완료해야 합니다.

- 의 설정 단계를 완료하세요. 설정
- 애플리케이션 소스가 준비되었는지 확인하세요. Bitbucket의 GitHub코드 리포지토리 또는 Amazon Elastic Container Registry (Amazon ECR) 의 컨테이너 이미지를 사용하여 앱 러너 서비스를 생성할 수 있습니다.

서비스 생성

이 섹션에서는 두 가지 App Runner 서비스 유형, 즉 소스 코드 기반과 컨테이너 이미지 기반의 생성 프로세스를 안내합니다.

> **Note**
>
> 서비스에 대한 아웃바운드 트래픽 VPC 커넥터를 생성하는 경우 이후 서비스 시작 프로세스에서 1회의 지연이 발생합니다. 새 서비스를 생성할 때 또는 이후에 서비스 업데이트를 통해 새 서비스에 대해 이 구성을 설정할 수 있습니다. 자세한 내용은 이 일회성 지연 가이드의 App Runner를 사용한 네트워킹 장을 참조하십시오.

코드 리포지토리에서 서비스 만들기

다음 섹션에서는 소스가 GitHub또는 Bitbucket의 코드 저장소인 경우 App Runner 서비스를 만드는 방법을 보여줍니다. 코드 리포지토리를 사용하는 경우 App Runner는 공급자 조직 또는 계정에 연결해야 합니다. 따라서 이러한 연결을 설정하는 데 도움이 필요합니다. App Runner 연결에 대한 자세한 내용은 을 참조하십시오the section called “연결”.

서비스를 만들면 App Runner는 애플리케이션 코드와 종속성이 포함된 Docker 이미지를 빌드합니다. 그런 다음 이 이미지의 컨테이너 인스턴스를 실행하는 서비스를 시작합니다.

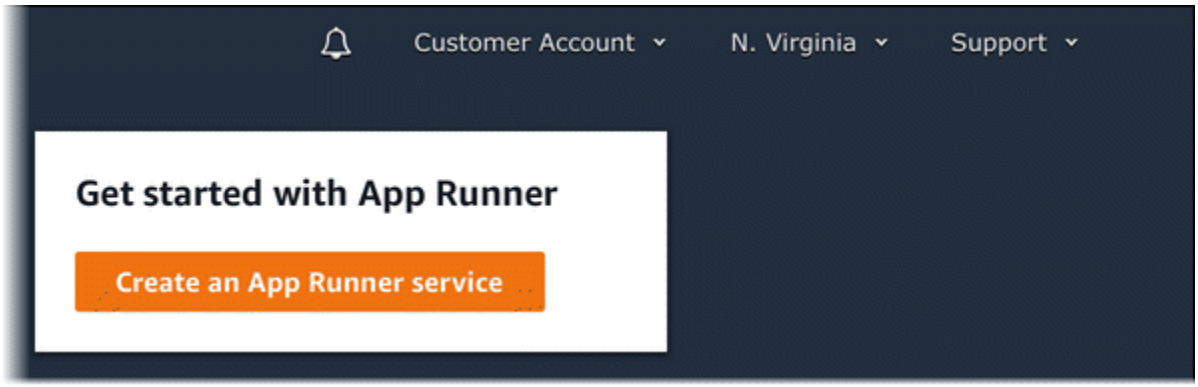
App Runner 콘솔을 사용하여 코드로 서비스 만들기

콘솔을 사용하여 App Runner 서비스를 만들려면

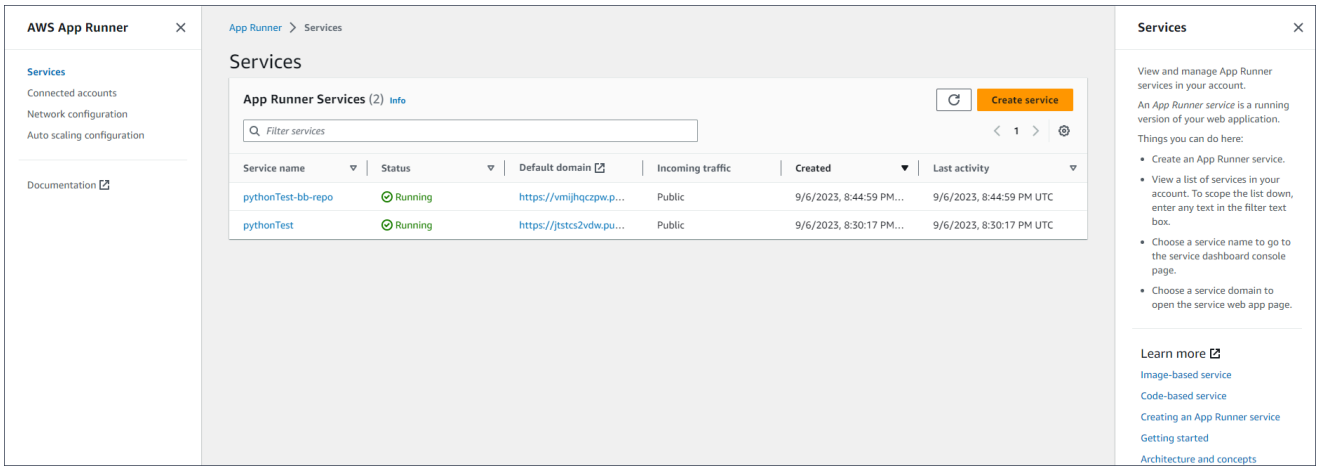
1. 소스 코드를 구성하세요.

   a. App Runner 콘솔을 열고 지역 목록에서 원하는 항목을 선택합니다. AWS 리전

b. 에 아직 App Runner 서비스가 AWS 계정 없는 경우 콘솔 홈페이지가 표시됩니다. 앱 러너 서비스 만들기를 선택합니다.



에 기존 서비스가 AWS 계정 있는 경우 서비스 목록이 있는 서비스 페이지가 표시됩니다. 서비스 생성을 선택합니다.



c. 소스 및 배포 페이지의 소스 섹션에서 리포지토리 유형으로 소스 코드 리포지토리를 선택합니다.

d. 제공자 유형을 선택합니다. 둘 중 하나를 GitHub선택하거나 Bitbucket을 선택합니다.

e. 그런 다음 이전에 사용한 적이 있는 공급자의 계정 또는 조직을 선택하거나 Add new (새로 추가) 를 선택합니다. 그런 다음 코드 리포지토리 자격 증명을 제공하고 연결할 계정 또는 조직을 선택하는 프로세스를 진행하세요.

f. 리포지토리의 경우 애플리케이션 코드가 들어 있는 리포지토리를 선택합니다.

g. Branch의 경우 배포하려는 브랜치를 선택합니다.

h. 소스 디렉터리의 경우 애플리케이션 코드 및 구성 파일이 저장되는 소스 리포지토리의 디렉터리를 입력합니다.

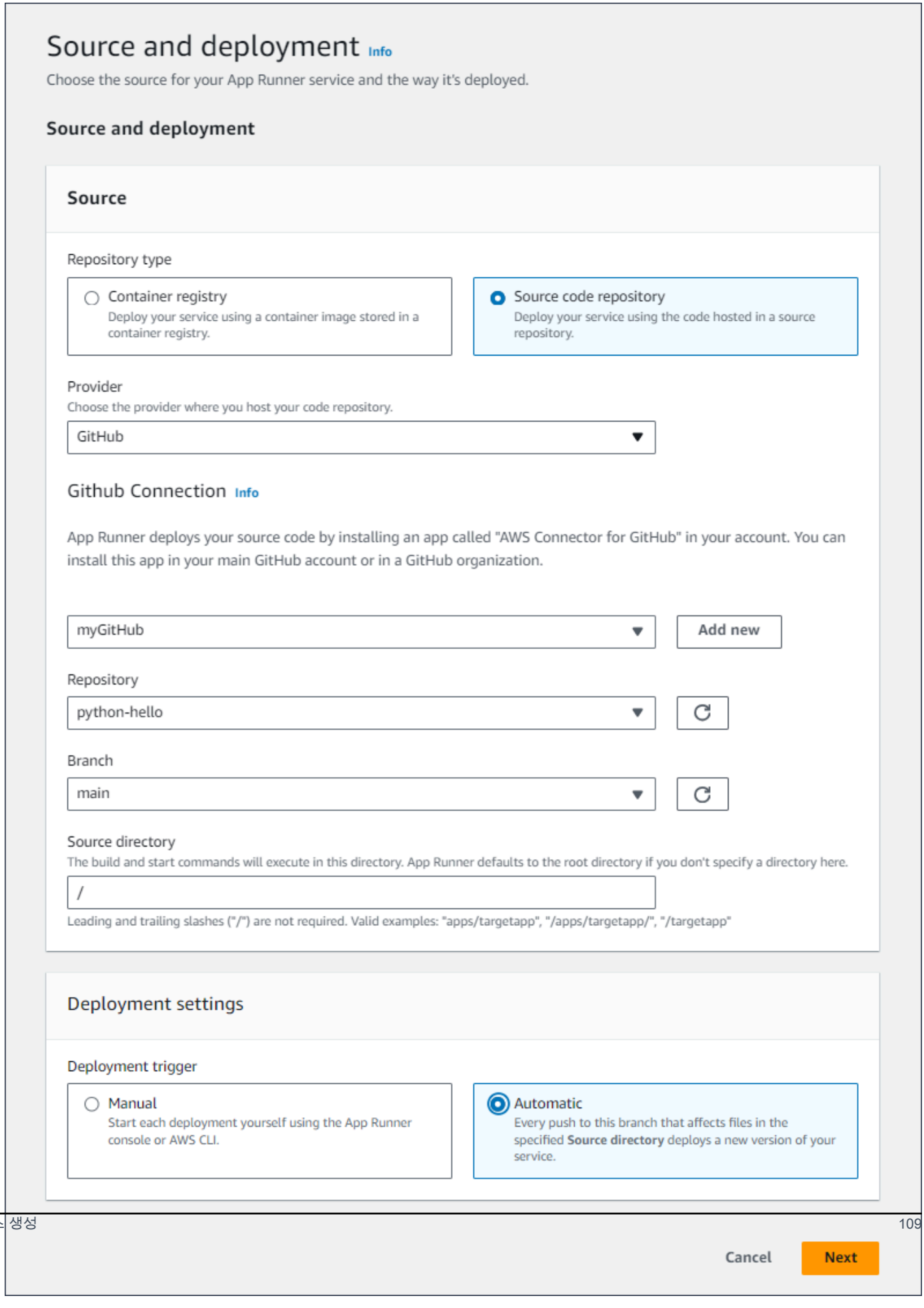
> **Note**
>
> build 및 start 명령은 지정한 소스 디렉토리에서 실행됩니다. App Runner는 루트부터 절대 경로로 처리합니다. 여기에 값을 지정하지 않으면 디렉터리는 기본적으로 리포지토리 루트가 됩니다.

2. 배포를 구성하세요.

   a. 배포 설정 섹션에서 수동 또는 자동을 선택합니다.

      배포 방법에 대한 자세한 내용은 을 참조하십시오the section called “배포 방법”.

   b. 다음을 선택합니다.


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. 애플리케이션 빌드를 구성합니다.

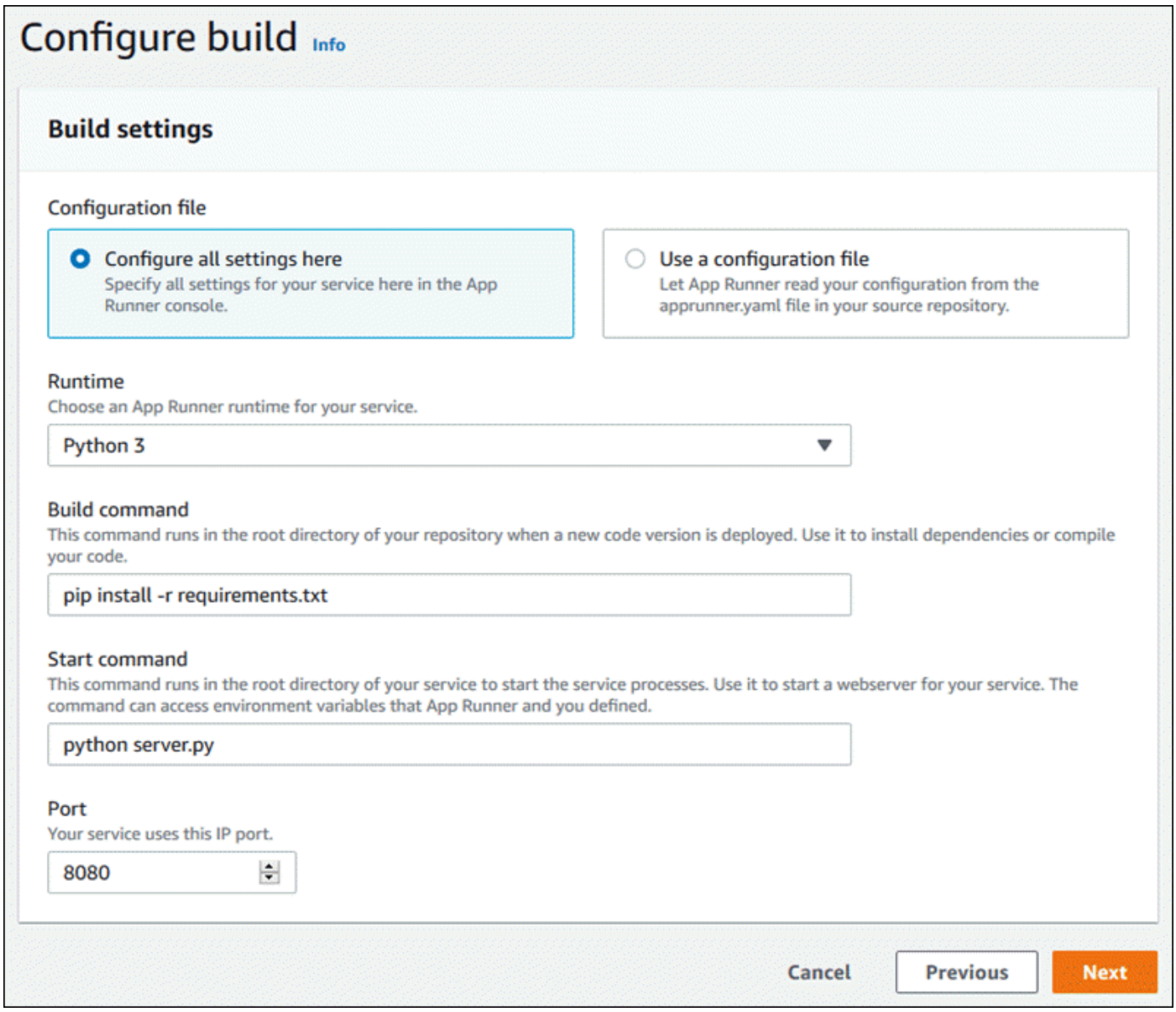
   a. 빌드 구성 페이지의 구성 파일에서 저장소에 App Runner 구성 파일이 없는 경우 여기에서 모든 설정 구성을 선택하고, 포함되어 있으면 구성 파일 사용을 선택합니다.

   > **Note**
   >
   > App Runner 구성 파일은 빌드 구성을 애플리케이션 소스의 일부로 유지 관리하는 방법입니다. 파일을 제공하면 App Runner는 파일에서 일부 값을 읽지만 콘솔에서 설정할 수 없게 합니다.

   b. 다음 빌드 설정을 제공하세요.

   - 런타임 - 애플리케이션의 특정 관리형 런타임을 선택합니다.
   - 빌드 명령 - 소스 코드에서 애플리케이션을 빌드하는 명령을 입력합니다. 이는 언어별 도구 또는 코드와 함께 제공되는 스크립트일 수 있습니다.
   - 시작 명령 - 웹 서비스를 시작하는 명령을 입력합니다.
   - 포트 - 웹 서비스가 수신하는 IP 포트를 입력합니다.

   c. 다음을 선택합니다.



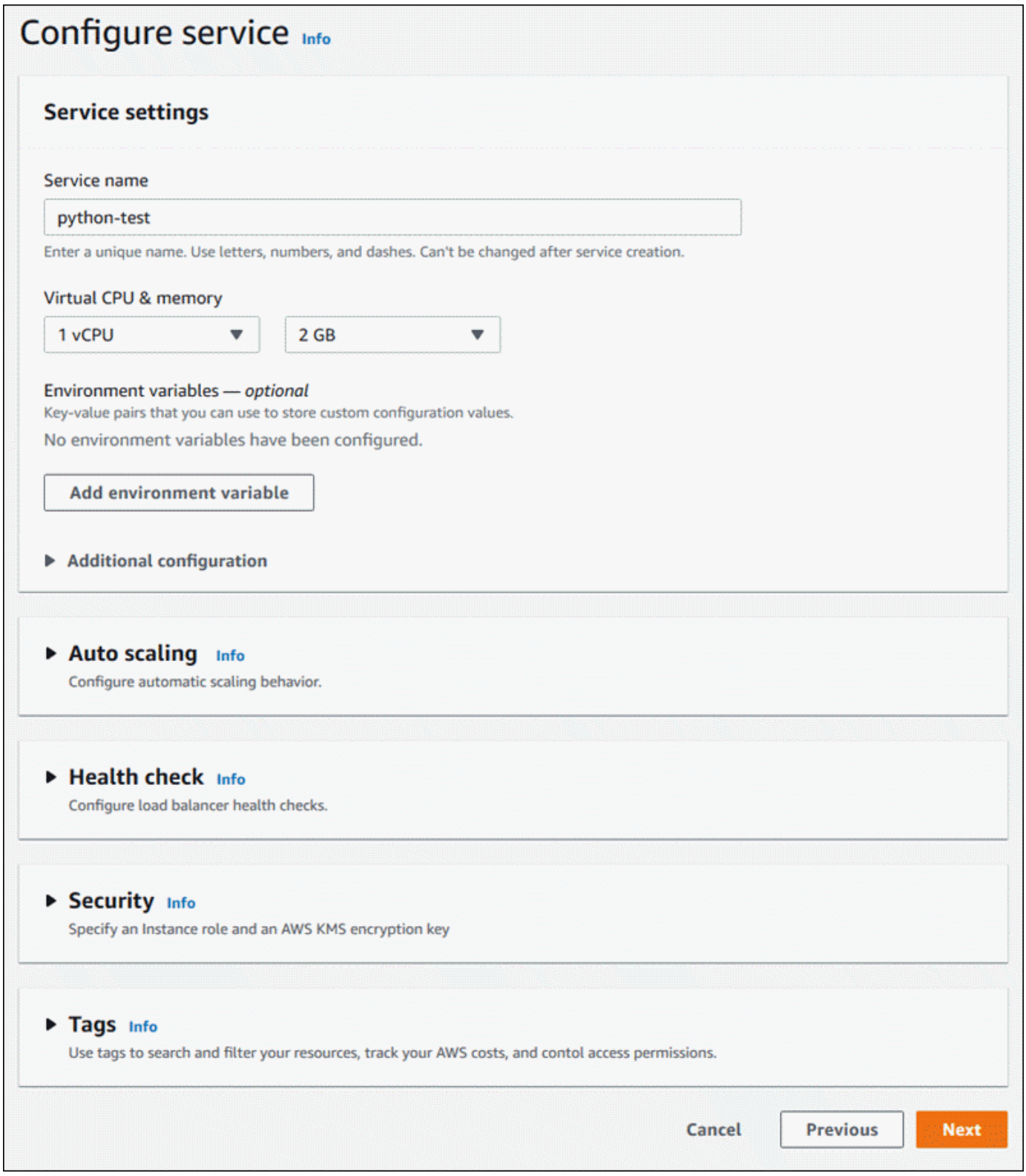
4. 서비스를 구성하세요.

   a. 서비스 구성 페이지의 서비스 설정 섹션에서 서비스 이름을 입력합니다.

   > **Note**
   >
   > 다른 모든 서비스 설정은 선택 사항이거나 콘솔에서 제공하는 기본값입니다.

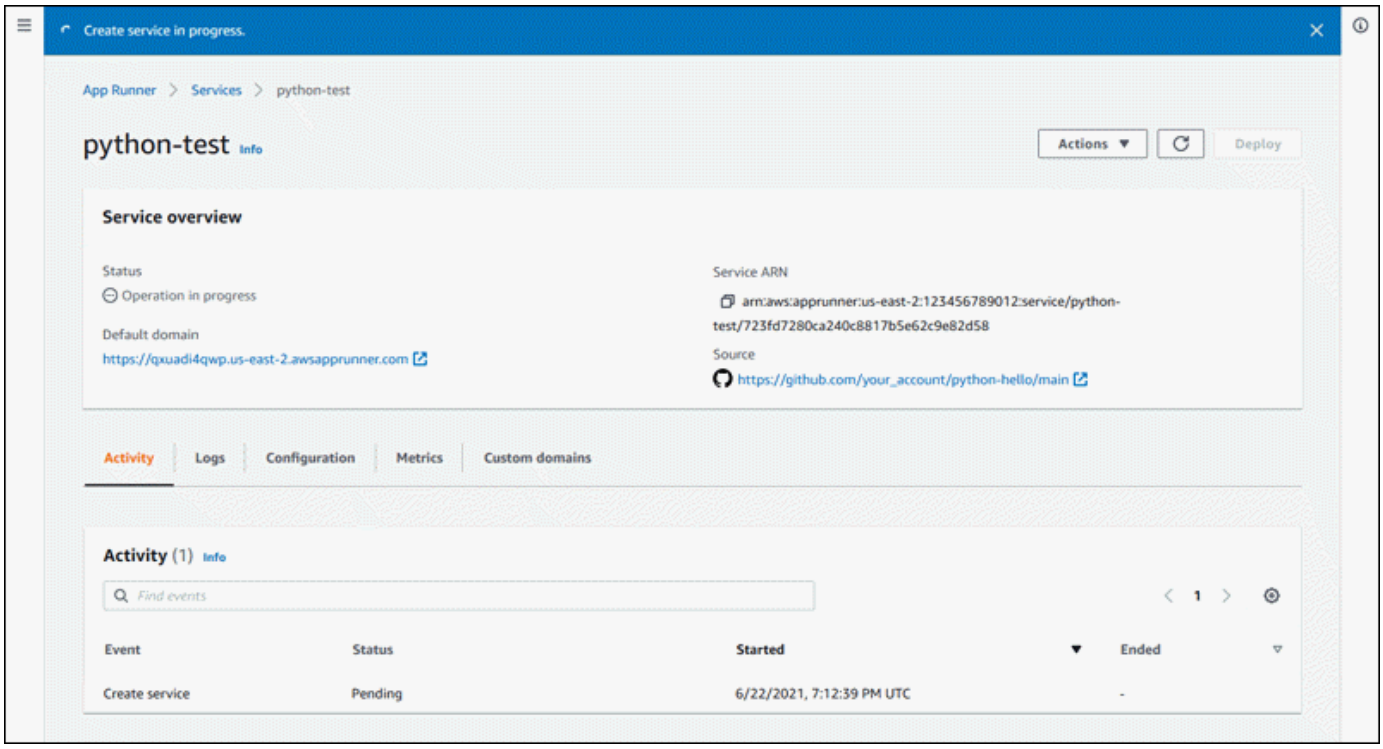
   b. 애플리케이션 요구 사항에 맞게 다른 설정을 변경하거나 추가할 수도 있습니다.

   c. 다음을 선택합니다.



5. 검토 및 생성 페이지에서 입력한 모든 세부 정보를 확인한 다음 생성 및 배포를 선택합니다.

결과: 서비스가 성공적으로 생성되면 콘솔에 새 서비스의 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



6. 서비스가 실행 중인지 확인하세요.

   a. 서비스 대시보드 페이지에서 서비스 상태가 Running (실행 중) 이 될 때까지 기다립니다.

   b. 기본 도메인 값을 선택합니다. 서비스 웹사이트의 URL입니다.

   c. 웹사이트를 사용하고 제대로 실행되고 있는지 확인하세요.

App Runner API를 사용하여 코드로 서비스를 생성하거나 AWS CLI

앱 러너 API를 사용하여 서비스를 생성하거나 API AWS CLI작업을 호출하십시오`CreateService`. 자세한 내용 및 예제는 을 참조하십시오 CreateService. 소스 코드 리포지토리 (GitHub 또는 Bitbucket) 의 특정 조직이나 계정을 사용하여 서비스를 처음 만드는 경우에는 를 CreateConnection호출하여 시작하십시오. 이렇게 하면 App Runner와 리포지토리 제공자의 조직 또는 계정이 연결됩니다. App Runner 연결에 대한 자세한 내용은 을 참조하십시오. the section called “연결”

호출 시 Service 객체가 표시된 상태에서 성공적인 응답이 `"Status": "CREATING"` 반환되면 서비스 생성이 시작됩니다.

예제 호출은 AWS App Runner API Reference에서 소스 코드 리포지토리 서비스 만들기를 참조하십시오.

Amazon ECR 이미지에서 서비스 생성

다음 섹션에서는 Amazon ECR에 저장된 컨테이너 이미지가 소스인 경우 App Runner 서비스를 생성하는 방법을 보여줍니다. 아마존 ECR은 다음과 같습니다. AWS 서비스따라서 Amazon ECR 이미지를 기반으로 서비스를 생성하려면 필요한 Amazon ECR 작업 권한이 포함된 액세스 역할을 App Runner에 제공해야 합니다.

> **Note**
>
> Amazon ECR 퍼블릭에 저장된 이미지는 공개적으로 사용할 수 있습니다. 따라서 Amazon ECR Public에 저장된 이미지는 액세스 역할이 필요하지 않습니다.

서비스가 생성되면 App Runner는 사용자가 제공한 이미지의 컨테이너 인스턴스를 실행하는 서비스를 시작합니다. 이 경우에는 빌드 단계가 없습니다.

자세한 정보는 이미지 기반 서비스을 참조하세요.

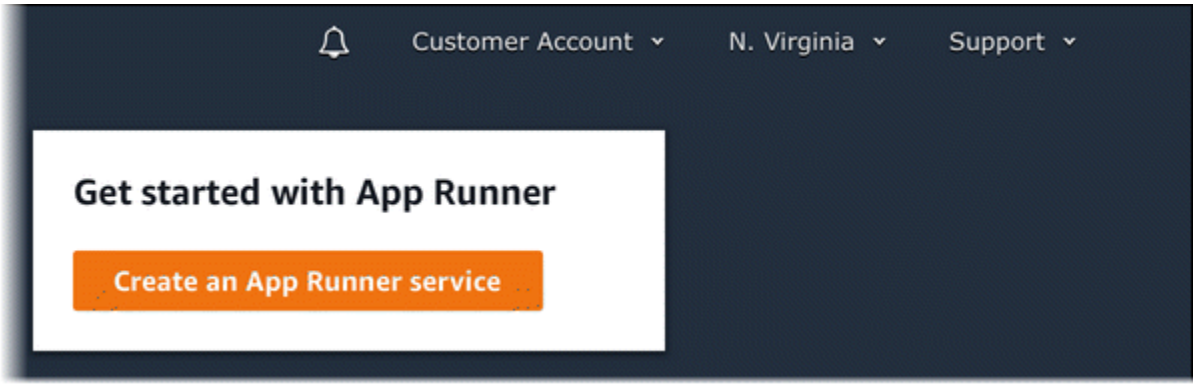
App Runner 콘솔을 사용하여 이미지에서 서비스 만들기

콘솔을 사용하여 App Runner 서비스를 만들려면

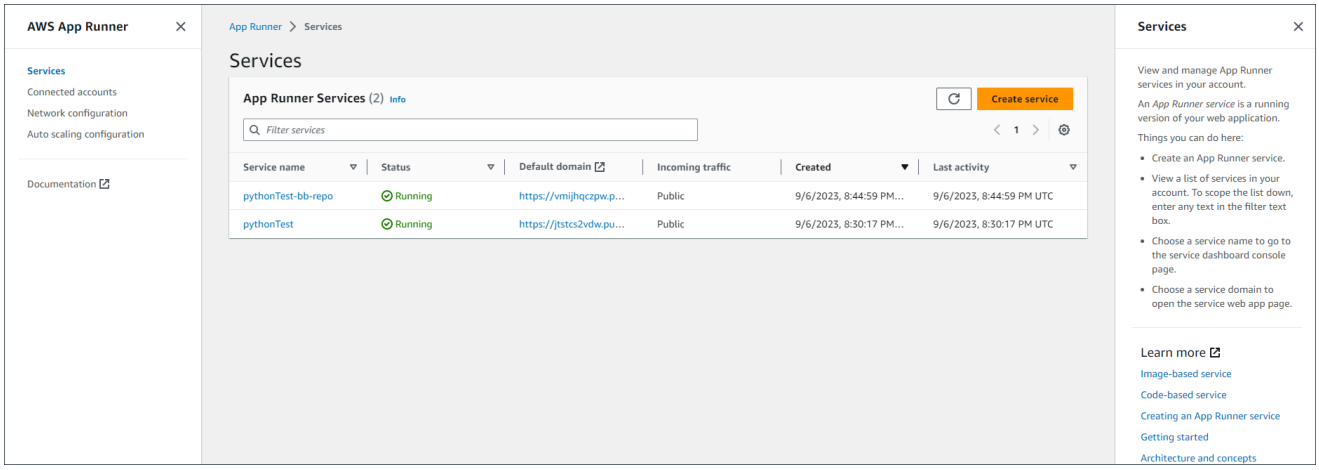
1. 소스 코드를 구성하세요.

   a. App Runner 콘솔을 열고 지역 목록에서 원하는 항목을 선택합니다. AWS 리전

   b. 에 아직 App Runner 서비스가 AWS 계정 없는 경우 콘솔 홈페이지가 표시됩니다. 앱 러너 서비스 만들기를 선택합니다.



   에 기존 서비스가 AWS 계정 있는 경우 서비스 목록이 있는 서비스 페이지가 표시됩니다. 서비스 생성을 선택합니다.



c. 소스 및 배포 페이지의 소스 섹션에서 리포지토리 유형으로 컨테이너 레지스트리를 선택합니다.

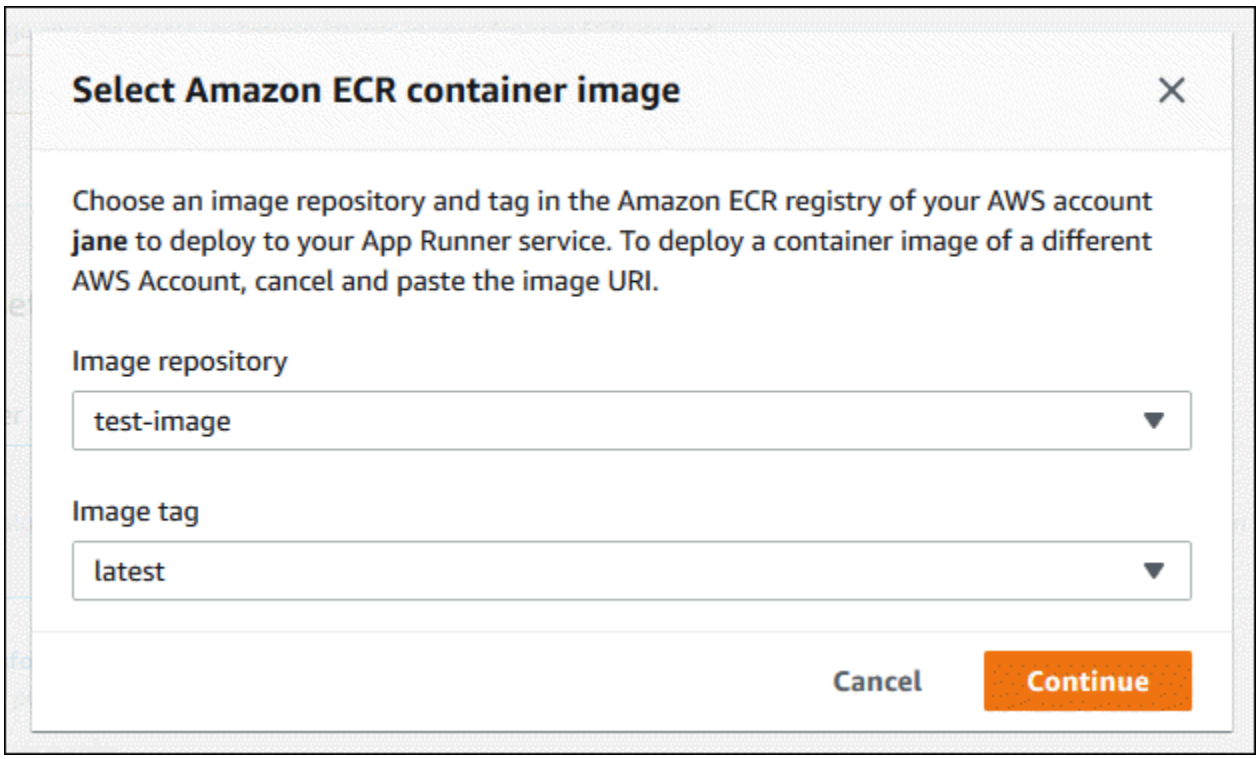
d. 공급자에서 이미지가 저장되어 있는 공급자를 선택합니다.

- 아마존 ECR — 아마존 ECR에 저장되는 프라이빗 이미지입니다.
- Amazon ECR 퍼블릭 — 공개적으로 읽을 수 있는 이미지로, Amazon ECR 퍼블릭에 저장됩니다.

e. 컨테이너 이미지 URI의 경우 찾아보기를 선택합니다.

f. Amazon ECR 컨테이너 이미지 선택 대화 상자의 이미지 리포지토리에서 이미지가 포함된 리포지토리를 선택합니다.

g. 이미지 태그의 경우 배포하려는 특정 이미지 태그 (예: latest) 를 선택한 다음 Continue를 선택합니다.



2. 배포를 구성하십시오.

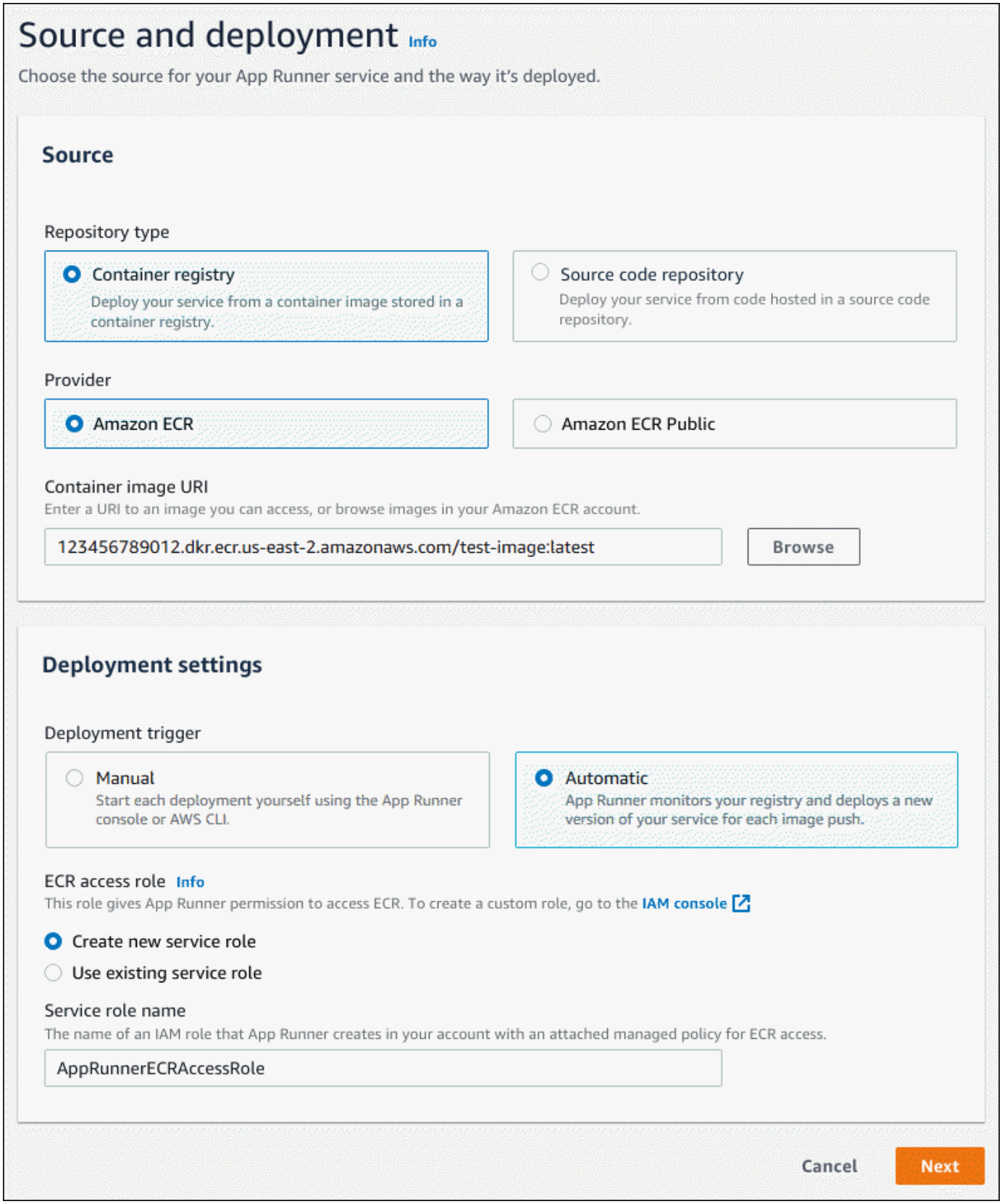
   a. 배포 설정 섹션에서 수동 또는 자동을 선택합니다.

   > **Note**
   >
   > App Runner는 Amazon ECR 퍼블릭 이미지 및 서비스가 속한 계정과 다른 AWS 계정에 속하는 Amazon ECR 리포지토리의 이미지에 대한 자동 배포를 지원하지 않습니다.

   배포 방법에 대한 자세한 내용은 을 참조하십시오. the section called “배포 방법”

   b. [Amazon ECR 공급자] ECR 액세스 역할의 경우 계정의 기존 서비스 역할을 선택하거나 새 역할을 생성하도록 선택합니다. 수동 배포를 사용하는 경우 배포 시 IAM 사용자 역할을 사용하도록 선택할 수도 있습니다.

   c. 다음을 선택합니다.


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

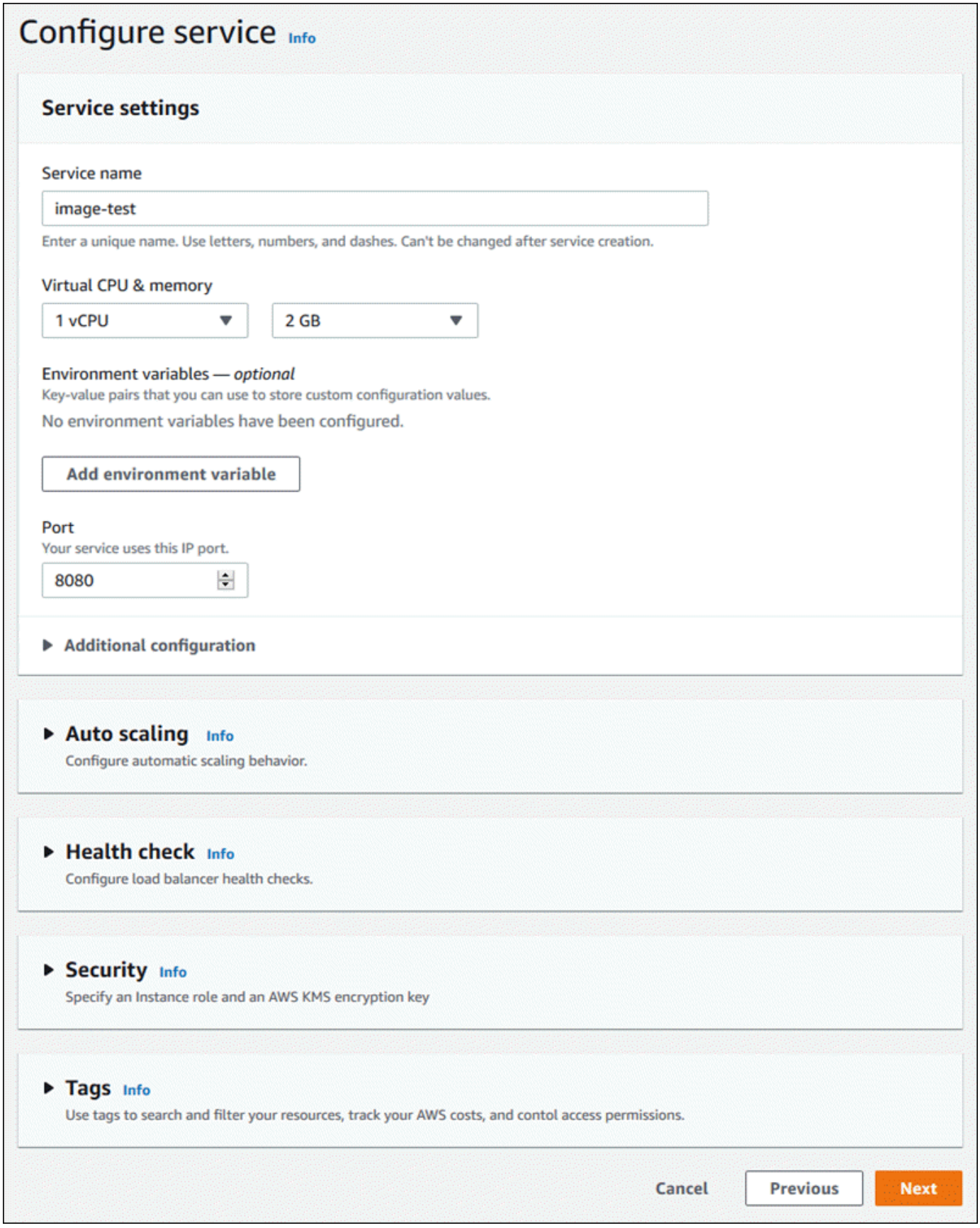
3. 서비스를 구성하세요.

   a. 서비스 구성 페이지의 서비스 설정 섹션에서 서비스 웹 사이트에서 수신하는 서비스 이름과 IP 포트를 입력합니다.

   > **Note**
   >
   > 다른 모든 서비스 설정은 선택 사항이거나 콘솔에서 제공하는 기본값입니다.

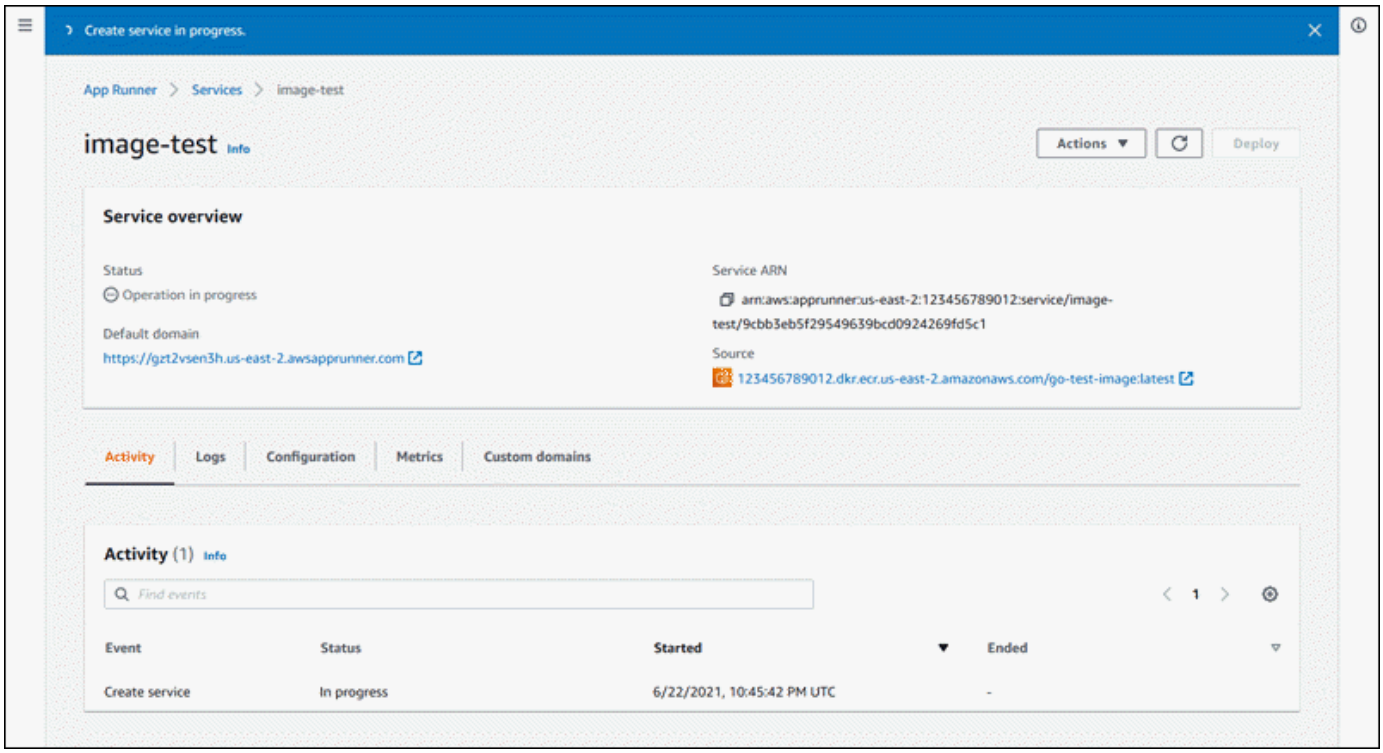
   b. (선택 사항) 애플리케이션의 요구 사항에 맞게 다른 설정을 변경하거나 추가합니다.

   c. 다음을 선택합니다.


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. 검토 및 생성 페이지에서 입력한 모든 세부 정보를 확인한 다음 생성 및 배포를 선택합니다.

   결과: 서비스가 성공적으로 생성되면 콘솔에 새 서비스의 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



5. 서비스가 실행 중인지 확인하세요.

   a. 서비스 대시보드 페이지에서 서비스 상태가 Running (실행 중) 이 될 때까지 기다립니다.

   b. 기본 도메인 값을 선택합니다. 서비스 웹사이트의 URL입니다.

   c. 웹사이트를 사용하고 제대로 실행되고 있는지 확인하세요.

App Runner API를 사용하여 이미지에서 서비스를 생성하거나 AWS CLI

앱 러너 API를 사용하여 서비스를 생성하거나 API AWS CLI작업을 호출하십시오 CreateService.

호출 시 Service 객체가 표시된 `"Status": "CREATING"` 성공적인 응답이 반환되면 서비스 생성이 시작됩니다.

예제 호출은 AWS App Runner API Reference에서 소스 이미지 리포지토리 서비스 만들기를 참조하십시오.

실패한 앱 러너 서비스 재구축

App Runner 서비스를 생성할 때 생성 실패 오류가 발생하는 경우 다음 중 하나를 수행할 수 있습니다.

- 의 단계에 따라 오류의 원인을 식별하십시오. the section called “서비스를 생성하지 못했습니다.”
- 소스 또는 구성에서 오류를 발견하면 필요에 따라 변경한 다음 서비스를 다시 빌드하세요.
- App Runner의 일시적인 문제로 서비스에 장애가 발생한 경우 소스나 구성을 변경하지 않고 장애가 발생한 서비스를 다시 빌드하세요.

앱 러너 콘솔 또는 앱 러너 API를 통해 실패한 서비스를 다시 빌드할 수 있습니다. AWS CLI

앱 러너 콘솔을 사용하여 실패한 앱 러너 서비스를 재빌드합니다.

Rebuild with updates

다양한 이유로 서비스 생성이 실패할 수 있습니다. 이런 경우에는 서비스를 재구축하기 전에 문제의 근본 원인을 파악하고 해결하는 것이 중요합니다. 자세한 정보는 the section called “서비스를 생성하지 못했습니다.”을 참조하세요.

실패한 서비스를 업데이트로 다시 빌드하려면

1. 서비스 페이지의 구성 탭으로 이동하여 편집을 선택합니다.

   이 페이지에는 모든 업데이트 목록이 표시된 요약 패널이 열립니다.
2. 필요한 사항을 변경하고 요약 패널에서 검토하십시오.
3. [저장 후 다시 작성] 을 선택합니다.

   서비스 페이지의 로그 탭에서 진행 상황을 모니터링할 수 있습니다.

Rebuild without updates

일시적인 문제로 인해 서비스 생성이 실패하는 경우 소스 또는 구성 설정을 수정하지 않고 서비스를 다시 빌드할 수 있습니다.

실패한 서비스를 업데이트 없이 다시 빌드하는 방법

- 서비스 페이지 오른쪽 상단에서 Rebuild를 선택합니다.

  서비스 페이지의 로그 탭에서 진행 상황을 모니터링할 수 있습니다.

- 서비스가 다시 생성되지 않는 경우 의 문제 해결 지침을 따르세요the section called “서비스를 생성하지 못했습니다.”. 필요에 따라 변경한 다음 서비스를 다시 빌드하세요.

앱 러너 API를 사용하여 실패한 앱 러너 서비스를 재빌드하거나 AWS CLI

Rebuild with updates

실패한 서비스를 다시 빌드하려면:

1. 의 the section called “서비스를 생성하지 못했습니다.” 지침에 따라 오류의 원인을 찾으십시오.
2. 소스 리포지토리의 브랜치나 이미지 또는 오류를 일으킨 구성을 필요에 따라 변경합니다.
3. 새 소스 코드 리포지토리 또는 소스 이미지 리포지토리 매개변수로 UpdateServiceAPI 작업을 호출하여 다시 빌드합니다. App Runner는 소스 코드 리포지토리에서 최신 커밋을 검색합니다.

Example 업데이트를 통한 재구축

다음 예시에서는 이미지 기반 서비스의 소스 구성을 업데이트하고 있습니다. 의 `Port` 값이 로 변경됩니다. `80`

이미지 기반 App Runner 서비스를 위한 `input.json` 파일 업데이트

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

API 작업 호출. `UpdateService`

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

Rebuild without updates

App Runner API 또는 를 사용하여 실패한 서비스를 다시 빌드하려면 서비스의 소스 또는 AWS CLI 구성을 변경하지 않고 UpdateServiceAPI 작업을 호출하십시오. App Runner의 일시적인 문제로 인해 서비스 생성이 실패한 경우에만 업데이트 없이 다시 빌드하도록 선택하세요.

App Runner에 새 애플리케이션 버전 배포

에서 AWS App Runner서비스를 생성할 때 애플리케이션 소스 (컨테이너 이미지 또는 소스 리포지토리) 를 구성합니다. App Runner는 서비스를 실행하기 위한 리소스를 프로비저닝하고 해당 리소스에 애플리케이션을 배포합니다.

이 항목에서는 새 버전이 출시될 때 애플리케이션 소스를 App Runner 서비스에 재배포하는 방법을 설명합니다. 이미지 리포지토리의 새 이미지 버전이거나 코드 리포지토리의 새 커밋일 수 있습니다. App Runner는 서비스에 배포하는 두 가지 방법, 즉 자동 및 수동을 제공합니다.

배포 방법

App Runner는 애플리케이션 배포가 시작되는 방식을 제어할 수 있는 다음과 같은 방법을 제공합니다.

자동 배포

서비스에 대한 지속적 통합 및 배포 (CI/CD) 동작을 원하는 경우 자동 배포를 사용하세요. App Runner는 이미지 또는 코드 저장소의 변경 사항을 모니터링합니다.

이미지 리포지토리 — 새 이미지 버전을 이미지 리포지토리에 푸시하거나 코드 리포지토리에 새 커밋을 푸시할 때마다 App Runner는 추가 조치 없이 이를 서비스에 자동으로 배포합니다.

코드 리포지토리 — 소스 디렉터리를 변경하는 코드 리포지토리에 새 커밋을 푸시할 때마다 App Runner는 전체 리포지토리를 배포합니다. 소스 디렉터리의 변경 사항만 자동 배포를 트리거하므로 소스 디렉터리 위치가 자동 배포의 범위에 어떤 영향을 미치는지 이해하는 것이 중요합니다.

- 최상위 디렉터리 (리포지토리 루트) — 서비스를 생성할 때 소스 디렉터리에 설정되는 기본값입니다. 소스 디렉터리가 이 값으로 설정되면 전체 리포지토리가 소스 디렉터리 내에 있음을 의미합니다. 따라서 이 경우 소스 리포지토리에 푸시하는 모든 커밋이 배포를 트리거합니다.
- 리포지토리 루트가 아닌 모든 디렉터리 경로 (기본값 아님) - 소스 디렉터리 내에서 푸시된 변경 사항만 자동 배포를 트리거하므로 원본 디렉터리에 없는 변경 내용을 리포지토리에 푸시해도 자동 배포가 트리거되지 않습니다. 따라서 푸시한 변경 내용을 원본 디렉터리 외부로 배포하려면 수동 배포를 사용해야 합니다.

> **Note**
>
> App Runner는 Amazon ECR 퍼블릭 이미지 및 서비스가 속한 계정과 다른 AWS 계정에 속하는 Amazon ECR 리포지토리의 이미지에 대한 자동 배포를 지원하지 않습니다.

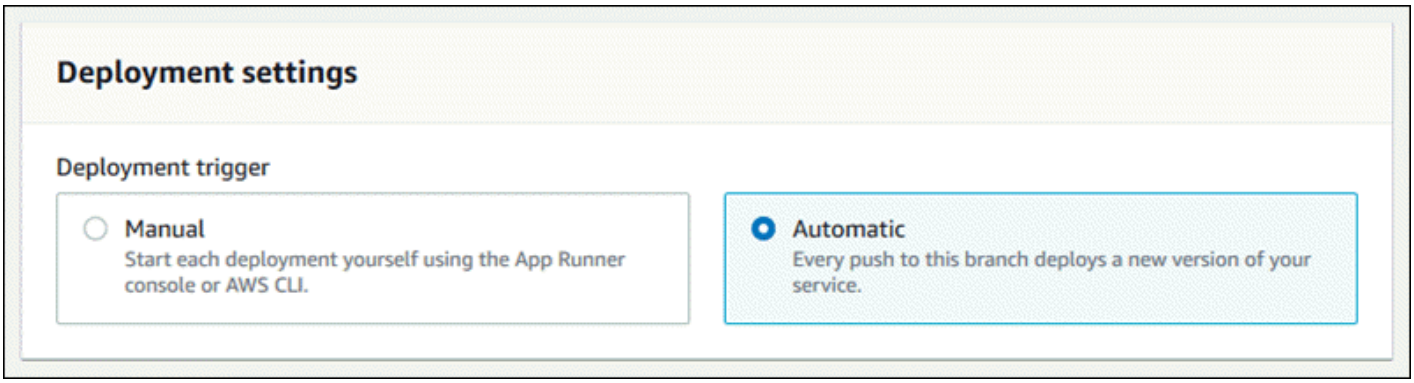
수동 배포

서비스에 대한 각 배포를 명시적으로 시작하려는 경우 수동 배포를 사용하세요. 서비스용으로 구성한 리포지토리에 배포하려는 새 버전이 있는 경우 배포를 시작합니다. 자세한 정보는 the section called “수동 배포”을 참조하세요.

> **Note**
>
> 수동 배포를 실행하면 App Runner는 전체 리포지토리의 소스를 배포합니다.

다음과 같은 방법으로 서비스의 배포 방법을 구성할 수 있습니다.

- 콘솔 - 새로 만들려는 서비스 또는 기존 서비스의 경우 원본 및 배포 구성 페이지의 배포 설정 섹션에서 수동 또는 자동을 선택합니다.



- API 또는 AWS CLI— or UpdateService작업을 호출할 때 CreateServiceSourceConfiguration매개 변수의 `AutoDeploymentsEnabled` 멤버를 `False` 수동 배포 또는 자동 `True` 배포용으로 설정합니다.

> **자동 배포와 수동 배포 비교**
>
> 자동 배포와 수동 배포 모두 동일한 결과를 산출합니다. 즉, 두 방법 모두 전체 리포지토리를 배포합니다.
> 두 메서드의 차이점은 트리거 메커니즘입니다.
>
> - 수동 배포는 콘솔에서의 배포, 에 대한 호출 또는 App Runner API에 대한 호출을 통해 트리거됩니다. AWS CLI다음 수동 배포 섹션에서는 이러한 절차에 대해 설명합니다.
> - 소스 디렉터리의 내용이 변경되면 자동 배포가 트리거됩니다.

수동 배포

수동 배포의 경우 서비스에 대한 각 배포를 명시적으로 시작해야 합니다. 새 버전의 애플리케이션 이미지 또는 코드를 배포할 준비가 되면 다음 섹션을 참조하여 콘솔과 API를 사용하여 배포를 수행하는 방법을 알아볼 수 있습니다.

> **Note**
>
> 수동 배포를 실행하면 App Runner는 전체 리포지토리의 소스를 배포합니다.

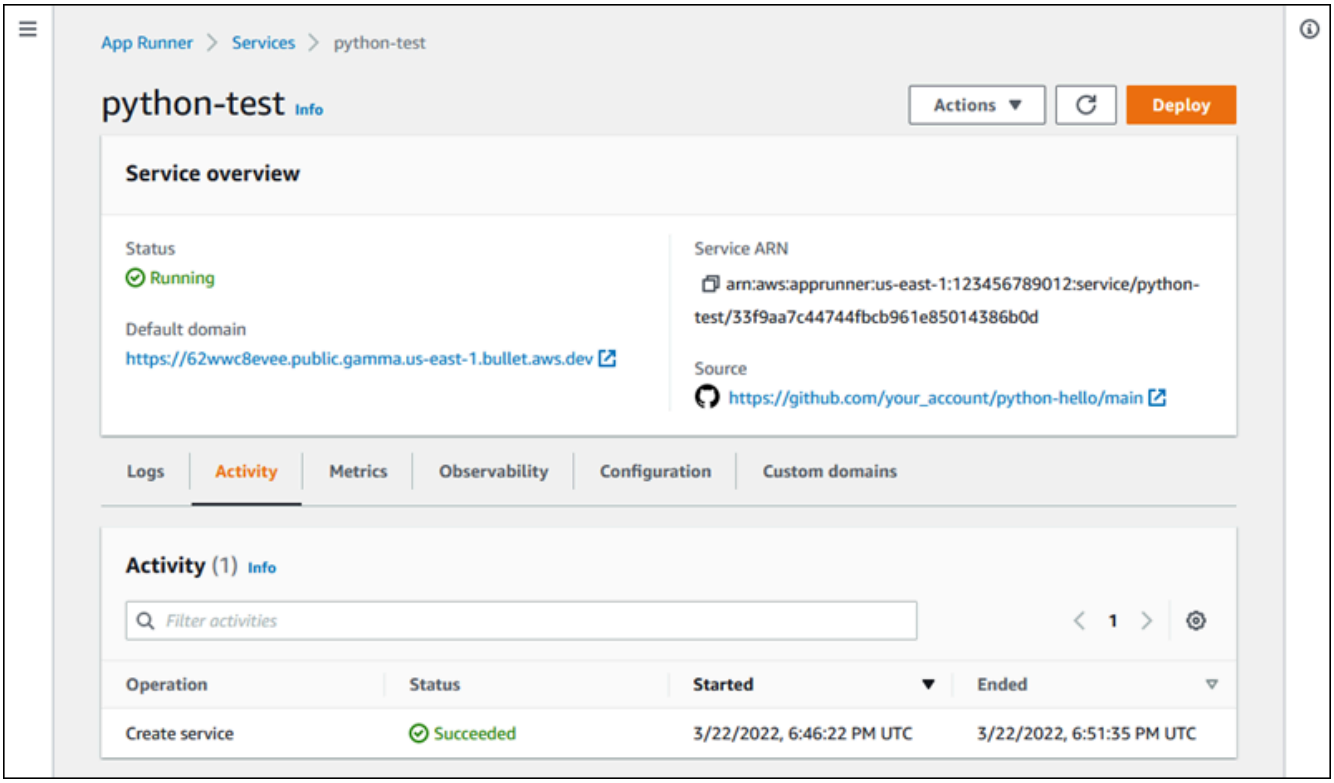
다음 방법 중 하나를 사용하여 애플리케이션 버전을 배포하십시오.

App Runner console

App Runner 콘솔을 사용하여 배포하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 배포를 선택합니다.

   결과: 새 버전 배포가 시작됩니다. 서비스 대시보드 페이지에서 서비스 상태가 작업 진행 중으로 변경됩니다.

4. 배포가 종료될 때까지 기다리세요. 서비스 대시보드 페이지에서 서비스 상태가 다시 Running (실행 중) 으로 변경되어야 합니다.

5. 배포가 성공했는지 확인하려면 서비스 대시보드 페이지에서 기본 도메인 값 (서비스 웹 사이트 URL) 을 선택합니다. 웹 애플리케이션을 검사하거나 웹 애플리케이션과 상호 작용하고 버전 변경을 확인하세요.

> **Note**
>
> 앱 러너 애플리케이션의 보안을 강화하기 위해*.awsapprunner.com 도메인은 공개 접미사 목록 (PSL) 에 등록되어 있습니다. 보안 강화를 위해 App Runner 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도 (CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

App Runner API or AWS CLI

앱 러너 API 또는 를 사용하여 배포하려면 API 작업을 AWS CLI호출하십시오. StartDeployment 전달해야 할 유일한 파라미터는 서비스 ARN입니다. 서비스를 생성할 때 애플리케이션 소스 위치를 이미 구성했으므로 App Runner에서 새 버전을 찾을 수 있습니다. 호출에서 성공적인 응답이 반환되면 배포가 시작됩니다.

앱 러너 서비스 구성

AWS App Runner 서비스를 생성할 때 다양한 구성 값을 설정합니다. 서비스를 만든 후 이러한 구성 설정 중 일부를 변경할 수 있습니다. 다른 설정은 서비스를 만드는 동안에만 적용할 수 있으며 이후에는 변경할 수 없습니다. 이 항목에서는 App Runner API, App Runner 콘솔 및 App Runner 구성 파일을 사용하여 서비스를 구성하는 방법에 대해 설명합니다.

주제

- 앱 러너 API를 사용하여 서비스를 구성하거나 AWS CLI
- App Runner 콘솔을 사용하여 서비스를 구성하십시오.
- App Runner 구성 파일을 사용하여 서비스를 구성하십시오.
- 서비스의 옵저버빌리티 구성
- 공유 가능한 리소스를 사용하여 서비스 설정을 구성합니다.
- 서비스 상태 점검 구성

앱 러너 API를 사용하여 서비스를 구성하거나 AWS CLI

API는 서비스 생성 후 변경할 수 있는 설정을 정의합니다. 다음 목록은 관련 작업, 유형 및 제한 사항에 대해 설명합니다.

- UpdateServiceaction — 생성 후 일부 구성 설정을 업데이트하기 위해 호출할 수 있습니다.
  - 업데이트 가능 - `SourceConfiguration`, `InstanceConfiguration`, 및 `HealthCheckConfiguration` 매개변수에서 설정을 업데이트할 수 있습니다. 하지만 `SourceConfiguration` 에서는 소스 유형을 코드에서 이미지로 또는 그 반대로 전환할 수 없습니다. 서비스를 생성할 때 제공한 것과 동일한 리포지토리 매개변수를 제공해야 합니다. 또는 둘 중 하나입니다. `CodeRepository` `ImageRepository`

    서비스와 관련된 개별 구성 리소스의 다음 ARN을 업데이트할 수도 있습니다.

    - `AutoScalingConfigurationArn`
    - `VpcConnectorArn`
- 업데이트할 수 없음 - CreateService작업에서 사용할 수 있는 *`ServiceName`* 및 *`EncryptionConfiguration`* 매개 변수는 변경할 수 없습니다. 생성한 후에는 변경할 수 없습니다. UpdateService액션에는 이러한 매개변수가 포함되지 않습니다.
- API 대 파일 — CodeConfiguration유형의 `ConfigurationSource` 매개 변수 (소스 코드 리포지토리의 `SourceConfiguration` 일부로 사용됨) 를 로 설정할 수 있습니다. `Repository` 이 경우 App Runner는 의 구성 설정을 무시하고 `CodeConfigurationValues` 저장소의 구성 파일에서 이러한 설정을 읽습니다. `ConfigurationSource`로 `API` 설정하면 App Runner는 API 호출에서 모든 구성 설정을 가져오고 구성 파일이 있더라도 해당 구성 파일을 무시합니다.

- TagResource조치 — 서비스를 생성한 후 호출하여 서비스에 태그를 추가하거나 기존 태그의 값을 업데이트할 수 있습니다.
- UntagResource조치 — 서비스를 생성한 후 호출하여 서비스에서 태그를 제거할 수 있습니다.

> **Note**
>
> 서비스에 대한 아웃바운드 트래픽 VPC 커넥터를 생성하는 경우 이후 서비스 시작 프로세스에서 1회의 지연이 발생합니다. 새 서비스를 생성할 때 또는 이후에 서비스 업데이트를 통해 새 서비스에 대해 이 구성을 설정할 수 있습니다. 자세한 내용은 이 일회성 지연 가이드의 App Runner를 사용한 네트워킹 장을 참조하십시오.

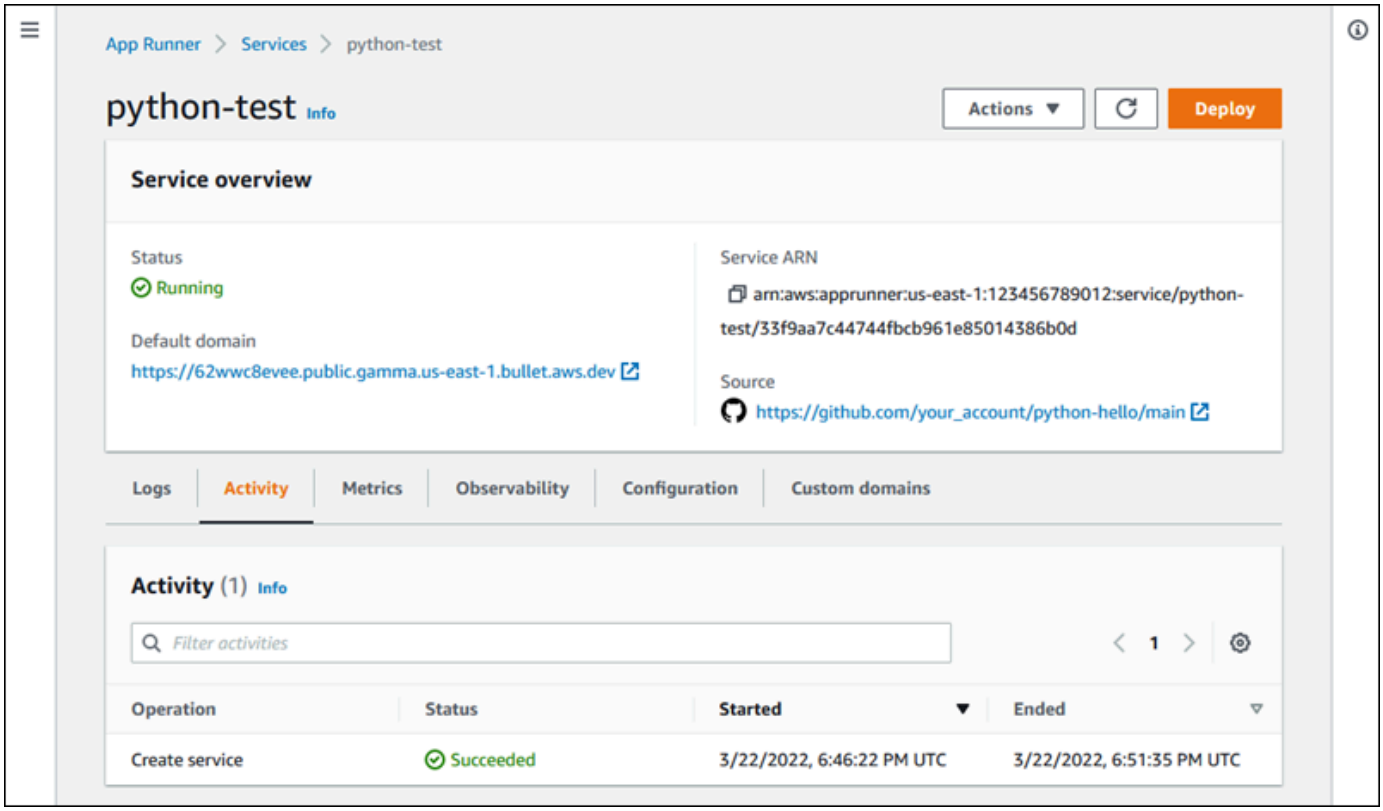
App Runner 콘솔을 사용하여 서비스를 구성하십시오.

콘솔은 App Runner API를 사용하여 구성 업데이트를 적용합니다. 이전 섹션에서 정의한 대로 API가 적용하는 업데이트 규칙에 따라 콘솔을 사용하여 구성할 수 있는 항목이 결정됩니다. 서비스 생성 중에 사용할 수 있었던 일부 설정은 나중에 수정할 수 없습니다. 또한 구성 파일을 사용하기로 결정하면 콘솔에 추가 설정이 숨겨지고 App Runner는 파일에서 해당 설정을 읽습니다.

서비스를 구성하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 구성 탭을 선택합니다.

   결과: 콘솔은 서비스의 현재 구성 설정을 소스 및 배포, 빌드 구성, 서비스 구성 등 여러 섹션에 표시합니다.
4. 모든 카테고리의 설정을 업데이트하려면 편집을 선택합니다.
5. 구성 편집 페이지에서 원하는 대로 변경한 다음 변경 내용 저장을 선택합니다.

> **Note**
>
> 서비스에 대한 아웃바운드 트래픽 VPC 커넥터를 생성하는 경우 이후 서비스 시작 프로세스에서 1회의 지연이 발생합니다. 새 서비스를 생성할 때 또는 이후에 서비스 업데이트를 통해 새 서비스에 대해 이 구성을 설정할 수 있습니다. 자세한 내용은 이 일회성 지연 가이드의 App Runner를 사용한 네트워킹 장을 참조하십시오.

App Runner 구성 파일을 사용하여 서비스를 구성하십시오.

App Runner 서비스를 만들거나 업데이트할 때 소스 저장소의 일부로 제공하는 구성 파일에서 일부 구성 설정을 읽도록 App Runner에 지시할 수 있습니다. 이렇게 하면 소스 제어 하에 있는 소스 코드와 관

련된 설정을 코드 자체와 함께 관리할 수 있습니다. 구성 파일은 콘솔이나 API로는 설정할 수 없는 특정 고급 설정도 제공합니다. 자세한 정보는 앱 러너 구성 파일을 참조하세요.

> **Note**
>
> 서비스에 대한 아웃바운드 트래픽 VPC 커넥터를 생성하는 경우 이후 서비스 시작 프로세스에서 1회의 지연이 발생합니다. 새 서비스를 생성할 때 또는 이후에 서비스 업데이트를 통해 새 서비스에 대해 이 구성을 설정할 수 있습니다. 자세한 내용은 이 일회성 지연 가이드의 App Runner를 사용한 네트워킹 장을 참조하십시오.

서비스의 옵저버빌리티 구성

AWS App Runner 여러 서비스와 통합되어 App Runner AWS 서비스를 위한 광범위한 옵저버빌리티 도구 모음을 제공합니다. 자세한 정보는 관찰성을 참조하세요.

App Runner는 라는 공유 가능한 리소스를 사용하여 일부 옵저버빌리티 기능을 활성화하고 동작을 구성할 수 있도록 지원합니다. ObservabilityConfiguration 서비스를 만들거나 업데이트할 때 관찰 가능성 구성 리소스를 제공할 수 있습니다. 새 App Runner 서비스를 만들면 App Runner 콘솔이 자동으로 생성합니다. 옵저버빌리티 구성을 제공하는 것은 선택 사항입니다. 제공하지 않는 경우 App Runner는 기본 옵저버빌리티 구성을 제공합니다.

여러 App Runner 서비스에서 단일 옵저버빌리티 구성을 공유하여 동일한 옵저버빌리티 동작을 갖도록 할 수 있습니다. 자세한 정보는 the section called “구성 리소스”을 참조하세요.

옵저버빌리티 구성을 사용하여 다음과 같은 옵저버빌리티 기능을 구성할 수 있습니다.

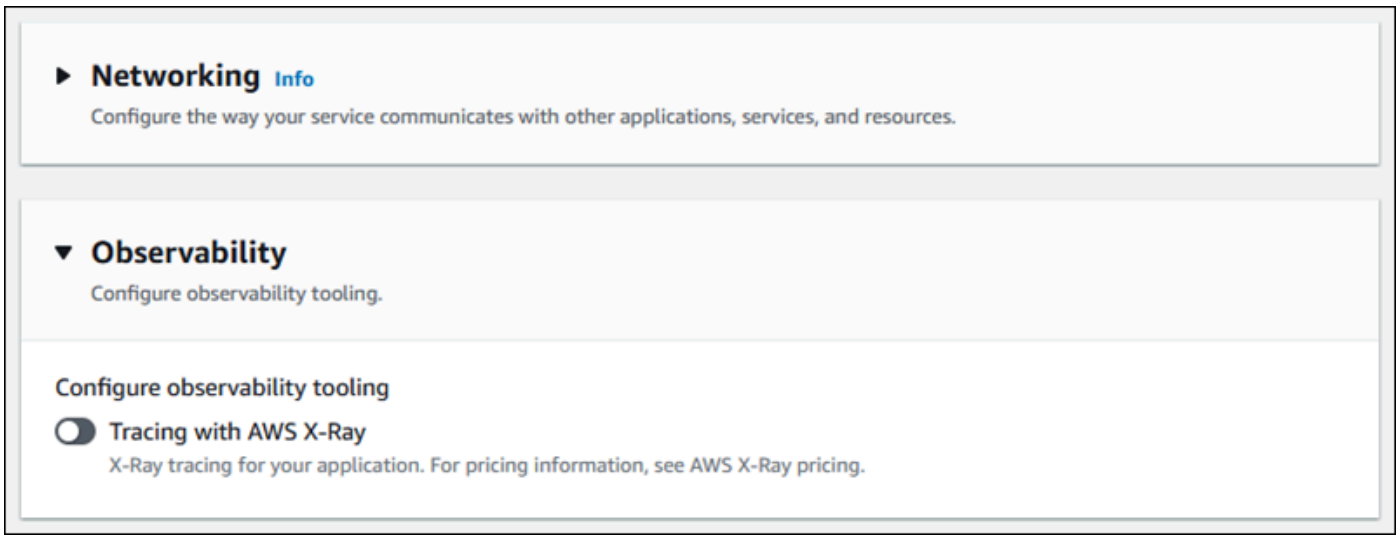
- 추적 구성 - 애플리케이션이 처리하는 요청과 애플리케이션에서 보내는 다운스트림 호출을 추적하기 위한 설정입니다. 추적에 대한 자세한 내용은 the section called “트레이싱 (X-Ray)”의 내용을 참조하세요.

옵저버빌리티 관리

다음 방법 중 하나를 사용하여 App Runner 서비스의 옵저버빌리티를 관리하세요.

App Runner console

App Runner 콘솔을 사용하여 서비스를 만들거나 나중에 구성을 업데이트할 때 서비스에 대한 관찰 기능을 구성할 수 있습니다. 콘솔 페이지에서 Observability 구성 섹션을 찾아보세요.



App Runner API or AWS CLI

CreateService또는 UpdateServiceApp Runner API 작업을 호출할 때 `ObservabilityConfiguration` 파라미터 객체를 사용하여 옵저버빌리티 기능을 활성화하고 서비스의 옵저버빌리티 구성 리소스를 지정할 수 있습니다.

다음 App Runner API 작업을 사용하여 옵저버빌리티 구성 리소스를 관리하세요.

- CreateObservabilityConfiguration— 새로운 옵저버빌리티 구성을 생성하거나 기존 옵저버빌리티 구성을 수정합니다.
- ListObservabilityConfigurations— 사용자와 관련된 통합 가시성 구성 목록을 요약 정보와 함께 반환합니다. AWS 계정
- DescribeObservabilityConfiguration— 관찰 가능성 구성의 전체 설명을 반환합니다.
- DeleteObservabilityConfiguration— 옵저버빌리티 구성을 삭제합니다. 특정 수정본 또는 최신 활성 버전을 삭제할 수 있습니다. 옵저버빌리티 구성 할당량에 도달하면 불필요한 옵저버빌리티 구성을 삭제해야 할 수도 있습니다. AWS 계정

공유 가능한 리소스를 사용하여 서비스 설정을 구성합니다.

일부 기능의 경우 AWS App Runner 서비스 간에 구성을 공유하는 것이 좋습니다. 예를 들어, 서비스 집합이 동일한 Auto Scaling 동작을 갖기를 원할 수 있습니다. 또는 모든 서비스에 대해 동일한 옵저버빌리티 설정을 원할 수도 있습니다. App Runner를 사용하면 공유 가능한 별도의 리소스를 사용하여 설정을 공유할 수 있습니다. 기능에 대한 구성 설정 세트를 정의하는 리소스를 만든 다음 이 구성 리소스의 Amazon 리소스 이름 (ARN) 을 하나 이상의 App Runner 서비스에 제공합니다.

App Runner는 다음 기능에 대해 공유 가능한 구성 리소스를 구현합니다.

- Auto Scaling
- Observability
- VPC 액세스

각 기능에 대한 문서 페이지는 사용 가능한 설정 및 관리 절차에 대한 정보를 제공합니다.

별도의 구성 리소스를 사용하는 기능은 몇 가지 설계 특성 및 고려 사항을 공유합니다.

- 수정 — 일부 구성 리소스에는 수정이 있을 수 있습니다. 수정 버전을 사용하는 두 구성 리소스의 예로는 오토 스케일링과 옵저버빌리티가 있습니다. 이러한 경우 각 구성에는 이름과 숫자 수정본이 있습니다. 한 구성의 여러 개정판은 이름이 같고 개정 번호도 다릅니다. 시나리오별로 다른 구성 이름을 사용할 수 있습니다. 각 이름에 대해 여러 수정 버전을 추가하여 특정 시나리오에 맞게 설정을 미세 조정할 수 있습니다.

  이름을 지정하여 만든 첫 번째 구성에는 개정 번호 1이 지정됩니다. 이름이 같은 후속 구성에는 연속적인 개정 번호가 부여됩니다 (2부터 시작). App Runner 서비스를 특정 구성 수정 버전 또는 최신 구성 수정 버전과 연결할 수 있습니다.
- 공유 — 여러 App Runner 서비스에서 단일 구성 리소스를 공유할 수 있습니다. 이는 이러한 서비스에서 동일한 구성을 유지하려는 경우에 유용합니다. 특히 리소스에서 수정 버전을 지원하는 경우 구성의 최신 버전을 사용하도록 여러 서비스를 구성할 수 있습니다. 구성 이름만 지정하고 수정은 지정하지 않는 방법으로 이 작업을 수행할 수 있습니다. 이 방법으로 구성한 모든 서비스는 서비스를 업데이트할 때 구성 업데이트를 받습니다. 구성 변경에 대한 자세한 내용은 을 참조하십시오the section called “구성”.
- 리소스 관리 - App Runner를 사용하여 구성을 만들고 삭제할 수 있습니다. 구성을 직접 업데이트할 수는 없습니다. 대신 개정을 지원하는 리소스의 경우 기존 구성 이름을 새로 수정하여 구성을 효과적으로 업데이트할 수 있습니다.

> **Note**
>
> Auto Scaling의 경우 App Runner 콘솔과 App Runner API를 사용하여 구성 및 여러 수정 버전을 만들 수 있습니다. 앱 러너 콘솔과 앱 러너 API 모두 구성 및 수정 버전을 삭제할 수 있습니다. 자세한 내용은 앱 러너 자동 스케일링 관리를 참조하세요.
> 옵저버빌리티 구성과 같은 다른 구성 유형의 경우 App Runner 콘솔을 사용하여 단일 수정 버전으로만 구성을 만들 수 있습니다. 추가 수정 버전을 만들고 구성을 삭제하려면 App Runner API를 사용해야 합니다.

- 리소스 할당량 — 각 구성 리소스에 사용할 수 있는 고유한 구성 이름 및 수정 개수에 대한 할당량이 정해져 있습니다. AWS 리전이 할당량에 도달하면 구성 이름을 삭제하거나 최소한 일부 수정 버전을 삭제해야 더 많이 만들 수 있습니다. Auto Scaling 구성 수정의 경우 앱 러너 콘솔 또는 앱 러너 API를 사용하여 수정할 수 있습니다. 자세한 내용은 앱 러너 자동 스케일링 관리를 참조하세요. 다른 리소스를 삭제하려면 App Runner API를 사용해야 합니다. 할당량에 대한 자세한 내용은 the section called “앱 러너 리소스 할당량” 섹션을 참조하세요.
- 리소스 비용 없음 — 구성 리소스를 만드는 데 추가 비용이 발생하지 않습니다. 기능 자체에는 비용이 발생할 수 있지만 (예: X-Ray Tracing을 켜면 정상 AWS X-Ray 비용이 청구됨) App Runner 서비스용으로 기능을 구성하는 App Runner 구성 리소스에는 비용이 발생하지 않을 수 있습니다.

서비스 상태 점검 구성

AWS App Runner 상태 확인을 수행하여 서비스 상태를 모니터링합니다. 기본 상태 점검 프로토콜은 TCP입니다. App Runner는 서비스에 할당된 도메인을 핑합니다. 또는 상태 확인 프로토콜을 HTTP로 설정할 수도 있습니다. App Runner는 웹 애플리케이션에 상태 점검 HTTP 요청을 보냅니다.

상태 확인과 관련된 몇 가지 설정을 구성할 수 있습니다. 다음 표에서는 상태 확인 설정과 해당 기본값에 대해 설명합니다.

설정	설명	기본값
프로토콜	App Runner가 서비스에 대한 상태 확인을 수행하는 데 사용하는 IP 프로토콜입니다. 프로토콜을 로 TCP 설정하면 App Runner는 애플리케이션이 수신하는 포트에서 서비스에 할당된 기본 도메인을 핑합니다. 프로토콜을 로 HTTP 설정하면 App Runner가 구성된 경로로 상태 확인 요청을 보냅니다.	TCP
경로	앱 러너가 HTTP 상태 확인 요청을 보내는 URL입니다. HTTP 검사에만 적용됩니다.	/
간격	상태 확인 간격(초)입니다.	5
제한 시간	상태 확인 응답이 실패했다고 결정하기 전에 대기하는 시간(초)입니다.	2

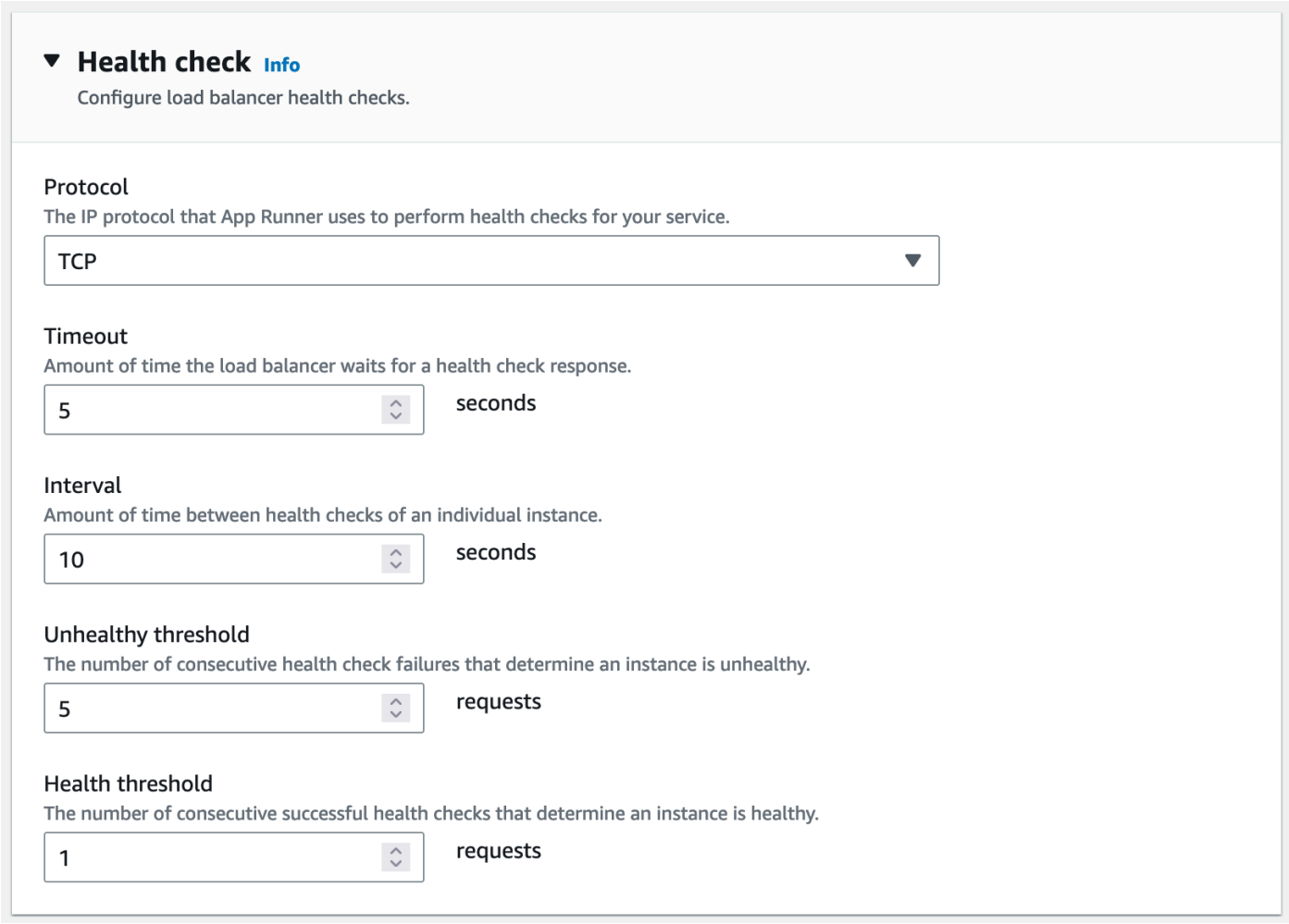
설정	설명	기본값
정상 임계값	App Runner가 서비스가 정상이라고 판단하기 전에 성공해야 하는 연속 확인 횟수입니다.	1
비정상 임계값	App Runner가 서비스가 비정상이라고 판단하기 전에 실패해야 하는 연속 확인 횟수입니다.	5

상태 확인 구성

다음 방법 중 하나를 사용하여 App Runner 서비스의 상태 확인을 구성하십시오.

App Runner console

App Runner 콘솔을 사용하여 App Runner 서비스를 만들거나 나중에 구성을 업데이트할 때 상태 점검 설정을 구성할 수 있습니다. 전체 콘솔 절차는 및 을 참조하십시오the section called “생성”. the section called “구성” 두 경우 모두 콘솔 페이지에서 Health check 구성 섹션을 찾아보십시오.



App Runner API or AWS CLI

CreateService또는 UpdateServiceAPI 작업을 호출할 때 `HealthCheckConfiguration` 파라미터를 사용하여 상태 확인 설정을 지정할 수 있습니다.

매개변수 구조에 대한 자세한 내용은 AWS App Runner API 참조를 참조하십시오 HealthCheckConfiguration.

앱 러너 연결 관리

에서 AWS App Runner서비스를 생성할 때는 애플리케이션 소스 (제공자와 함께 저장되는 컨테이너 이미지 또는 소스 리포지토리) 를 구성합니다. App Runner는 공급자와 인증되고 승인된 연결을 설정해야 합니다. 그러면 App Runner가 리포지토리를 읽고 서비스에 배포할 수 있습니다. App Runner에 저장된 코드에 액세스하는 서비스를 만들 때는 연결 설정이 필요하지 않습니다. AWS 계정

App Runner는 연결이라는 리소스에 연결 정보를 유지 관리합니다. App Runner 콘솔 및 이 가이드에서는 연결을 연결된 계정이라고도 합니다. 타사 연결 정보가 필요한 서비스를 만들 때는 App Runner에 연결 리소스가 필요합니다. 다음은 연결에 대한 몇 가지 중요한 정보입니다.

- 공급자 — 현재 앱 러너에는 GitHub또는 Bitbucket과의 연결 리소스가 필요합니다.
- 공유 - 연결 리소스를 사용하여 동일한 리포지토리 공급자 계정을 사용하는 여러 App Runner 서비스를 만들 수 있습니다.
- 리소스 관리 — App Runner에서 연결을 만들고 삭제할 수 있습니다. 하지만 기존 연결은 수정할 수 없습니다.
- 리소스 할당량 — 연결 리소스에는 각 AWS 계정 리소스에 할당된 할당량이 할당량 할당량과 연결되어 AWS 리전있습니다. 이 할당량에 도달하면 연결을 삭제해야 새 제공업체 계정에 연결할 수 있습니다. 다음 섹션에 설명된 대로 App Runner 콘솔 또는 API를 사용하여 연결을 삭제할 수 있습니다. the section called “연결 관리” 자세한 정보는 the section called “앱 러너 리소스 할당량”을 참조하세요.

연결 관리

다음 방법 중 하나를 사용하여 App Runner 연결을 관리하십시오.

App Runner console

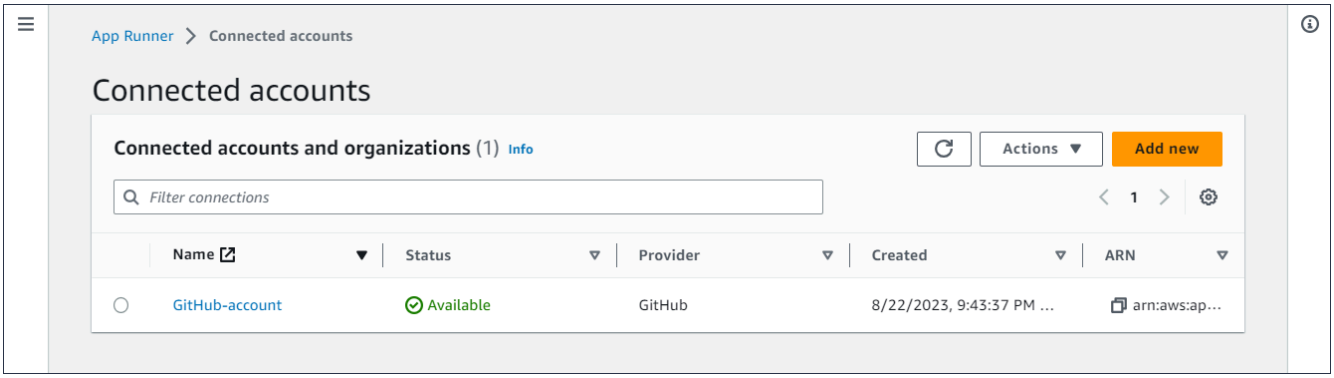
App Runner 콘솔을 사용하여 서비스를 생성할 때는 연결 세부 정보를 제공합니다. 연결 리소스를 명시적으로 생성할 필요는 없습니다. 콘솔에서 이전에 연결했던 Bitbucket 계정 GitHub 또는 Bitbucket 계정에 연결할지 아니면 새 계정에 연결할지 선택할 수 있습니다. 필요한 경우 App Runner에서 연결 리소스를 자동으로 생성합니다. 새 연결의 경우 일부 공급자는 인증 핸드셰이크를 완료해야 연결을 사용할 수 있습니다. 콘솔은 이 프로세스를 안내합니다.

콘솔에는 기존 연결을 관리하는 페이지도 있습니다. 서비스를 생성할 때 인증 핸드셰이크를 수행하지 않았다면 연결에 대한 인증 핸드셰이크를 완료할 수 있습니다. 더 이상 사용하지 않는 연결을 삭제할 수도 있습니다. 다음 절차는 저장소 제공자 연결을 관리하는 방법을 보여줍니다.

계정의 연결을 관리하려면

1. App Runner 콘솔을 열고 지역 목록에서 사용자 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 연결된 계정을 선택합니다.

   그러면 콘솔에 사용자 계정의 저장소 제공자 연결 목록이 표시됩니다.



3. 이제 목록에 있는 모든 연결에서 다음 작업 중 하나를 수행할 수 있습니다.

   - GitHub/Bitbucket 계정 또는 조직 열기 — 연결 이름을 선택합니다.
   - 인증 핸드셰이크 완료 - 연결을 선택한 다음 작업 메뉴에서 핸드셰이크 완료를 선택합니다. 콘솔은 인증 핸드셰이크 프로세스를 안내합니다.
   - 연결 삭제 - 연결을 선택한 다음 작업 메뉴에서 삭제를 선택합니다. 삭제 프롬프트의 지침을 따르십시오.

App Runner API or AWS CLI

다음 App Runner API 작업을 사용하여 연결을 관리할 수 있습니다.

- CreateConnection— 리포지토리 공급자 계정에 대한 연결을 생성합니다. 연결이 생성된 후에는 App Runner 콘솔을 사용하여 인증 핸드셰이크를 수동으로 완료해야 합니다. 이 프로세스는 이전 섹션에 설명되어 있습니다.
- ListConnections— 사용자와 관련된 App Runner 연결 목록을 반환합니다. AWS 계정
- DeleteConnection— 연결을 삭제합니다. 연결 할당량에 도달하면 불필요한 연결을 삭제해야 할 수도 있습니다. AWS 계정

앱 러너 자동 스케일링 관리

AWS App Runner App Runner 애플리케이션의 컴퓨팅 리소스, 특히 인스턴스를 자동으로 늘리거나 줄입니다. 자동 크기 조정은 트래픽이 많을 때 적절한 요청 처리를 제공하고 트래픽 속도가 느려질 때 비용을 줄여줍니다.

오토 스케일링 구성

몇 가지 파라미터를 구성하여 서비스의 Auto Scaling 동작을 조정할 수 있습니다. App Runner는 라는 공유 가능한 리소스에서 Auto Scaling 설정을 유지 관리합니다. AutoScalingConfiguration 서비스에 할

당하기 전에 독립형 Auto Scaling 구성을 생성하고 유지 관리할 수 있습니다. 서비스에 연결한 후에도 구성을 계속 유지 관리할 수 있습니다. 새 서비스를 생성하거나 기존 서비스를 구성하는 중에 새 Auto Scaling 구성을 생성하도록 선택할 수도 있습니다. 새 Auto Scaling 구성이 생성되면 이를 서비스에 연결하고 서비스를 생성 또는 구성하는 프로세스를 계속할 수 있습니다.

이름 지정 및 수정

Auto Scaling 구성에는 이름과 숫자 수정이 있습니다. 구성의 여러 수정 버전은 이름이 같고 수정 번호도 다릅니다. 고가용성 또는 저렴한 비용과 같은 다양한 Auto Scaling 시나리오에 다른 구성 이름을 사용할 수 있습니다. 각 이름에 대해 여러 수정 버전을 추가하여 특정 시나리오에 맞게 설정을 미세 조정할 수 있습니다. 최대 10개의 고유한 Auto Scaling 구성 이름과 각 구성에 대해 최대 5개의 수정 사항을 가질 수 있습니다. 한도에 도달하여 더 생성해야 하는 경우 하나를 삭제한 다음 새로 만들 수 있습니다. App Runner에서는 기본 구성으로 설정되어 있거나 활성 서비스에서 사용 중인 구성을 삭제할 수 없습니다. 할당량에 대한 자세한 내용은 the section called “앱 러너 리소스 할당량” 섹션을 참조하세요.

기본 구성 설정

App Runner 서비스를 만들거나 업데이트할 때 Auto Scaling 구성 리소스를 제공할 수 있습니다. Auto Scaling 구성을 제공하는 것은 선택 사항입니다. 제공하지 않는 경우 App Runner는 권장 값이 포함된 기본 Auto Scaling 구성을 제공합니다. Auto Scaling 구성 기능은 App Runner에서 제공하는 기본값을 사용하는 대신 자체 기본 Auto Scaling 구성을 설정하는 옵션을 제공합니다. 다른 Auto Scaling 구성을 기본 구성으로 지정하면 해당 구성이 향후 새로 생성하는 서비스에 기본 구성으로 자동 할당됩니다. 새 기본 지정은 기존 서비스에 대해 이전에 설정된 연결에는 영향을 주지 않습니다.

Auto Scaling을 통한 서비스 구성

여러 App Runner 서비스에서 단일 Auto Scaling 구성을 공유하여 서비스가 동일한 Auto Scaling 동작을 유지하도록 할 수 있습니다. App Runner 콘솔 또는 App Runner API를 사용하여 Auto Scaling 구성을 구성하는 방법에 대한 자세한 내용은 이 항목의 다음 섹션을 참조하십시오. 공유 가능한 리소스에 대한 보다 일반적인 정보는 을 참조하십시오. the section called “구성 리소스”

구성 가능한 설정

다음과 같은 Auto Scaling 설정을 구성할 수 있습니다.

- 최대 동시성 — 인스턴스가 처리하는 최대 동시 요청 수입니다. 동시 요청 수가 이 할당량을 초과하면 App Runner는 서비스를 확장합니다.
- 최대 크기 - 서비스를 확장할 수 있는 최대 인스턴스 수입니다. 서비스 트래픽을 동시에 처리할 수 있는 최대 인스턴스 수입니다.

- 최소 크기 — App Runner가 서비스에 프로비저닝할 수 있는 최소 인스턴스 수입니다. 서비스에는 항상 최소 이 개수의 프로비저닝된 인스턴스가 있습니다. 이러한 인스턴스 중 일부는 트래픽을 능동적으로 처리합니다. 나머지는 비용 효율적인 컴퓨팅 파워 리저브의 일부이며 빠르게 활성화할 수 있습니다. 프로비저닝된 모든 인스턴스의 메모리 사용량에 따라 비용을 지불합니다. 활성 서브셋의 CPU 사용량에 대해서만 비용을 지불합니다.

> **Note**
>
> vCPU 리소스 수는 App Runner가 서비스에 제공할 수 있는 인스턴스 수를 결정합니다. 서비스에 있는 Fargate 온디맨드 vCPU 리소스 수에 대한 조정 가능한 할당량 값입니다. AWS Fargate (Fargate) 계정의 vCPU 할당량 설정을 보거나 할당량 증가를 요청하려면 의 Service Quotas 콘솔을 사용하십시오. AWS Management Console자세한 내용은 Amazon Elastic 컨테이너AWS Fargate 서비스 개발자 안내서의 서비스 할당량을 참조하십시오.

서비스의 Auto Scaling 관리

다음 방법 중 하나를 사용하여 App Runner 서비스의 Auto Scaling을 관리하세요.

App Runner console

App Runner 콘솔을 사용하여 서비스를 만들거나 서비스 구성을 업데이트할 때 Auto Scaling 구성을 지정할 수 있습니다.

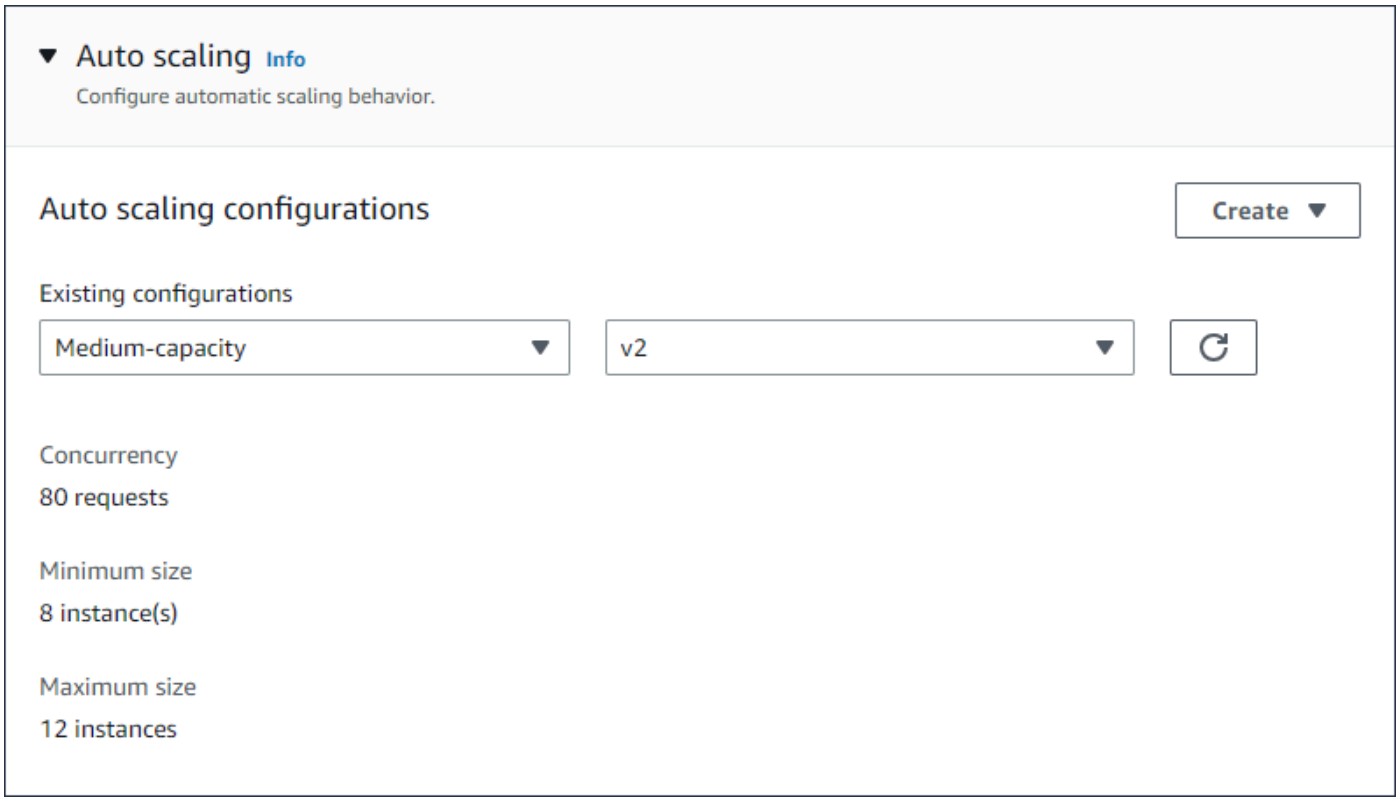
> **Note**
>
> 서비스와 관련된 Auto Scaling 구성 또는 수정 버전을 변경하면 서비스가 다시 배포됩니다.

Auto Scaling 구성 페이지는 서비스에 대한 Auto Scaling을 구성하는 몇 가지 옵션을 제공합니다.

- 기존 구성 및 수정 버전을 할당하려면 — 기존 구성 드롭다운에서 값을 선택합니다. 옆에 있는 드롭다운에는 최신 수정 버전이 기본값으로 표시됩니다. 선택하려는 다른 개정이 있는 경우 개정 드롭다운에서 선택하십시오. 수정 버전의 구성 값이 표시됩니다.
- 새 Auto Scaling 구성을 생성하고 할당하려면 — 생성 메뉴에서 새 ASC 생성을 선택합니다. 그러면 사용자 지정 Auto Scaling 구성 추가 페이지가 시작됩니다. Auto Scaling 파라미터의 구성 이름과 값을 입력합니다. 그런 다음 추가를 선택합니다. App Runner는 사용자를 위해 새 Auto

Scaling 구성 리소스를 생성하고 새 구성을 선택하고 표시한 상태로 Auto Scaling 섹션으로 돌아갑니다.

- 새 수정 버전을 생성하고 할당하려면 먼저 기존 구성 드롭다운에서 구성 이름을 선택합니다. 그런 다음 만들기 메뉴에서 ASC 개정 만들기를 선택합니다. 그러면 사용자 지정 Auto Scaling 구성 추가 페이지가 시작됩니다. Auto Scaling 파라미터의 값을 입력합니다. 그런 다음 추가를 선택합니다. App Runner가 새 Auto Scaling 구성 수정 버전을 생성하고 새 수정 버전을 선택하고 표시한 상태로 Auto Scaling 섹션으로 돌아갑니다.



App Runner API or AWS CLI

CreateService또는 UpdateServiceApp Runner API 작업을 호출할 때 `AutoScalingConfigurationArn` 파라미터를 사용하여 서비스의 Auto Scaling 구성 리소스를 지정할 수 있습니다.

다음 섹션에서는 Auto Scaling 구성 리소스를 관리하기 위한 지침을 제공합니다.

자동 스케일링 구성 리소스 관리

다음 방법 중 하나를 사용하여 계정의 App Runner Auto Scaling 구성 및 수정을 관리하십시오.

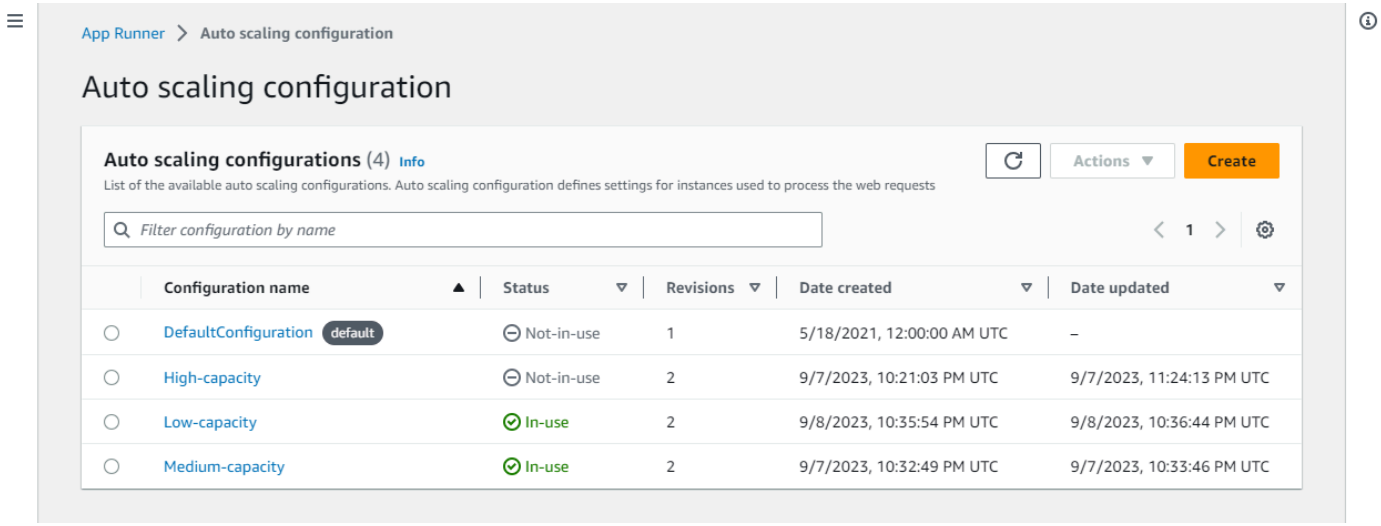
App Runner console

자동 스케일링 구성 관리

Auto Scaling 구성 페이지에는 계정에서 설정한 Auto Scaling 구성이 나열됩니다. 이 페이지에서 Auto Scaling 구성을 만들고 관리한 다음 나중에 하나 이상의 App Runner 서비스에 할당할 수 있습니다.

이 페이지에서 다음 중 하나를 수행할 수 있습니다.

- 새 Auto Scaling 구성을 생성합니다.
- 기존 Auto Scaling 구성에 대한 새 수정 버전을 생성합니다.
- Auto Scaling 구성을 삭제합니다.
- Auto Scaling 구성을 기본값으로 설정합니다.



계정에서 Auto Scaling 구성을 관리하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 Auto Scaling 구성을 선택합니다. 콘솔에는 계정의 Auto Scaling 구성 목록이 표시됩니다.

   이제 다음 중 하나를 수행할 수 있습니다.

   - 새 Auto Scaling 구성을 생성하려면 다음 단계를 따르십시오.

     a. Auto Scaling 구성 페이지에서 생성을 선택합니다.

Auto Scaling 구성 생성 페이지가 표시됩니다.

b. 구성 이름, 동시성, 최소 크기, 최대 크기 값을 입력합니다.

c. (선택 사항) 태그를 추가하려면 자동 새 태그를 선택합니다. 그런 다음 나타나는 필드에 이름과 값 (선택 사항) 을 입력합니다.

d. 생성을 선택합니다.

- 기존 Auto Scaling 구성에 대한 새 수정 버전을 생성하려면 다음 단계를 따르십시오.

  a. Auto Scaling 구성 페이지에서 새 수정이 필요한 구성 옆에 있는 라디오 버튼을 선택합니다. 그런 다음 작업 메뉴에서 수정 버전 만들기를 선택합니다.

     리비전 만들기 페이지가 표시됩니다.

  b. 켜기에서 동시성, 최소 크기, 최대 크기 값을 입력합니다.

  c. (선택 사항) 태그를 추가하려면 새 태그 자동을 선택합니다. 그런 다음 나타나는 필드에 이름과 값 (선택 사항) 을 입력합니다.

  d. 생성을 선택합니다.

- Auto Scaling 구성을 삭제하려면 다음 단계를 따르십시오.

  a. Auto Scaling 구성 페이지에서 삭제하려는 구성 옆에 있는 라디오 버튼을 선택합니다.

  b. 작업 메뉴에서 삭제를 선택합니다.

  c. 삭제를 계속하려면 확인 대화 상자에서 삭제를 선택합니다. 그렇지 않으면 취소를 선택합니다.

  > **Note**
  >
  > App Runner는 삭제 선택 항목이 기본값으로 설정되어 있지 않거나 활성 서비스에서 현재 사용 중인지 확인합니다.

- Auto Scaling 구성을 기본값으로 설정하려면 다음 단계를 따르십시오.

  a. Auto Scaling 구성 페이지에서 기본값으로 설정해야 하는 구성 옆에 있는 라디오 버튼을 선택합니다.

  b. 작업 메뉴에서 기본값으로 설정을 선택합니다.

  c. App Runner에서 새로 만드는 모든 서비스의 기본 구성으로 최신 버전을 사용한다는 내용의 대화 상자가 표시됩니다. 확인을 선택하여 계속 진행하십시오. 그렇지 않으면 취소를 선택합니다.

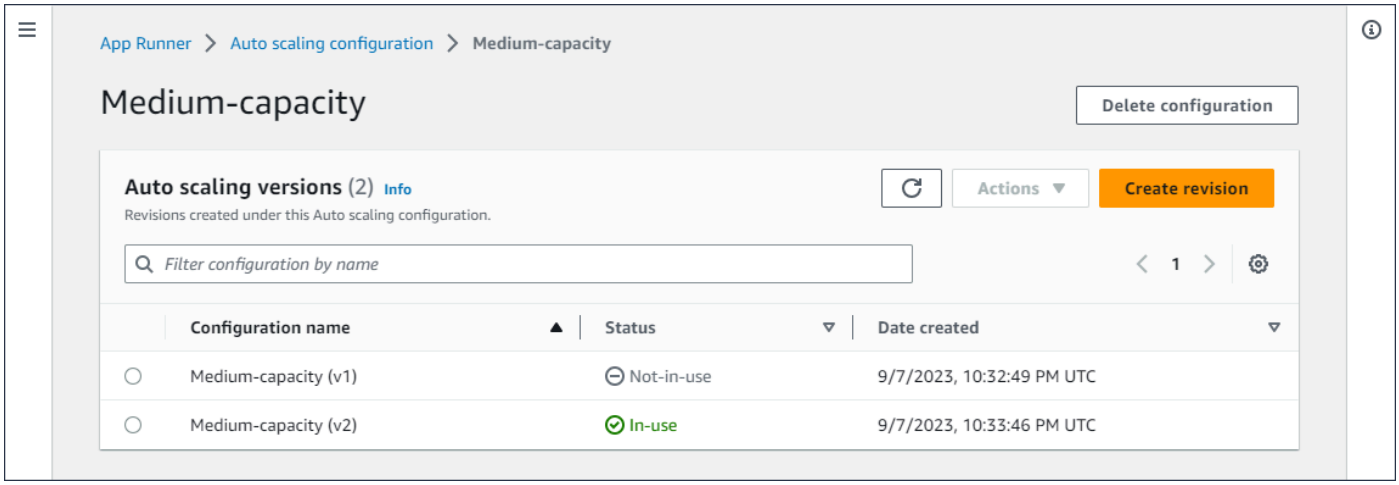
> **Note**
>
> - Auto Scaling 구성을 기본값으로 설정하면 향후 새로 생성하는 서비스에 자동으로 기본 구성으로 할당됩니다.
> - 새 기본 지정은 기존 서비스에 대해 이전에 설정된 연결에는 영향을 주지 않습니다.
> - 지정된 기본 Auto Scaling 구성에 수정 사항이 있는 경우 App Runner는 최신 버전을 기본값으로 할당합니다.

수정 내용 관리

콘솔에는 Auto Scaling 수정이라고 하는 기존 Auto Scaling 수정 버전을 만들고 관리할 수 있는 페이지도 있습니다. Auto Scaling 구성 페이지에서 구성 이름을 선택하여 이 페이지에 액세스할 수 있습니다.

Auto Scaling 수정 페이지에서 다음 중 하나를 수행할 수 있습니다.

- 새 Auto Scaling 수정 버전을 생성합니다.
- Auto Scaling 구성 수정 버전을 기본값으로 설정합니다.
- 수정 버전 삭제.
- 모든 관련 수정 버전을 포함하여 전체 Auto Scaling 구성을 삭제합니다.
- 개정판의 구성 세부 정보를 확인하세요.
- 개정판과 관련된 서비스 목록을 볼 수 있습니다.
- 나열된 서비스의 수정 버전을 변경합니다.



계정에서 Auto Scaling 수정 사항을 관리하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 Auto Scaling 구성을 선택합니다. 콘솔에는 계정의 Auto Scaling 구성 목록이 표시됩니다. 자동 스케일링 구성 관리섹션의 이전 절차에는 이 페이지의 화면 이미지가 포함되어 있습니다.
3. 이제 특정 Auto Scaling 구성을 자세히 분석하여 모든 수정 사항을 보고 관리할 수 있습니다. Auto Scaling 구성 창의 구성 이름 열에서 Auto Scaling 구성 이름을 선택합니다. 라디오 버튼 대신 실제 이름을 선택하십시오. 그러면 Auto Scaling 수정 페이지에서 해당 구성의 모든 수정 목록이 표시됩니다.
4. 이제 다음 중 하나를 수행할 수 있습니다.
   - 기존 Auto Scaling 구성에 대한 새 수정 버전을 생성하려면 다음 단계를 따르십시오.
     a. Auto Scaling 수정 페이지에서 수정 버전 생성을 선택합니다.

        수정 버전 생성 페이지가 표시됩니다.
     b. 동시성, 최소 크기, 최대 크기 값을 입력합니다.
     c. (선택 사항) 태그를 추가하려면 자동 새 태그를 선택합니다. 그런 다음 나타나는 필드에 이름과 값 (선택 사항) 을 입력합니다.
     d. 생성을 선택합니다.
   - 모든 관련 수정 버전을 포함하여 전체 Auto Scaling 구성을 삭제하려면 다음 단계를 따르십시오.

     a. 페이지 오른쪽 상단에서 구성 삭제를 선택합니다.

b. 삭제를 계속하려면 확인 대화 상자에서 삭제를 선택합니다. 그렇지 않으면 취소를 선택합니다.

> **Note**
>
> App Runner는 삭제 선택 항목이 기본값으로 설정되어 있지 않거나 활성 서비스에서 현재 사용 중인지 확인합니다.

- Auto Scaling 버전을 기본값으로 설정하려면 다음 단계를 따르십시오.

  a. 기본값으로 설정해야 하는 수정 버전 옆의 라디오 버튼을 선택합니다.

  b. 작업 메뉴에서 기본값으로 설정을 선택합니다.

> **Note**
>
> - Auto Scaling 구성을 기본값으로 설정하면 향후 새로 생성하는 서비스에 자동으로 기본 구성으로 할당됩니다.
> - 새 기본 지정은 기존 서비스에 대해 이전에 설정된 연결에는 영향을 주지 않습니다.

- 개정판의 구성 세부 정보를 보려면 다음 단계를 따르십시오.

  - 수정 버전 옆에 있는 라디오 버튼을 선택합니다.

    ARN을 포함한 수정 버전의 구성 세부 정보가 하단 분할 패널에 표시됩니다. 이 절차의 마지막에 있는 화면 이미지를 참조하십시오.

- 개정판과 관련된 서비스 목록을 보려면 다음 단계를 따르십시오.

  - 수정 버전 옆에 있는 라디오 버튼을 선택합니다.

    서비스 패널은 아래쪽 분할 패널의 수정 구성 세부 정보 아래에 표시됩니다. 패널에는 이 Auto Scaling 구성 수정 버전을 사용하는 모든 서비스가 나열됩니다. 이 절차의 끝에 있는 화면 이미지를 참조하십시오.

- 나열된 서비스의 버전을 변경하려면 다음 단계를 따르십시오.

  a. 아직 수정하지 않았다면 개정 옆에 있는 라디오 버튼을 선택하십시오.

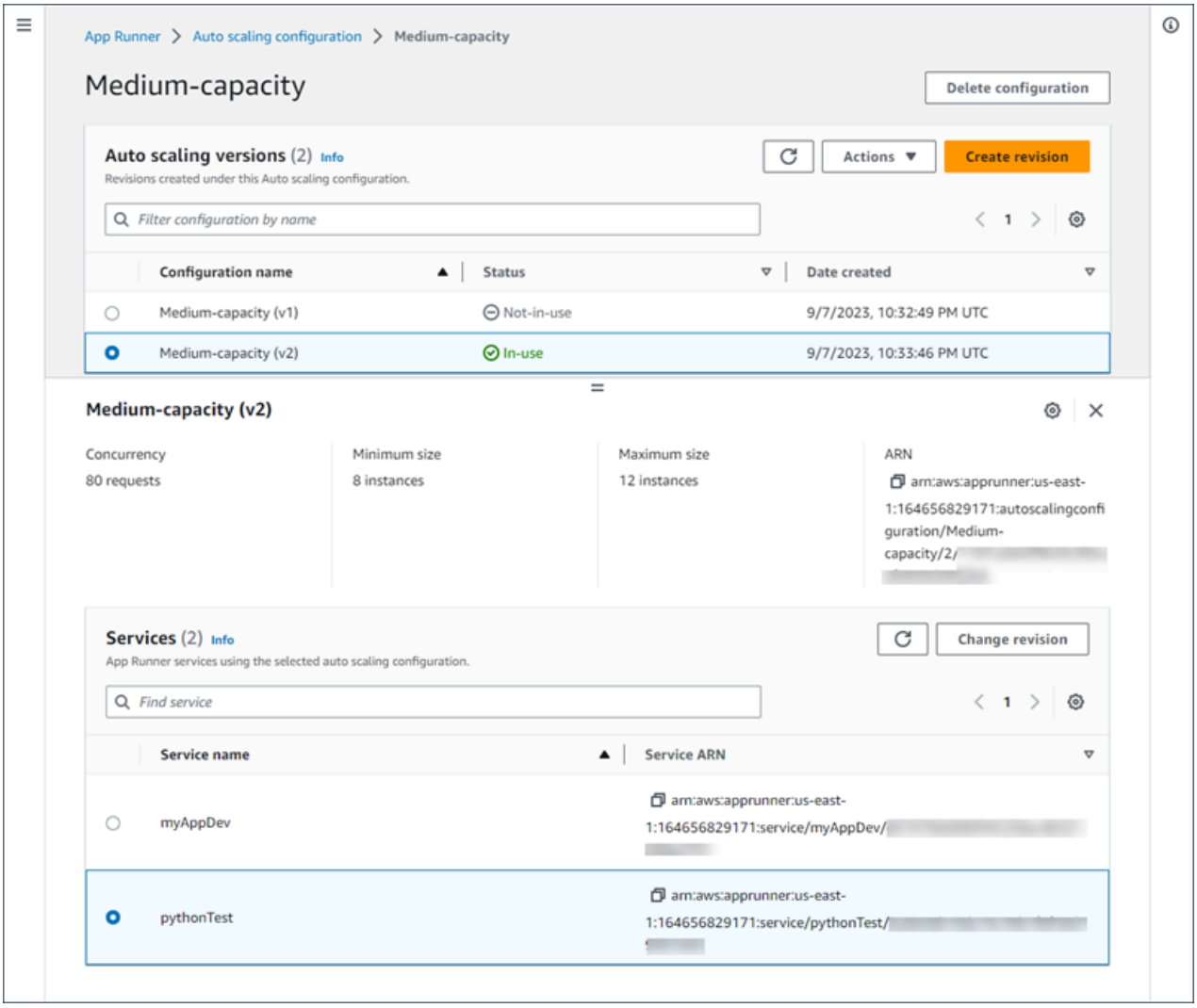
서비스 패널은 아래쪽 분할 패널의 수정 구성 세부 정보 아래에 표시됩니다. 패널에는 이 Auto Scaling 구성 수정 버전을 사용하는 모든 서비스가 나열됩니다. 이 절차의 끝에 있는 화면 이미지를 참조하십시오.

b. 서비스 패널에서 수정하려는 서비스 옆에 있는 라디오 버튼을 선택합니다. 그런 다음 수정 내용 변경을 선택합니다.

c. ASC 수정 변경 패널이 표시됩니다. 드롭다운에서 사용 가능한 수정 버전 중에서 선택합니다. 이전에 선택한 Auto Scaling 구성의 개정판만 사용할 수 있습니다. 다른 Auto Scaling 구성으로 변경해야 하는 경우 이전 섹션의 절차를 따르십시오the section called “서비스의 Auto Scaling 관리”.

업데이트를 선택하여 변경을 계속하십시오. 그렇지 않으면 취소를 선택합니다.

> **Note**
>
> 서비스와 관련된 수정 버전을 변경하면 서비스가 다시 배포됩니다.
> 업데이트된 연결을 보려면 이 패널에서 새로 고침을 선택해야 합니다.
> 진행 중인 활동과 서비스 재배포의 상태를 보려면 패널 이동 경로를 사용하여 App Runner > 서비스로 이동하여 서비스를 선택한 다음 서비스 개요 패널에서 로그 탭을 확인하십시오.



App Runner API or AWS CLI

다음 App Runner API 작업을 사용하여 Auto Scaling 구성 리소스를 관리하세요.

- CreateAutoScalingConfiguration— 새 Auto Scaling 구성을 생성하거나 기존 구성을 수정합니다.
- UpdateDefaultAutoScalingConfiguration—Auto Scaling 구성을 기본값으로 설정합니다. 기존의 기본 Auto Scaling 구성은 기본값이 아닌 것으로 자동 설정됩니다.
- ListAutoScalingConfigurations— 사용자와 관련된 Auto Scaling 구성 목록을 요약 정보와 함께 반환합니다. AWS 계정
- ListServicesForAutoScalingConfiguration— Auto Scaling 구성을 사용하여 연결된 App Runner 서비스 목록을 반환합니다.

- DescribeAutoScalingConfiguration— Auto Scaling 구성에 대한 전체 설명을 반환합니다.
- DeleteAutoScalingConfiguration— Auto Scaling 구성을 삭제합니다. 최상위 Auto Scaling 구성, 특정 수정 버전 또는 최상위 구성과 관련된 모든 수정 버전을 삭제할 수 있습니다. 모든 수정 버전을 삭제하려면 선택적 `DeleteAllRevisions` 매개변수를 사용하십시오. 의 Auto Scaling 구성 리소스 할당량에 도달하면 불필요한 Auto Scaling 구성을 삭제해야 할 수 있습니다. AWS 계정

App Runner 서비스의 사용자 지정 도메인 이름 관리

서비스를 만들면 App Runner가 AWS App Runner 서비스에 도메인 이름을 할당합니다. 이 도메인은 App Runner가 소유한 `awsapprunner.com` 도메인의 하위 도메인입니다. 도메인 이름을 사용하여 서비스에서 실행 중인 웹 애플리케이션에 액세스할 수 있습니다.

> **Note**
>
> 앱 러너 애플리케이션의 보안을 강화하기 위해*.awsapprunner.com 도메인은 공개 접미사 목록 (PSL) 에 등록되어 있습니다. 보안 강화를 위해 App Runner 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

도메인 이름을 소유하고 있는 경우 이 이름을 App Runner 서비스에 연결할 수 있습니다. App Runner에서 새 도메인의 유효성을 검사한 후에는 App Runner 도메인 외에도 도메인을 사용하여 애플리케이션에 액세스할 수 있습니다. 최대 5개의 사용자 지정 도메인을 연결할 수 있습니다.

> **Note**
>
> 도메인의 `www` 하위 도메인을 선택적으로 포함할 수 있습니다. 하지만 현재는 API에서만 지원됩니다. App Runner 콘솔은 `www` 도메인의 하위 도메인 포함을 지원하지 않습니다.

맞춤 도메인을 서비스에 연결 (링크) 하십시오.

사용자 지정 도메인을 서비스에 연결할 때는 CNAME 레코드와 DNS 대상 레코드를 DNS 서버에 추가해야 합니다. 다음 섹션에서는 CNAME 레코드 및 DNS 대상 레코드에 대한 정보와 이를 사용하는 방법을 제공합니다.

> **Note**
>
> Amazon Route 53을 DNS 공급자로 사용하는 경우 App Runner는 App Runner 웹 애플리케이션에 연결하기 위해 필요한 인증서 검증 및 DNS 레코드를 사용하여 사용자 지정 도메인을 자동으로 구성합니다. 이는 App Runner 콘솔을 사용하여 사용자 지정 도메인을 서비스에 연결할 때 발생합니다. 다음 커스텀 도메인 관리 항목에서는 자세한 정보를 제공합니다.

CNAME 레코드

맞춤 도메인을 서비스에 연결하면 App Runner에서 인증서 검증을 위한 인증서 검증 레코드 세트를 제공합니다. 이러한 인증서 검증 레코드를 도메인 이름 시스템 (DNS) 서버에 추가해야 합니다. App Runner에서 제공하는 인증서 검증 레코드를 DNS 서버에 추가합니다. 이렇게 하면 App Runner에서 도메인의 소유 또는 제어 여부를 검증할 수 있습니다.

> **Note**
>
> 사용자 지정 도메인 인증서를 자동 갱신하려면 DNS 서버에서 인증서 검증 레코드를 삭제하지 마세요. 인증서 갱신과 관련된 문제를 해결하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “커스텀 도메인 인증서 갱신”

App Runner는 ACM을 사용하여 도메인을 확인합니다. DNS 레코드에서 CAA 레코드를 사용하는 경우 하나 이상의 CAA 레코드가 참조하는지 확인하십시오. `amazon.com` 그렇지 않으면 ACM이 도메인을 확인하고 도메인을 성공적으로 생성할 수 없습니다.

CAA와 관련된 오류가 발생하는 경우 다음 링크를 참조하여 해결 방법을 알아보십시오.

- CAA (인증 기관 인증) 문제
- ACM 인증서 발급 또는 갱신 시 발생하는 CAA 오류를 해결하려면 어떻게 해야 합니까?
- 사용자 지정 도메인 이름

> **Note**
>
> Amazon Route 53을 DNS 공급자로 사용하는 경우 App Runner는 App Runner 웹 애플리케이션에 연결하기 위해 필요한 인증서 검증 및 DNS 레코드를 사용하여 사용자 지정 도메인을 자

동으로 구성합니다. 이는 App Runner 콘솔을 사용하여 사용자 지정 도메인을 서비스에 연결할 때 발생합니다. 다음 커스텀 도메인 관리 항목에서는 자세한 정보를 제공합니다.

DNS 대상 레코드

DNS 대상 레코드를 DNS 서버에 추가하여 앱 러너 도메인을 타겟팅합니다. 이 옵션을 선택한 경우 사용자 지정 도메인과 www 하위 도메인용 레코드 하나를 추가합니다. 그런 다음 App Runner 콘솔에서 사용자 지정 도메인 상태가 Active로 전환될 때까지 기다리세요. 일반적으로 몇 분 정도 소요되지만 최대 24~48시간 (1~2일) 이 소요될 수 있습니다. 사용자 지정 도메인이 검증되면 App Runner는 이 도메인에서 웹 애플리케이션으로 트래픽을 라우팅하기 시작합니다.

> **Note**
>
> 앱 러너 서비스와의 호환성을 높이려면 Amazon Route 53을 DNS 공급자로 사용하는 것이 좋습니다. Amazon Route 53을 사용하여 퍼블릭 DNS 레코드를 관리하지 않는 경우 DNS 공급자에게 문의하여 레코드 추가 방법을 알아보십시오.
>
> Amazon Route 53을 DNS 공급자로 사용하는 경우 하위 도메인에 CNAME 또는 별칭 레코드를 추가할 수 있습니다. 루트 도메인의 경우 별칭 레코드를 사용해야 합니다.

Amazon Route 53 또는 다른 공급자로부터 도메인 이름을 구입할 수 있습니다. Amazon Route 53에서 도메인 이름을 구매하려면 Amazon Route 53 개발자 안내서의 새 도메인 등록을 참조하십시오.

Route 53에서 DNS 대상을 구성하는 방법에 대한 지침은 Amazon Route 53 개발자 안내서의 리소스로의 트래픽 라우팅을 참조하십시오.

Shopify, Hover 등과 같은 GoDaddy 다른 등록 기관에서 DNS 대상을 구성하는 방법에 대한 지침은 DNS 대상 레코드 추가에 대한 해당 등록 기관의 특정 설명서를 참조하십시오.

App Runner 서비스에 연결할 도메인을 지정하십시오.

다음과 같은 방법으로 App Runner 서비스에 연결할 도메인을 지정할 수 있습니다.

- 루트 도메인 — DNS에는 루트 도메인 이름에 대한 CNAME 레코드를 생성하지 못할 수 있는 몇 가지 고유한 제한이 있습니다. 예를 들어 도메인 이름이 인 경우 트래픽을 App `example.com` Runner 서비스로 라우팅하는 CNAME 레코드를 만들 수 있습니다. `acme.example.com` 하지만 트래픽을 App Runner 서비스로 라우팅하는 CNAME 레코드를 만들 수는 없습니다. `example.com` 루트 도메인을 만들려면 별칭 레코드를 추가해야 합니다.

별칭 레코드는 Route 53에만 해당되며 CNAME 레코드에 비해 다음과 같은 이점이 있습니다.

- 루트 도메인이나 하위 도메인에 대해 별칭 레코드를 생성할 수 있으므로 Route 53은 보다 유연하게 사용할 수 있습니다. 예를 들어 도메인 이름이 인 경우 App `example.com` Runner 서비스에 대한 `example.com` 또는 요청을 App Runner 서비스로 `acme.example.com` 라우팅하는 레코드를 생성할 수 있습니다.
- 비용 효율성이 더 높습니다. Route 53은 별칭 레코드를 사용하여 트래픽을 라우팅하는 요청에 대해서는 요금을 부과하지 않기 때문입니다.

- 하위 도메인 — 예: 또는. `login.example.com` `admin.login.example.com` 선택적으로 동일한 작업의 일부로 `www` 하위 도메인을 연결할 수도 있습니다. 하위 도메인에 CNAME 또는 별칭 레코드를 추가할 수 있습니다.
- 와일드카드 — 예:. `*.example.com` 이 경우에는 `www` 옵션을 사용할 수 없습니다. 와일드카드는 루트 도메인의 직속 하위 도메인으로만 지정할 수 있으며 단독으로만 지정할 수 있습니다. 다음은 유효한 사양이 아닙니다:`login*.example.com`,. `*.login.example.com` 이 와일드카드 사양은 모든 직속 하위 도메인을 연결하며 루트 도메인 자체를 연결하지는 않습니다. 루트 도메인은 별도의 작업으로 연결해야 합니다.

보다 구체적인 도메인 연결이 덜 구체적인 도메인 연결보다 우선합니다. 오버라이드를 예로 들 수 있습니다. `login.example.com` `*.example.com` 보다 구체적인 연결의 인증서와 CNAME이 사용됩니다.

다음 예는 여러 사용자 지정 도메인 연결을 사용하는 방법을 보여줍니다.

1. 서비스 홈페이지에 `example.com` 연결하세요. 연결할 수 `www` 있도록 설정합니다`www.example.com`.
2. 서비스의 로그인 페이지에 `login.example.com` 연결하세요.
3. 사용자 지정 '찾을 수 없음' 페이지에 `*.example.com` 연결하세요.

커스텀 도메인 연결 해제 (연결 해제)

App Runner 서비스에서 커스텀 도메인을 분리 (연결 해제) 할 수 있습니다. 도메인 연결을 해제하면 App Runner는 이 도메인에서 웹 애플리케이션으로의 트래픽 라우팅을 중단합니다.

> **Note**
>
> DNS 서버에서 연결을 끊은 도메인의 레코드를 삭제해야 합니다.

App Runner는 내부적으로 도메인 유효성을 추적하는 인증서를 생성합니다. 이러한 인증서는 AWS Certificate Manager (ACM) 에 저장됩니다. App Runner는 도메인이 서비스에서 분리되거나 서비스가 삭제된 후 7일 동안 이러한 인증서를 삭제하지 않습니다.

커스텀 도메인 관리

다음 방법 중 하나를 사용하여 App Runner 서비스의 사용자 지정 도메인을 관리합니다.

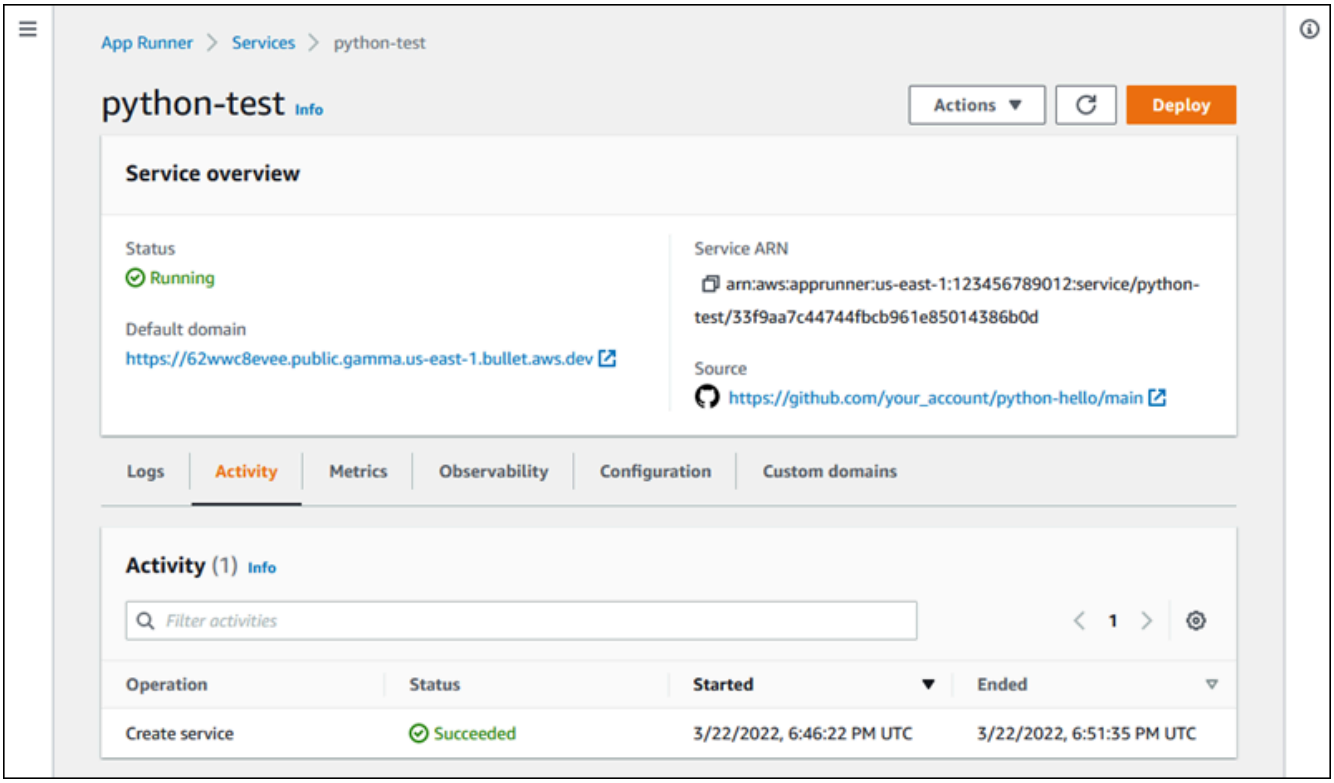
> **Note**
>
> 앱 러너 서비스와의 호환성을 높이려면 Amazon Route 53을 DNS 공급자로 사용하는 것이 좋습니다. Amazon Route 53을 사용하여 퍼블릭 DNS 레코드를 관리하지 않는 경우 DNS 공급자에게 문의하여 레코드 추가 방법을 알아보십시오.
> Amazon Route 53을 DNS 공급자로 사용하는 경우 하위 도메인에 CNAME 또는 별칭 레코드를 추가할 수 있습니다. 루트 도메인의 경우 별칭 레코드를 사용해야 합니다.

App Runner console

App Runner 콘솔을 사용하여 커스텀 도메인을 연결 (링크) 하려면

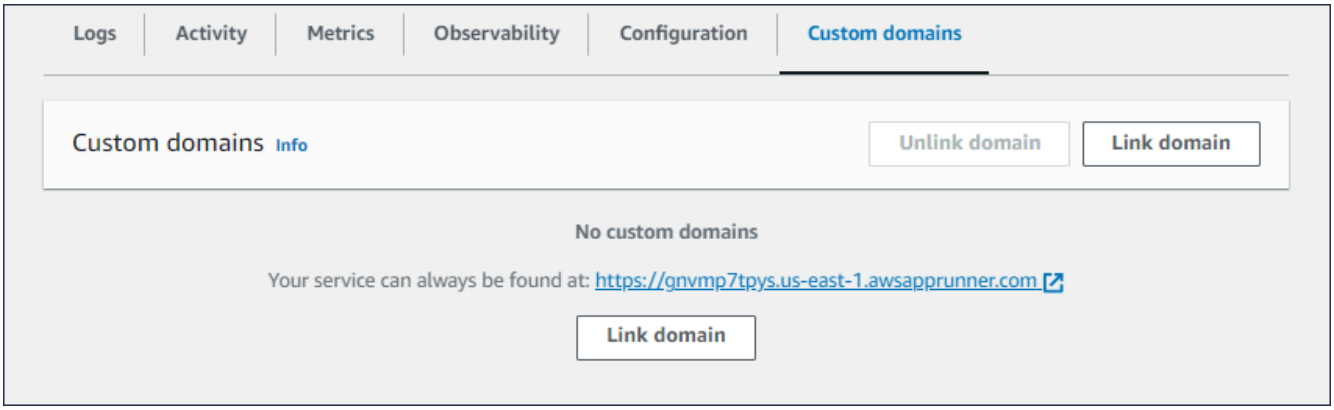
1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

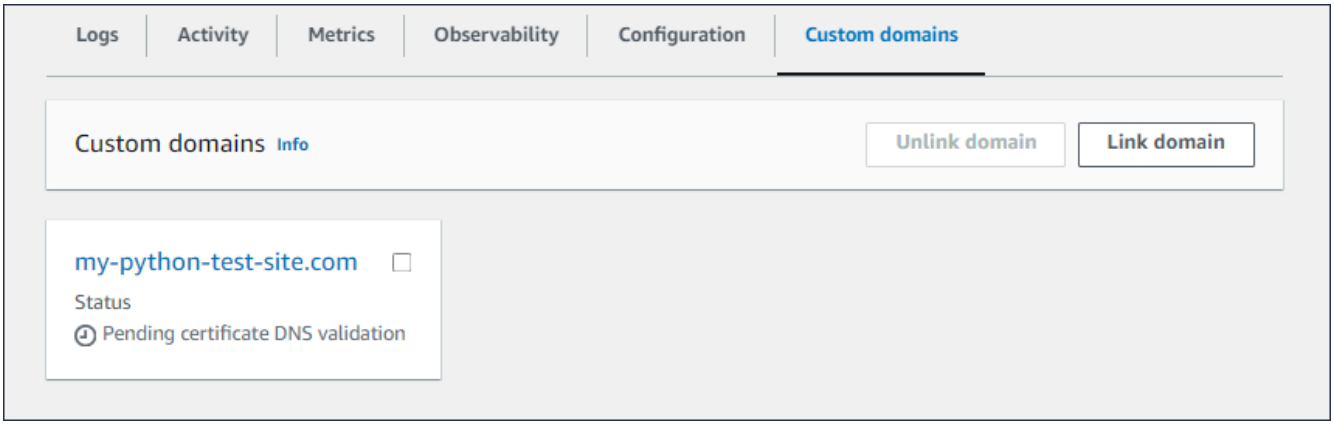
   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 커스텀 도메인 탭을 선택합니다.

   콘솔에 서비스와 연결된 커스텀 도메인이 표시되거나 커스텀 도메인 없음이 표시됩니다.

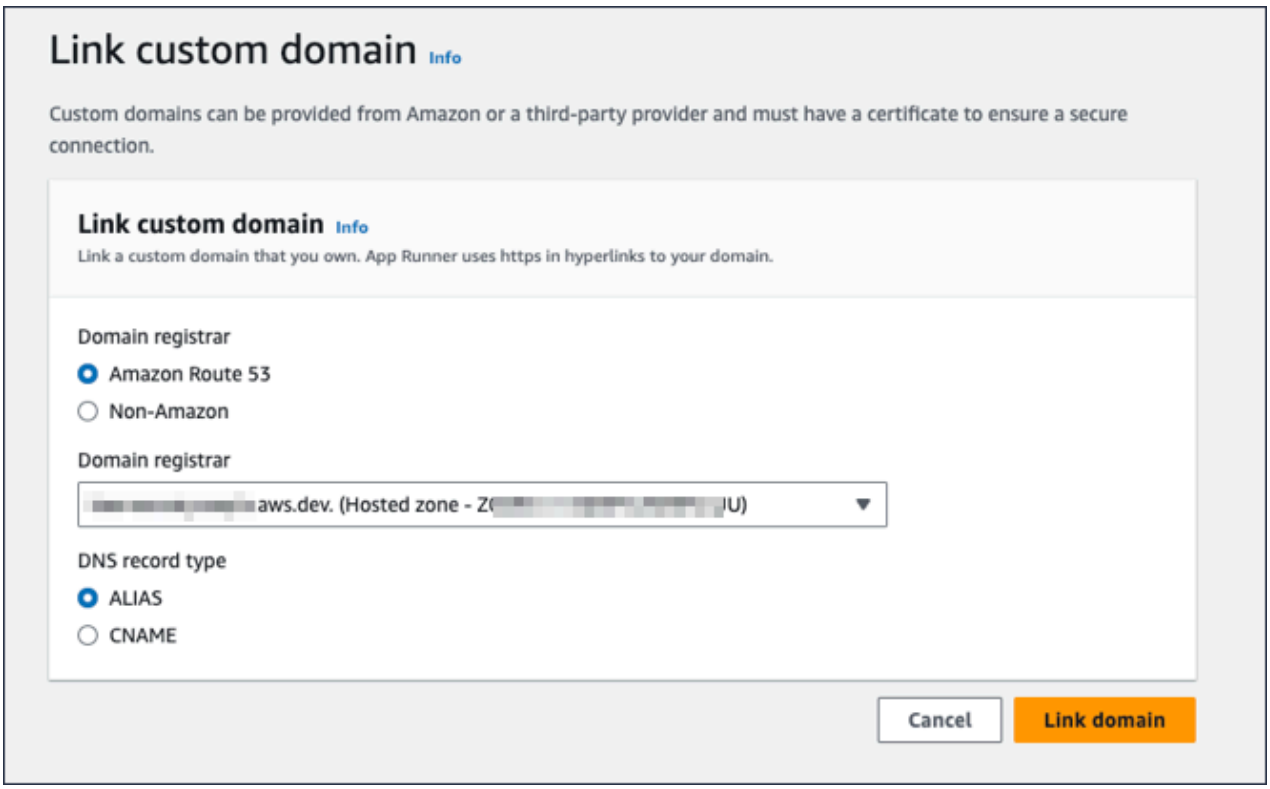




4. 커스텀 도메인 탭에서 도메인 연결을 선택합니다.
5. 커스텀 도메인 연결 페이지가 표시됩니다.
   - 사용자 지정 도메인이 Amazon Route 53에 등록된 경우 도메인 등록 기관으로 Amazon Route 53을 선택하십시오.
     a. 드롭다운 목록에서 도메인 이름을 선택합니다. 이 목록에는 Route 53 도메인 이름의 이름과 호스팅 영역 ID가 표시됩니다.

        > **Note**
        >
        > 먼저 다른 앱 러너 리소스를 관리하는 데 사용하는 것과 동일한 AWS 계정에서 Amazon Route 53 서비스를 사용하여 Route 53 도메인을 생성해야 합니다.

     b. DNS 레코드 유형을 선택합니다.
     c. 링크 도메인을 선택합니다.

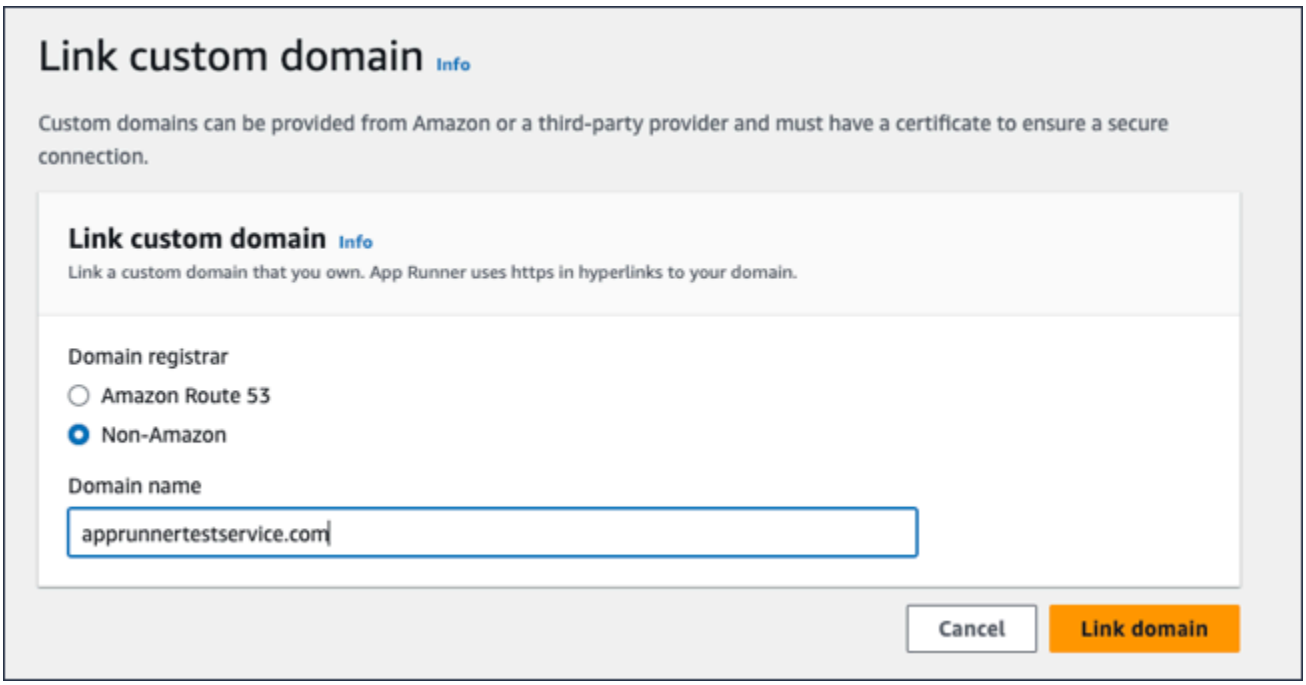


> **Note**
>
> App Runner에서 자동 구성 시도가 실패했다는 오류 메시지를 표시하는 경우 DNS 레코드를 수동으로 구성하여 계속 진행할 수 있습니다. 이 문제는 동일한 도메인 이름을 이전에 서비스에서 연결 해제했지만 서비스를 가리키는 DNS 공급자 레코드가 나중에 삭제되지 않은 경우 발생할 수 있습니다. 이 경우 App Runner는 이러한 레코드를 자동으로 덮어쓰지 못하도록 차단됩니다. DNS 구성을 완료하려면 이 절차의 나머지 단계를 건너뛰고 안내를 따르세요. 아마존 Route 53 별칭 레코드 구성

- 사용자 지정 도메인이 다른 도메인 등록 기관에 등록된 경우 도메인 등록 기관으로 Non-Amazon을 선택합니다.

  a. 도메인 이름을 입력합니다.

  b. 링크 도메인을 선택합니다.



6. DNS 구성 페이지가 표시됩니다.

   - Amazon Route 53 가 DNS 공급자인 경우 이 단계는 선택 사항입니다.

     이 시점에서 App Runner는 필요한 인증서 검증 및 DNS 레코드로 Route 53 도메인을 자동으로 구성했습니다.

     > **Note**
     >
     > 동일한 도메인 이름을 이전에 서비스에서 연결 해제했다면 서비스를 가리키는 DNS 공급자 레코드가 이후에 삭제되지 않았다면 App Runner에서 시도한 자동 구성이 실패했을 수 있습니다. 이 문제를 해결하고 DNS 연결을 완료하려면 DNS 구성 페이지에서 (1) 및 (2) 단계를 진행하여 현재 대상 및 인증서 레코드를 DNS 공급자에게 복사하십시오.

   - 인증서 검증 레코드와 DNS 대상 레코드를 복사하여 DNS 서버에 추가합니다. 그러면 App Runner에서 도메인을 소유하거나 제어하고 있는지 확인할 수 있습니다.

     > **Note**
     >
     > 사용자 지정 도메인 인증서를 자동 갱신하려면 DNS 서버에서 인증서 검증 레코드를 삭제하지 마세요.

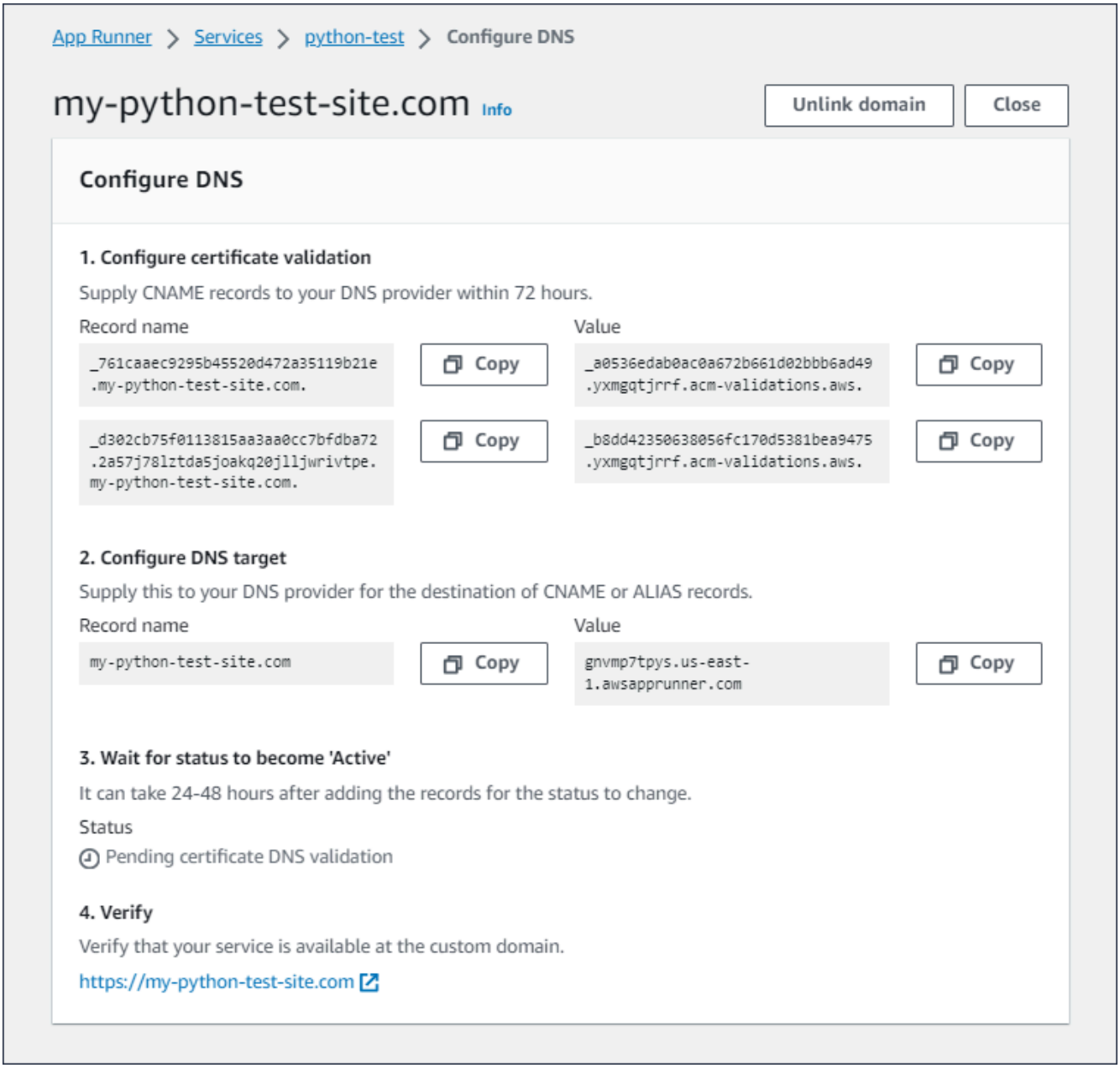
- 인증서 검증 구성에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서의 DNS 검증을 참조하십시오.
- Amazon Route 53 별칭 레코드를 사용하여 DNS 대상을 구성하는 방법에 대한 자세한 내용은 을 참조하십시오the section called “아마존 Route 53 별칭 레코드 구성”.

- Amazon Route 53 이외의 DNS 공급자를 사용하는 경우 다음 단계를 따르십시오.

  - 인증서 검증 레코드와 DNS 대상 레코드를 복사하여 DNS 서버에 추가합니다. 그러면 App Runner에서 도메인을 소유하거나 제어하고 있는지 확인할 수 있습니다.

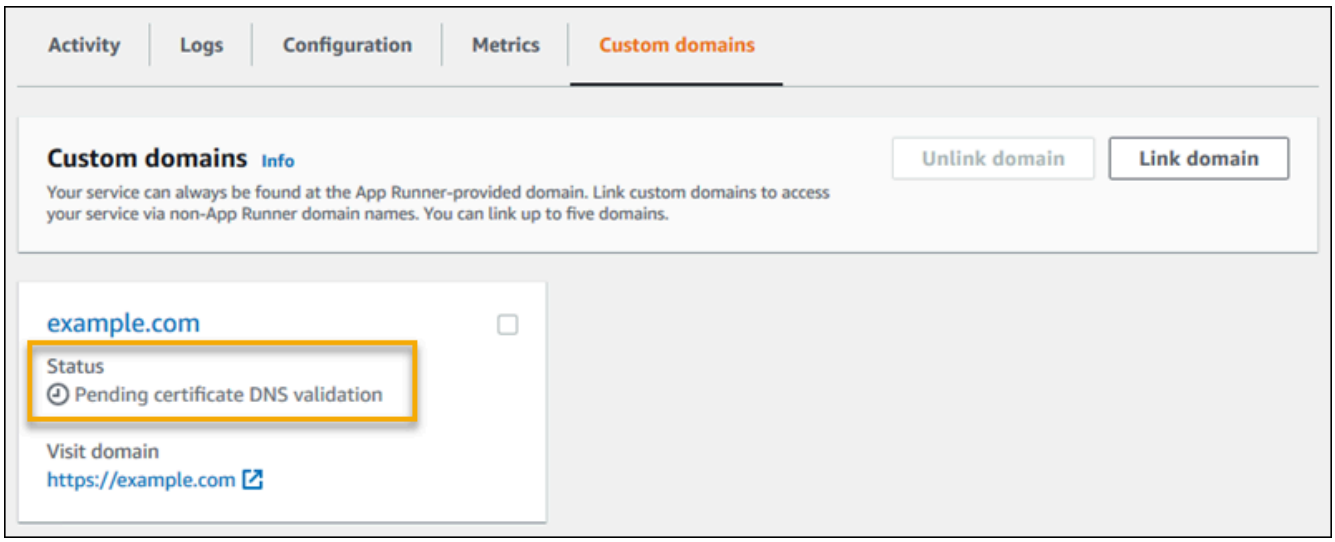
    > **Note**
    >
    > 사용자 지정 도메인 인증서를 자동 갱신하려면 DNS 서버에서 인증서 검증 레코드를 삭제하지 마세요.

    - 인증서 검증 구성에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서의 DNS 검증을 참조하십시오.
    - Shopify, Hover 등과 같은 GoDaddy 다른 등록 기관에서 DNS 대상을 구성하는 방법에 대한 지침은 DNS 대상 추가에 대한 해당 등록 기관의 해당 설명서를 참조하십시오.

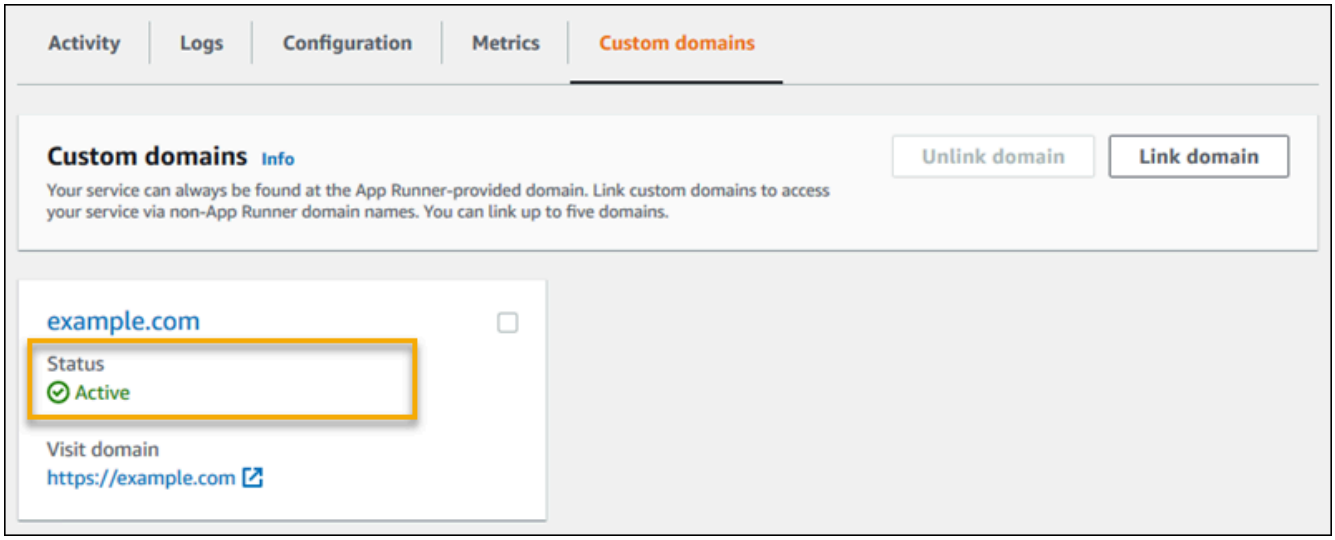


7. 닫기를 선택합니다.

   콘솔에 대시보드가 다시 표시됩니다. 사용자 지정 도메인 탭에는 보류 중인 인증서 DNS 검증 상태에서 방금 연결한 도메인이 표시된 새 타일이 있습니다.



8. 도메인 상태가 Active로 변경되면 도메인을 탐색하여 해당 도메인이 트래픽을 라우팅하는 데 작동하는지 확인하세요.



> **Note**
>
> 커스텀 도메인과 관련된 오류를 해결하는 방법에 대한 지침은 을 참조하십시오. the section called “사용자 지정 도메인 이름”

App Runner 콘솔을 사용하여 커스텀 도메인을 연결 해제 (연결 해제) 하려면

1. 커스텀 도메인 탭에서 연결을 끊으려는 도메인의 타일을 선택한 다음 도메인 연결 해제를 선택합니다.

2. 도메인 연결 해제 대화 상자에서 도메인 연결 해제를 선택하여 작업을 확인합니다.

> **Note**
>
> DNS 서버와의 연결을 끊은 도메인의 레코드를 삭제해야 합니다.

App Runner API or AWS CLI

App Runner API를 사용하여 서비스에 사용자 지정 도메인을 연결하거나 API AWS CLI작업을 호출하십시오. AssociateCustomDomain 호출이 성공하면 서비스와 연결된 사용자 지정 도메인을 설명하는 CustomDomain개체가 반환됩니다. 개체는 `CREATING` 상태를 나타내며 CertificateValidationRecord개체 목록을 포함합니다. 이 호출은 DNS 대상을 구성하는 데 사용할 수 있는 대상 별칭도 반환합니다. 이러한 레코드는 DNS에 추가할 수 있습니다.

App Runner API를 사용하여 서비스에서 사용자 지정 도메인을 분리하려면 API AWS CLI작업을 호출하십시오. DisassociateCustomDomain 호출이 성공하면 서비스와의 연결이 끊어지는 커스텀 도메인을 설명하는 CustomDomain객체가 반환됩니다. 개체에 상태가 표시됩니다. `DELETING`

주제

- 대상 DNS를 위한 Amazon Route 53 별칭 레코드를 구성합니다.

대상 DNS를 위한 Amazon Route 53 별칭 레코드를 구성합니다.

> **Note**
>
> Amazon Route 53이 DNS 공급자인 경우에는 이 절차를 따르지 않아도 됩니다. 이 경우 App Runner는 필수 인증서 검증 및 DNS 레코드로 Route 53 도메인을 자동으로 구성하여 App Runner 웹 애플리케이션에 연결합니다.
> App Runner의 자동 구성 시도가 실패한 경우 이 절차에 따라 DNS 구성을 완료하십시오. 동일한 도메인 이름이 이전에 서비스에서 연결 해제된 경우 서비스를 가리키는 DNS 공급자 레코드가 이후에 삭제되지 않으면 App Runner가 이러한 레코드를 자동으로 덮어쓰지 못하도록 차단됩니다. 이 절차에서는 Route 53 DNS에 수동으로 복사하는 방법을 설명합니다.

Amazon Route 53을 DNS 공급자로 사용하여 트래픽을 앱 러너 서비스로 라우팅할 수 있습니다. 가용성과 확장성이 뛰어난 도메인 이름 시스템 (DNS) 웹 서비스입니다. Amazon Route 53 레코드에는 트

래픽이 앱 러너 서비스로 라우팅되는 방식을 제어하는 설정이 포함되어 있습니다. CNAME 레코드 또는 ALIAS 레코드를 생성합니다. CNAME 레코드와 별칭 레코드를 비교하려면 Amazon Route 53 개발자 안내서의 별칭 레코드와 비 별칭 레코드 간의 선택을 참조하십시오.

> **Note**
>
> Amazon Route 53는 현재 2022년 8월 1일 이후에 생성된 서비스에 대해 별칭 레코드를 지원합니다.

Amazon Route 53 console

Amazon Route 53 별칭 레코드를 구성하려면

1. Route 53 콘솔에 AWS Management Console 로그인하여 엽니다.
2. 탐색 창에서 호스팅 영역(Hosted zones)을 선택합니다.
3. 트래픽을 App Runner 서비스로 라우팅하는 데 사용할 호스팅 영역의 이름을 선택합니다.
4. Create Record Set(레코드 세트 생성)를 선택합니다.
5. 다음 값을 지정하십시오:
   - 라우팅 정책: 해당하는 라우팅 정책을 선택합니다. 자세한 내용은 라우팅 정책 선택을 참조하십시오.
   - 레코드 이름: 트래픽을 App Runner 서비스로 라우팅하는 데 사용할 도메인 이름을 입력합니다. 기본값은 호스팅 영역 이름입니다. 예를 들어 호스팅 영역의 이름이 `example.com` 이고 사용자 환경으로 트래픽을 라우팅하는 `acme.example.com` 데 사용하려는 경우 를 입력합니다`acme`.
   - 값/트래픽 라우팅 대상: App Runner 애플리케이션에 대한 Alias를 선택한 다음 엔드포인트가 있는 지역을 선택합니다. 트래픽을 라우팅하려는 애플리케이션의 도메인 이름을 선택합니다.
   - 레코드 유형: 기본값인 A — IPv4 주소를 그대로 사용합니다.
   - 대상 상태 평가: 기본값인 Yes를 그대로 사용합니다.
6. 레코드 생성을 선택합니다.

생성한 Route 53 별칭 레코드는 60초 이내에 모든 Route 53 서버에 전파됩니다. Route 53 서버가 별칭 레코드와 함께 전파되면 생성한 별칭 레코드의 이름을 사용하여 트래픽을 App Runner 서비스로 라우팅할 수 있습니다.

DNS 변경 사항이 전파되는 데 너무 오래 걸리는 경우 문제를 해결하는 방법에 대한 자세한 내용은 Route 53 및 퍼블릭 리졸버에서 DNS 변경 내용이 전파되는 데 시간이 오래 걸리는 이유를 참조하십시오. .

Amazon Route 53 API or AWS CLI

Amazon Route 53 API를 사용하여 Amazon Route 53 별칭 레코드를 구성하거나 ChangeResourceRecordSetsAPI 작업을 AWS CLI 호출하려면 Route 53의 대상 호스팅 영역 ID에 대해 알아보려면 서비스 엔드포인트를 참조하십시오.

앱 러너 서비스 일시 중지 및 재개

웹 애플리케이션을 일시적으로 비활성화하고 코드 실행을 중지해야 하는 경우 서비스를 일시 중지할 수 있습니다. AWS App Runner App Runner는 서비스에 대한 컴퓨팅 용량을 0으로 줄입니다.

애플리케이션을 다시 실행할 준비가 되면 App Runner 서비스를 재개할 수 있습니다. App Runner는 새로운 컴퓨팅 파워를 프로비저닝하고, 애플리케이션을 배포한 후 애플리케이션을 실행합니다. 애플리케이션 소스는 재배포되지 않으므로 빌드가 필요하지 않습니다. 대신 App Runner는 현재 배포된 버전으로 재개됩니다. 애플리케이션은 해당 App Runner 도메인을 유지합니다.

Important

- 서비스를 일시 중지하면 애플리케이션 상태가 손실됩니다. 예를 들어 코드에서 사용한 임시 스토리지는 모두 손실됩니다. 코드의 경우 서비스를 일시 중지했다가 다시 시작하는 것은 새 서비스에 배포하는 것과 같습니다.
- 코드의 결함 (예: 발견된 버그 또는 보안 문제) 으로 인해 서비스를 일시 중지한 경우 서비스를 재개하기 전에 새 버전을 배포할 수 없습니다.

  따라서 서비스를 계속 실행하고 안정적인 마지막 애플리케이션 버전으로 롤백하는 것이 좋습니다.
- 서비스를 재개하면 App Runner는 서비스를 일시 중지하기 전에 마지막으로 사용한 애플리케이션 버전을 배포합니다. 서비스를 일시 중지한 후 새 소스 버전을 추가한 경우 자동 배포를 선택했더라도 App Runner가 자동으로 배포하지 않습니다. 예를 들어 이미지 리포지토리에 새 이미지 버전이 있거나 코드 리포지토리에 새 커밋이 있다고 가정해 보겠습니다. 이러한 버전은 자동으로 배포되지 않습니다.

새 버전을 배포하려면 App Runner 서비스를 재개한 후 수동 배포를 수행하거나 소스 저장소에 다른 버전을 추가하세요.

일시 중지 및 삭제 비교

App Runner 서비스를 일시 중지하여 일시적으로 비활성화하십시오. 컴퓨팅 리소스만 종료되고 저장된 데이터 (예: 애플리케이션 버전의 컨테이너 이미지) 는 그대로 유지됩니다. 서비스를 빠르게 재개할 수 있습니다. 애플리케이션을 새 컴퓨팅 리소스에 배포할 준비가 된 것입니다. App Runner 도메인은 동일하게 유지됩니다.

App Runner 서비스를 삭제하여 영구적으로 제거하세요. 저장된 데이터가 삭제됩니다. 서비스를 다시 만들어야 하는 경우 App Runner는 소스를 다시 가져와야 하며, 코드 저장소인 경우 빌드해야 합니다. 웹 애플리케이션은 새로운 App Runner 도메인을 가져옵니다.

서비스가 일시 중지된 경우

서비스를 일시 중지하고 일시 중지됨 상태가 되면 API 호출 또는 콘솔 작업을 비롯한 작업 요청에 다르게 응답합니다. 서비스가 일시 중지된 경우에도 런타임에 영향을 주는 방식으로 서비스의 정의 또는 구성을 수정하지 않는 App Runner 작업을 수행할 수 있습니다. 즉, 실행 중인 서비스의 동작, 규모 또는 기타 특성이 변경되는 작업으로 인해 일시 중지된 서비스에서는 해당 작업을 수행할 수 없습니다.

다음 목록은 일시 중지된 서비스에서 수행할 수 있는 API 작업과 수행할 수 없는 API 작업에 대한 정보를 제공합니다. 동등한 콘솔 작업도 마찬가지로 허용되거나 거부됩니다.

일시 중지된 서비스에서 수행할 수 있는 작업

- *`List*`*및 *`Describe*`* 작업 — 정보만 읽는 작업.
- *`DeleteService`*— 서비스는 언제든지 삭제할 수 있습니다.
- *`TagResource`*, *`UntagResource`* — 태그는 서비스와 연결되어 있지만 서비스 정의의 일부가 아니며 런타임 동작에 영향을 주지 않습니다.

일시 중지된 서비스에서 수행할 수 없는 작업

- *`StartDeployment`*작업 (또는 콘솔을 사용한 수동 배포)
- *`UpdateService`*(또는 콘솔을 사용한 구성 변경, 태그 지정 변경 제외)
- *`CreateCustomDomainAssociations`*, *`DeleteCustomDomainAssociations`*

- *CreateConnection*, *DeleteConnection*

서비스 일시 중지 및 재개

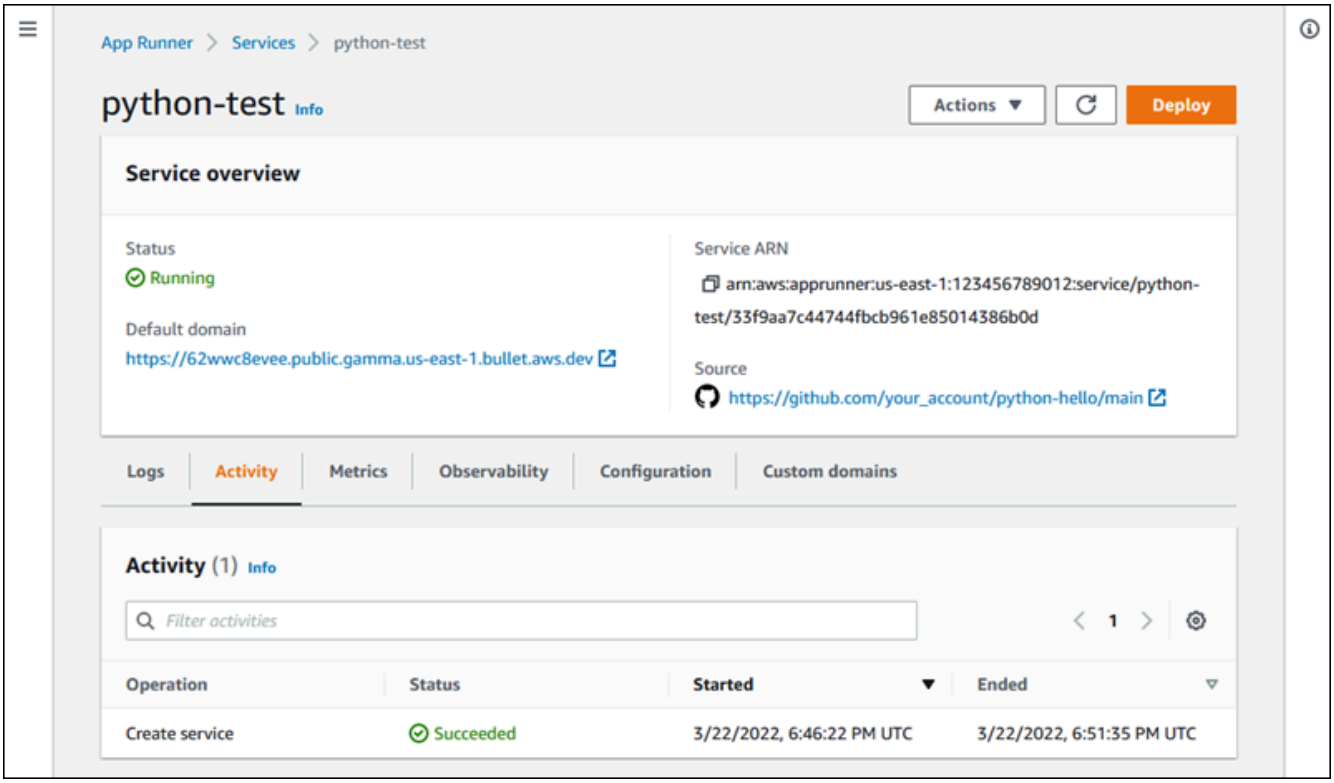
다음 방법 중 하나를 사용하여 App Runner 서비스를 일시 중지하고 재개하십시오.

App Runner console

App Runner 콘솔을 사용하여 서비스를 일시 중지하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 작업을 선택한 다음 일시 중지를 선택합니다.

   서비스 대시보드 페이지에서 서비스 상태가 작업 진행 중으로 변경된 다음 일시 중지됨으로 변경됩니다. 이제 서비스가 일시 중지되었습니다.

App Runner 콘솔을 사용하여 서비스를 재개하려면

1. 작업을 선택한 다음 [다시 시작] 을 선택합니다.

   서비스 대시보드 페이지에서 서비스 상태가 작업 진행 중으로 변경됩니다.
2. 서비스가 재개될 때까지 기다리세요. 서비스 대시보드 페이지에서 서비스 상태가 다시 Running으로 변경됩니다.
3. 서비스 재개가 성공했는지 확인하려면 서비스 대시보드 페이지에서 App Runner 도메인 값을 선택합니다. 서비스 웹 사이트의 URL입니다. 웹 애플리케이션이 제대로 실행되고 있는지 확인하세요.

App Runner API or AWS CLI

App Runner API를 사용하여 서비스를 일시 중지하거나 API AWS CLI작업을 호출하십시오. PauseService 호출 시 서비스 객체가 표시된 `"Status": "OPERATION_IN_PROGRESS"` 성공적인 응답이 반환되면 App Runner는 서비스를 일시 중지하기 시작합니다.

App Runner API를 사용하여 서비스를 재개하려면 API 작업을 호출하세요. AWS CLIResumeService 호출 시 서비스 객체가 표시된 `"Status": "OPERATION_IN_PROGRESS"` 상태에서 성공적인 응답이 반환되면 App Runner는 서비스를 재개하기 시작합니다.

앱 러너 서비스 삭제

서비스에서 실행 중인 웹 애플리케이션을 종료하려는 경우 AWS App Runner 서비스를 삭제하면 됩니다. 서비스를 삭제하면 실행 중인 웹 서비스가 중지되고, 기본 리소스가 제거되고, 관련 데이터가 삭제됩니다.

다음 이유 중 하나 이상의 이유로 App Runner 서비스를 삭제해야 할 수 있습니다.

- 웹 애플리케이션이 더 이상 필요하지 않습니다. 예를 들어, 사용 중지되었거나 사용을 마친 개발 버전인 경우를 들 수 있습니다.
- App Runner 서비스 할당량에 도달했습니다. — 동일한 AWS 리전 서비스에 새 서비스를 만들고 싶은데 계정에 연결된 할당량에 도달했습니다. 자세한 정보는 the section called “앱 러너 리소스 할당량”을 참조하세요.
- 보안 또는 개인정보 보호 고려사항 — App Runner가 서비스용으로 저장하는 데이터를 삭제하기를 원할 것입니다.

일시 중지 및 삭제 비교

App Runner 서비스를 일시 중지하여 일시적으로 비활성화하십시오. 컴퓨팅 리소스만 종료되고 저장된 데이터 (예: 애플리케이션 버전의 컨테이너 이미지) 는 그대로 유지됩니다. 서비스를 빠르게 재개할 수 있습니다. 애플리케이션을 새 컴퓨팅 리소스에 배포할 준비가 된 것입니다. App Runner 도메인은 동일하게 유지됩니다.

App Runner 서비스를 삭제하여 영구적으로 제거하세요. 저장된 데이터가 삭제됩니다. 서비스를 다시 만들어야 하는 경우 App Runner는 소스를 다시 가져와야 하며, 코드 저장소인 경우 빌드해야 합니다. 웹 애플리케이션은 새로운 App Runner 도메인을 가져옵니다.

App Runner는 무엇을 삭제하나요?

서비스를 삭제하면 App Runner는 일부 관련 항목을 삭제하고 다른 항목은 삭제하지 않습니다. 다음 목록은 세부 정보를 제공합니다.

App Runner가 삭제하는 항목:

- 컨테이너 이미지 — 배포한 이미지 또는 App Runner가 소스 코드에서 빌드한 이미지의 복사본입니다. 앱 러너 소유의 AWS 계정 내부 데이터를 사용하여 Amazon Elastic Container 레지스트리 (Amazon ECR) 에 저장됩니다.
- 서비스 구성 — App Runner 서비스와 관련된 구성 설정입니다. 앱 러너 소유의 AWS 계정 내부 데이터를 사용하여 Amazon DynamoDB에 저장됩니다.

앱 러너가 삭제하지 않는 항목:

- 연결 — 서비스와 연결된 연결이 있을 수 있습니다. App Runner 연결은 여러 App Runner 서비스 간에 공유될 수 있는 별도의 리소스입니다. 연결이 더 이상 필요하지 않은 경우 명시적으로 삭제할 수 있습니다. 자세한 정보는 the section called “연결”을 참조하세요.
- 커스텀 도메인 인증서 — 커스텀 도메인을 App Runner 서비스에 연결하는 경우 App Runner는 내부적으로 도메인 유효성을 추적하는 인증서를 생성합니다. 이러한 정보는 AWS Certificate Manager (ACM) 에 저장됩니다. App Runner는 도메인이 서비스에서 연결이 해제된 후 또는 서비스가 삭제된 후 7일 동안 인증서를 삭제하지 않습니다. 자세한 정보는 the section called “사용자 지정 도메인 이름”을 참조하세요.

서비스 삭제

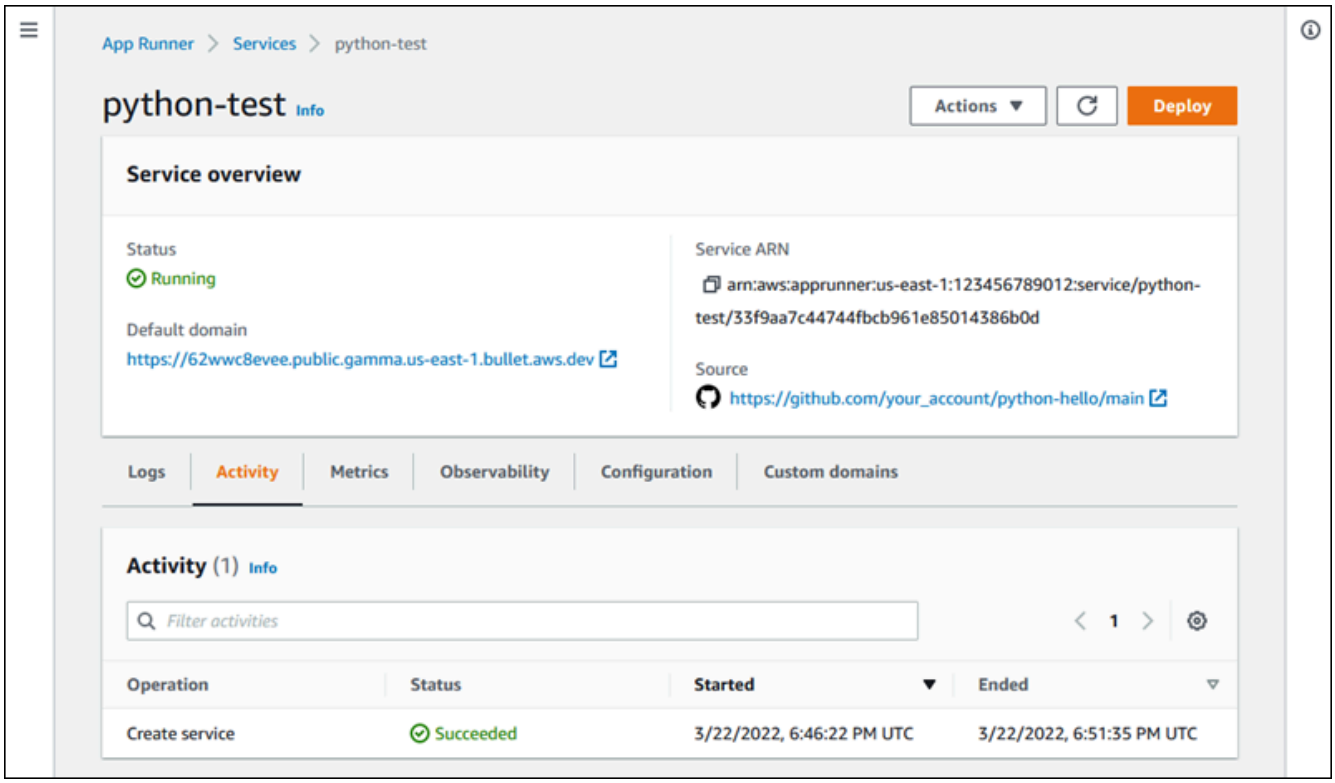
다음 방법 중 하나를 사용하여 App Runner 서비스를 삭제합니다.

App Runner console

App Runner 콘솔을 사용하여 서비스를 삭제하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 작업을 선택한 후 삭제를 선택합니다.

   콘솔은 서비스 페이지로 이동합니다. 삭제된 서비스는 작업 진행 중 상태로 표시된 후 목록에서 사라집니다. 이제 서비스가 삭제되었습니다.

App Runner API or AWS CLI

앱 러너 AWS CLI API를 사용하여 서비스를 삭제하려면 DeleteServiceAPI 작업을 호출하십시오. 호출 시 서비스 객체가 표시된 `"Status": "OPERATION_IN_PROGRESS"` 상태에서 성공적인 응답이 반환되면 App Runner는 서비스 삭제를 시작합니다.

환경 변수 참조

App Runner를 사용하면 서비스를 생성하거나 서비스를 업데이트할 때 시크릿과 구성을 서비스의 환경 변수로 참조할 수 있습니다.

일반 텍스트의 제한 시간, 재시도 횟수 등 중요하지 않은 구성 데이터를 키-값 쌍으로 참조할 수 있습니다. 일반 텍스트로 참조하는 구성 데이터는 암호화되지 않으며 App Runner 서비스 구성 및 애플리케이션 로그에서 다른 사용자가 볼 수 있습니다.

> **Note**
>
> 보안상의 이유로 App Runner 서비스에서 일반 텍스트의 민감한 데이터를 참조하지 마세요.

민감한 데이터를 환경 변수로 참조

App Runner는 민감한 데이터를 서비스의 환경 변수로 안전하게 참조할 수 있도록 지원합니다. 참조하려는 민감한 데이터를 AWS Systems Manager Parameter Store에 저장하는 것을 고려해 보세요. AWS Secrets Manager 그런 다음 App Runner 콘솔에서 또는 API를 호출하여 서비스에서 환경 변수로 안전하게 참조할 수 있습니다. 이렇게 하면 암호 및 매개변수 관리를 애플리케이션 코드 및 서비스 구성에서 효과적으로 분리하여 App Runner에서 실행되는 애플리케이션의 전반적인 보안을 개선합니다.

> **Note**
>
> App Runner는 Secrets Manager 및 SSM 파라미터 저장소를 환경 변수로 참조하는 것에 대해서는 비용을 청구하지 않습니다. 하지만 Secrets Manager와 SSM 파라미터 스토어 사용에 대해서는 표준 요금을 지불해야 합니다.
> 요금에 대한 자세한 내용은 다음을 참조하십시오.
>
> - AWS Secrets Manager 가격
> - AWS SSM 파라미터 스토어 가격

민감한 데이터를 환경 변수로 참조하는 프로세스는 다음과 같습니다.

1. API 키, 데이터베이스 자격 증명, 데이터베이스 연결 매개변수 또는 애플리케이션 버전과 같은 민감한 데이터를 매개변수 저장소 중 하나 또는 매개변수 저장소에 암호 AWS Secrets Manager 또는 AWS Systems Manager 매개변수로 저장합니다.
2. 앱 러너가 Secrets Manager 및 SSM 파라미터 스토어에 저장된 비밀과 파라미터에 액세스할 수 있도록 인스턴스 역할의 IAM 정책을 업데이트하세요. 자세한 내용은 Permissions 단원을 참조하십시오.
3. 이름을 지정하고 Amazon 리소스 이름 (ARN) 을 제공하여 보안 정보와 파라미터를 환경 변수로 안전하게 참조할 수 있습니다. 서비스를 생성하거나 서비스 구성을 업데이트할 때 환경 변수를 추가할 수 있습니다. 다음 옵션 중 하나를 사용하여 환경 변수를 추가할 수 있습니다.
   - 앱 러너 콘솔
   - 앱 러너 API
   - `apprunner.yaml` 구성 파일

   > **Note**
   >
   > App Runner 서비스를 만들거나 업데이트할 때 환경 변수의 `PORT` 이름으로 할당할 수 없습니다. App Runner 서비스용으로 예약된 환경 변수입니다.

   암호 및 매개변수를 참조하는 방법에 대한 자세한 내용은 환경 변수 관리를 참조하십시오.

> **Note**
>
> App Runner는 암호 및 매개변수 ARN에 대한 참조만 저장하므로 App Runner 서비스 구성 및 애플리케이션 로그에서 다른 사용자는 민감한 데이터를 볼 수 없습니다.

고려 사항

- Parameter Store에서 AWS Secrets Manager AWS Systems Manager 또는 Parameter Store에 있는 비밀과 파라미터에 액세스할 수 있는 적절한 권한으로 인스턴스 역할을 업데이트해야 합니다. 자세한 내용은 Permissions 단원을 참조하십시오.
- AWS Systems Manager 파라미터 스토어가 시작 또는 업데이트하려는 서비스와 AWS 계정 동일한 위치에 있는지 확인하십시오. 현재는 계정 간에 SSM 파라미터 스토어 파라미터를 참조할 수 없습니다.

- 암호와 매개변수 값이 순환되거나 변경되더라도 App Runner 서비스에서 자동으로 업데이트되지 않습니다. App Runner는 배포 중에 암호와 매개변수만 가져오므로 App Runner 서비스를 재배포하십시오.
- App Runner 서비스의 SDK를 통해 AWS Systems Manager 파라미터 스토어를 직접 AWS Secrets Manager 호출할 수도 있습니다.
- 오류를 방지하려면 환경 변수로 참조할 때 다음 사항을 확인하세요.
  - 암호의 올바른 ARN을 지정합니다.
  - 파라미터의 올바른 이름 또는 ARN을 지정합니다.

권한

AWS Secrets Manager 또는 SSM 파라미터 스토어에 저장된 시크릿 및 파라미터를 참조할 수 있게 하려면 인스턴스 역할의 IAM 정책에 Secrets Manager 및 SSM 파라미터 스토어에 액세스할 수 있는 적절한 권한을 추가합니다.

> **Note**
>
> App Runner는 사용자의 허가 없이는 계정의 리소스에 액세스할 수 없습니다. IAM 정책 업데이트를 통해 권한을 제공합니다.

다음 정책 템플릿을 사용하여 IAM 콘솔에서 인스턴스 역할을 업데이트할 수 있습니다. 특정 요구 사항에 맞게 이러한 정책 템플릿을 수정할 수 있습니다. 인스턴스 역할 업데이트에 대한 자세한 내용은 IAM 사용 설명서의 역할 수정을 참조하십시오.

> **Note**
>
> 환경 변수를 생성할 때 App Runner 콘솔에서 다음 템플릿을 복사할 수도 있습니다.

다음 템플릿을 인스턴스 역할에 복사하여 시크릿을 참조할 권한을 추가하세요. AWS Secrets Manager

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

다음 템플릿을 인스턴스 역할에 복사하여 AWS Systems Manager파라미터 저장소의 참조 파라미터에 대한 권한을 추가하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter_name>"
      ]
    }
  ]
}
```

환경 변수 관리

다음 방법 중 하나를 사용하여 App Runner 서비스의 환경 변수를 관리합니다.

- the section called “앱 러너 콘솔”
- the section called “앱 러너 API 또는 AWS CLI”

앱 러너 콘솔

App Runner 콘솔에서 서비스를 만들거나 서비스를 업데이트할 때 환경 변수를 추가할 수 있습니다.

환경 변수 추가

환경 변수를 추가하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스를 생성 중인지 업데이트 중인지에 따라 다음 단계 중 하나를 수행하십시오.
   - 새 서비스를 생성하는 경우 App Runner 서비스 생성을 선택하고 서비스 구성으로 이동합니다.
   - 기존 서비스를 업데이트하는 경우 업데이트하려는 서비스를 선택하고 서비스의 구성 탭으로 이동합니다.
3. 서비스 설정에서 환경 변수 - 선택 사항으로 이동합니다.
4. 요구 사항에 따라 다음 옵션 중 하나를 선택하십시오.
   - 환경 변수 소스에서 일반 텍스트를 선택하고 환경 변수 이름 및 환경 변수 값에 각각 해당 키-값 쌍을 입력합니다.

     > **Note**
     >
     > 중요하지 않은 데이터를 참조하려면 일반 텍스트를 선택합니다. 이 데이터는 암호화되지 않으며 App Runner 서비스 구성 및 애플리케이션 로그에서 다른 사용자가 볼 수 있습니다.

   - 서비스에 환경 변수로 저장된 시크릿을 참조하려면 환경 변수 소스에서 Secrets Manager를 선택합니다. AWS Secrets Manager 환경 변수 이름 및 환경 변수 값에서 참조하는 암호의 환경 변수 이름과 Amazon 리소스 이름 (ARN) 을 각각 입력합니다.
   - SSM 파라미터 스토어에 저장된 파라미터를 서비스의 환경 변수로 참조하려면 환경 변수 소스에서 SSM 파라미터 스토어를 선택합니다. 환경 변수 이름 및 환경 변수 값에서 참조하는 매개 변수의 환경 변수 이름과 ARN을 각각 입력합니다.

     > **Note**
     >
     > - App Runner 서비스를 만들거나 업데이트할 때는 환경 변수 `PORT` 이름으로 할당할 수 없습니다. App Runner 서비스용으로 예약된 환경 변수입니다.
     > - SSM 파라미터 스토어 파라미터가 시작하려는 서비스와 AWS 리전 동일한 경우, 전체 Amazon 리소스 이름 (ARN) 또는 파라미터 이름을 지정할 수 있습니다. 파라미터가 다른 지역에 있는 경우 전체 ARN을 지정해야 합니다.

- 참조하는 파라미터가 시작 또는 업데이트 중인 서비스와 동일한 계정에 있는지 확인하세요. 현재는 계정 간에 SSM 파라미터 스토어 파라미터를 참조할 수 없습니다.

5. 다른 환경 변수를 참조하려면 환경 변수 추가를 선택합니다.
6. IAM 정책 템플릿을 확장하여 AWS Secrets Manager 및 SSM 파라미터 스토어에 제공된 IAM 정책 템플릿을 보고 복사할 수 있습니다. 인스턴스 역할의 IAM 정책을 필요한 권한으로 아직 업데이트하지 않은 경우에만 이 작업을 수행하면 됩니다. 자세한 내용은 Permissions 단원을 참조하십시오.

환경 변수 제거

환경 변수를 삭제하기 전에 애플리케이션 코드가 동일한 내용을 반영하도록 업데이트되었는지 확인하십시오. 애플리케이션 코드가 업데이트되지 않으면 App Runner 서비스가 실패할 수 있습니다.

환경 변수를 제거하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 업데이트하려는 서비스의 구성 탭으로 이동합니다.
3. 서비스 설정에서 환경 변수 - 선택사항으로 이동합니다.
4. 제거하려는 환경 변수 옆의 제거를 선택합니다. 삭제를 확인하는 메시지가 나타납니다.
5. 삭제를 선택합니다.

앱 러너 API 또는 AWS CLI

Secrets Manager 및 SSM 파라미터 스토어에 저장된 민감한 데이터를 서비스의 환경 변수로 추가하여 참조할 수 있습니다.

> **Note**
>
> 앱 러너가 Secrets Manager 및 SSM 파라미터 스토어에 저장된 비밀과 파라미터에 액세스할 수 있도록 인스턴스 역할의 IAM 정책을 업데이트하세요. 자세한 내용은 Permissions 단원을 참조하십시오.

시크릿 및 구성을 환경 변수로 참조하려면

1. Secrets Manager 또는 SSM 파라미터 저장소에서 시크릿 또는 컨피그레이션을 생성합니다.

다음 예제는 SSM 파라미터 저장소를 사용하여 시크릿과 파라미터를 생성하는 방법을 보여줍니다.

Example 시크릿 생성 - 요청

다음 예제는 데이터베이스 자격 증명을 나타내는 시크릿을 생성하는 방법을 보여줍니다.

```
aws secretsmanager create-secret \
–name DevRdsCredentials \
–description "Rds credentials for development account." \
–secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
```

Example 시크릿 생성 - 응답

```
arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
```

Example 구성 생성 - 요청

다음 예제는 RDS 연결 문자열을 나타내는 파라미터를 만드는 방법을 보여줍니다.

```
aws systemsmanager put-parameter \
–name DevRdsConnectionString \
–value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
–type "String" \
–description "Rds connection string for development account."
```

Example 구성 생성 - 응답

```
arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
```

2. Secrets Manager 및 SSM 파라미터 스토어에 저장된 시크릿과 컨피그레이션을 환경 변수로 추가하여 참조할 수 있습니다. App Runner 서비스를 생성하거나 업데이트할 때 환경 변수를 추가할 수 있습니다.

   다음 예시는 코드 기반 및 이미지 기반 App Runner 서비스에서 암호와 구성을 환경 변수로 참조하는 방법을 보여줍니다.

Example 이미지 기반 앱 러너 서비스를 위한 입력.json 파일

```
{
  "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 이미지 기반 앱 러너 서비스 — 요청

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

Example 이미지 기반 앱 러너 서비스 — 응답

```
{
...
      "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 코드 기반 앱 러너 서비스를 위한.json 파일 입력

```
{
  "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 코드 기반 앱 러너 서비스 — 요청

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

Example 코드 기반 앱 러너 서비스 — 응답

```
{
...
   "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. 추가된 apprunner.yaml 비밀이 반영되도록 모델이 업데이트되었습니다.

   다음은 업데이트된 apprunner.yaml 모델의 예시입니다.

   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"
```

앱 러너를 통한 네트워킹

이 장에서는 AWS App Runner 서비스의 네트워킹 구성을 설명합니다.

이 장에서는 다음 내용을 배우게 됩니다.

- 수신 트래픽을 프라이빗 및 퍼블릭 엔드포인트에 맞게 구성하는 방법 자세한 내용은 수신 트래픽에 대한 네트워킹 구성 설정을 참조하십시오.
- Amazon VPC에서 실행 중인 다른 애플리케이션에 액세스하도록 발신 트래픽을 구성하는 방법 자세한 내용은 발신 트래픽에 대한 VPC 액세스 활성화를 참조하십시오.

> **Note**
>
> App Runner는 현재 퍼블릭 수신 트래픽에 대해서만 이중 스택 (IPv4 및 IPv6) 주소 유형을 지원합니다. 발신 트래픽과 비공개 수신 트래픽의 경우 IPv4만 지원됩니다.

주제

용어

네트워크 트래픽을 요구 사항에 맞게 사용자 지정하는 방법을 알아보려면 이 장에서 사용되는 다음 용어를 이해해 보겠습니다.

일반 용어

Amazon VPC (가상 사설 클라우드) 와 연결하는 데 필요한 것이 무엇인지 알아보려면 다음 용어를 이해해 보겠습니다.

- VPC: Amazon VPC는 논리적으로 격리된 가상 네트워크로, 리소스 배치, 연결 및 보안을 포함하여 가상 네트워킹 환경을 완벽하게 제어할 수 있습니다. 자체 데이터 센터에서 운영하는 기존 네트워크와 매우 유사한 가상 네트워크입니다.

- VPC 인터페이스 엔드포인트: VPC 인터페이스 엔드포인트, 즉 AWS PrivateLink 리소스는 VPC를 엔드포인트 서비스에 연결합니다. VPC 인터페이스 엔드포인트를 생성하여 Network Load Balancer를 사용하여 트래픽을 분산하는 엔드포인트 서비스에 트래픽을 전송합니다. 엔드포인트 서비스로 전송되는 트래픽은 DNS를 사용하여 확인됩니다.
- 지역: 각 지역은 App Runner 서비스를 호스팅할 수 있는 별도의 지리적 영역입니다.
- 가용 영역: 가용 영역은 지역 내 격리된 위치입니다. AWS 이중 전원, 네트워킹 및 연결을 갖춘 하나 이상의 개별 데이터 센터입니다. 가용 영역을 통해 프로덕션 애플리케이션은 고가용성, 내결함성 및 확장성을 갖춥니다.
- 서브넷: 서브넷은 VPC의 IP 주소 범위입니다. 서브넷은 단일 가용 영역에 상주해야 합니다. 지정된 서브넷에서 AWS 리소스를 시작할 수 있습니다. 인터넷에 연결되어야 하는 리소스에는 퍼블릭 서브넷을 사용하고, 인터넷에 연결되지 않는 리소스에는 프라이빗 서브넷을 사용하십시오.
- 보안 그룹: 보안 그룹은 연결된 리소스에 도달하거나 나가는 것이 허용되는 트래픽을 제어합니다. 보안 그룹은 각 서브넷의 AWS 리소스를 보호하기 위한 추가 보안 계층을 제공하여 네트워크 트래픽을 더 잘 제어할 수 있도록 합니다. VPC를 생성할 경우 VPC는 기본 보안 그룹과 함께 제공됩니다. 각 VPC 대해 추가 보안 그룹을 생성할 수 있습니다. 보안 그룹은 보안 그룹이 생성된 VPC 내의 리소스와만 연결할 수 있습니다.
- 이중 스택: 이중 스택은 IPv4 및 IPv6 엔드포인트 모두에서 네트워크 트래픽을 지원하는 주소 유형입니다.

발신 트래픽 구성과 관련된 용어

VPC 커넥터

VPC 커넥터는 앱 러너 서비스가 프라이빗 Amazon VPC에서 실행되는 애플리케이션에 액세스할 수 있도록 하는 앱 러너 리소스입니다.

수신 트래픽 구성과 관련된 용어

Amazon VPC 내에서만 서비스에 비공개로 액세스할 수 있는 방법을 알아보려면 다음 용어를 이해해 보겠습니다.

- VPC 인그레스 연결: VPC 인그레스 연결은 들어오는 트래픽에 대한 앱 러너 엔드포인트를 제공하는 앱 러너 리소스입니다. App Runner 콘솔에서 들어오는 트래픽에 대해 프라이빗 엔드포인트를 선택하면 App Runner는 VPC 인그레스 연결 리소스를 백그라운드에서 할당합니다. VPC 인그레스 연결 리소스는 앱 러너 서비스를 Amazon VPC의 VPC 인터페이스 엔드포인트에 연결합니다.

> **Note**
>
> App Runner API를 사용하는 경우 VPC 인그레스 연결 리소스가 자동으로 생성되지 않습니다.

- 프라이빗 엔드포인트: 프라이빗 엔드포인트는 Amazon VPC 내에서만 액세스할 수 있도록 수신 네트워크 트래픽을 구성하기 위해 선택하는 App Runner 콘솔 옵션입니다.

수신 트래픽에 대한 네트워킹 구성 설정

프라이빗 또는 퍼블릭 엔드포인트에서 들어오는 트래픽을 수신하도록 서비스를 구성할 수 있습니다.

퍼블릭 엔드포인트는 기본 구성입니다. 퍼블릭 인터넷에서 들어오는 모든 트래픽에 서비스를 엽니다. 또한 서비스에 맞는 IPv4 (인터넷 프로토콜 버전 4) 또는 이중 스택 (IPv4 및 IPv6) 주소 유형 중에서 유연하게 선택할 수 있습니다.

프라이빗 엔드포인트는 Amazon VPC의 트래픽만 App Runner 서비스에 액세스할 수 있도록 허용합니다. 이는 App Runner 서비스를 위한 AWS PrivateLink 리소스인 VPC 인터페이스 엔드포인트를 설정함으로써 달성됩니다. 따라서 Amazon VPC와 앱 러너 서비스 간에 프라이빗 연결이 생성됩니다.

> **Note**
>
> App Runner는 현재 퍼블릭 엔드포인트에 대해서만 이중 스택 (IPv4 및 IPv6) 주소 유형을 지원합니다. 프라이빗 엔드포인트의 경우 IPv4만 지원됩니다.

다음은 수신 트래픽에 대한 네트워크 구성 설정의 일부로 다루어지는 항목입니다.

- Amazon VPC 내에서만 서비스를 비공개로 사용할 수 있도록 수신 트래픽을 구성하는 방법 자세한 내용은 수신 트래픽에 대한 프라이빗 엔드포인트 활성화를 참조하십시오.
- 이중 스택 주소 유형에서 인터넷 트래픽을 수신하도록 서비스를 구성하는 방법 자세한 내용은 퍼블릭 수신 트래픽에 대한 이중 스택 활성화를 참조하십시오.

헤더

App Runner를 사용하면 애플리케이션으로 들어오는 트래픽의 원본 소스 IPv4 및 IPv6 주소에 액세스할 수 있습니다. 요청 헤더를 할당하여 원본 소스 IP 주소를 보존할 수 있습니다. `X-Forwarded-For` 이렇게 하면 애플리케이션이 필요할 때 원본 소스 IP 주소를 가져올 수 있습니다.

> **Note**
>
> 서비스가 프라이빗 엔드포인트를 사용하도록 구성된 경우 `X-Forwarded-For` 요청 헤더를 사용하여 원본 소스 IP 주소에 액세스할 수 없습니다. 사용하면 잘못된 값을 검색합니다.

들어오는 트래픽에 대한 프라이빗 엔드포인트 활성화

AWS App Runner 서비스를 생성하면 기본적으로 인터넷을 통해 서비스에 액세스할 수 있습니다. 하지만 App Runner 서비스를 비공개로 설정하고 Amazon VPC (가상 사설 클라우드) 내에서만 액세스할 수 있도록 설정할 수도 있습니다.

App Runner 서비스를 비공개로 사용하면 들어오는 트래픽을 완벽하게 제어하여 보안을 한층 강화할 수 있습니다. 이는 내부 API, 회사 웹 애플리케이션 또는 더 높은 수준의 개인 정보 보호 및 보안이 필요하거나 특정 규정 준수 요구 사항을 충족해야 하는 아직 개발 중인 애플리케이션을 실행하는 등 다양한 사용 사례에서 유용합니다.

> **Note**
>
> App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
>
> 모범 사례를 포함하여 인프라 보안 및 보안 그룹에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 다음 주제: 보안 그룹을 사용하여 네트워크 트래픽 제어 및 AWS 리소스로의 트래픽 제어를 참조하십시오.

App Runner 서비스가 비공개인 경우 Amazon VPC 내에서 서비스에 액세스할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스 또는 VPN 연결은 필요하지 않습니다.

> **Note**
>
> App Runner는 현재 퍼블릭 수신 트래픽에 대해서만 이중 스택 (IPv4 및 IPv6) 주소 유형을 지원합니다. 발신 트래픽과 비공개 수신 트래픽의 경우 IPv4만 지원됩니다.

고려 사항

- App Runner의 VPC 인터페이스 엔드포인트를 설정하기 전에 가이드의 고려 사항을 검토하십시오.AWS PrivateLink
- VPC 엔드포인트 정책은 앱 러너에 지원되지 않습니다. 기본적으로 VPC 인터페이스 엔드포인트를 통해 App Runner에 대한 전체 액세스가 허용됩니다. 또는 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하여 VPC 인터페이스 엔드포인트를 통해 App Runner로 들어오는 트래픽을 제어할 수 있습니다.
- App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우, WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
- 프라이빗 엔드포인트를 활성화하면 VPC에서만 서비스에 액세스할 수 있고 인터넷에서는 액세스할 수 없습니다.
- 가용성을 높이려면 VPC 인터페이스 엔드포인트에 대해 서로 다른 가용 영역에서 최소 두 개의 서브넷을 선택하는 것이 좋습니다. 서브넷을 하나만 사용하는 것은 권장하지 않습니다.
- 동일한 VPC 인터페이스 엔드포인트를 사용하여 VPC의 여러 App Runner 서비스에 액세스할 수 있습니다.

이 섹션에 사용되는 용어에 대한 자세한 내용은 용어를 참조하십시오.

권한

다음은 프라이빗 엔드포인트를 활성화하는 데 필요한 권한 목록입니다.

- ec2: CreateTags
- ec2: CreateVpcEndpoint
- ec2: ModifyVpcEndpoint
- ec2: DeleteVpcEndpoints

- ec2: DescribeSubnets
- ec2: DescribeVpcEndpoints
- ec2: DescribeVpcs

VPC 인터페이스 엔드포인트

VPC 인터페이스 엔드포인트는 Amazon VPC를 엔드포인트 서비스에 연결하는 AWS PrivateLink리소스입니다. VPC 인터페이스 엔드포인트를 전달하여 App Runner 서비스에 액세스할 수 있는 Amazon VPC를 지정할 수 있습니다. VPC 인터페이스 엔드포인트를 만들려면 다음을 지정합니다.

- 연결을 활성화하기 위한 Amazon VPC.
- 보안 그룹을 추가합니다. 기본적으로 보안 그룹은 VPC 인터페이스 엔드포인트에 할당됩니다. 사용자 지정 보안 그룹을 연결하여 들어오는 네트워크 트래픽을 추가로 제어할 수 있습니다.
- 서브넷을 추가합니다. 가용성을 높이려면 App Runner 서비스에 액세스할 각 가용 영역에 대해 최소 두 개의 서브넷을 선택하는 것이 좋습니다. 네트워크 인터페이스 엔드포인트는 VPC 인터페이스 엔드포인트에 대해 활성화한 각 서브넷에 생성됩니다. App Runner로 향하는 트래픽의 진입점 역할을 하는 요청자 관리 네트워크 인터페이스입니다. 요청자 관리형 네트워크 인터페이스는 AWS 서비스가 사용자를 대신하여 VPC에서 생성하는 네트워크 인터페이스입니다.
- API를 사용하는 경우 App Runner VPC 인터페이스 엔드포인트를 추가하세요. `Servicename` 예:

```
com.amazonaws.region.apprunner.requests
```

다음 AWS 서비스 중 하나를 사용하여 VPC 인터페이스 엔드포인트를 생성할 수 있습니다.

- 앱 러너 콘솔. 자세한 내용은 프라이빗 엔드포인트 관리를 참조하십시오.
- Amazon VPC 콘솔 또는 API, 그리고 AWS Command Line Interface ()AWS CLI자세한 내용은 AWS PrivateLinkAWS PrivateLink 가이드의 AWS 서비스 액세스를 참조하십시오.

> **Note**
>
> 요금을 기준으로AWS PrivateLink 사용하는 각 VPC 인터페이스 엔드포인트에 대해 요금이 부과됩니다. 따라서 비용 효율성을 높이기 위해 동일한 VPC 인터페이스 엔드포인트를 사용하여 VPC 내의 여러 App Runner 서비스에 액세스할 수 있습니다. 하지만 격리를 강화하려면 App Runner 서비스마다 다른 VPC 인터페이스 엔드포인트를 연결하는 것이 좋습니다.

VPC Ingress Connection

VPC 인그레스 연결은 들어오는 트래픽에 대한 앱 러너 엔드포인트를 지정하는 앱 러너 리소스입니다. App Runner 콘솔에서 들어오는 트래픽에 대해 프라이빗 엔드포인트를 선택하면 App Runner는 VPC 인그레스 연결 리소스를 백그라운드에서 할당합니다. Amazon VPC의 트래픽만 App Runner 서비스에 액세스하도록 허용하려면 이 옵션을 선택합니다. VPC 인그레스 연결 리소스는 앱 러너 서비스를 Amazon VPC의 VPC 인터페이스 엔드포인트에 연결합니다. API 작업을 사용하여 수신 트래픽에 대한 네트워크 설정을 구성하는 경우에만 VPC 인그레스 연결 리소스를 생성할 수 있습니다. VPC 인그레스 연결 리소스를 생성하는 방법에 대한 자세한 내용은 API 레퍼런스를 참조하십시오 CreateVpcIngressConnection.AWS App Runner

> **Note**
>
> 앱 러너의 VPC 인그레스 연결 리소스 하나를 Amazon VPC의 VPC 인터페이스 엔드포인트 하나에 연결할 수 있습니다. 또한 각 App Runner 서비스에 대해 하나의 VPC 인그레스 연결 리소스만 생성할 수 있습니다.

프라이빗 엔드포인트

프라이빗 엔드포인트는 Amazon VPC에서 들어오는 트래픽만 수신하려는 경우 선택할 수 있는 App Runner 콘솔 옵션입니다. App Runner 콘솔에서 프라이빗 엔드포인트 옵션을 선택하면 VPC 인터페이스 엔드포인트를 구성하여 서비스를 VPC에 연결할 수 있습니다. App Runner는 백그라운드에서 사용자가 구성한 VPC 인터페이스 엔드포인트에 VPC 인그레스 연결 리소스를 할당합니다.

> **Note**
>
> 프라이빗 엔드포인트에는 IPv4 네트워크 트래픽만 지원됩니다.

요약

Amazon VPC의 트래픽만 App Runner 서비스에 액세스하도록 허용하여 서비스를 비공개로 설정하십시오. 이를 위해서는 앱 러너 또는 Amazon VPC를 사용하여 선택한 Amazon VPC에 대한 VPC 인터페이스 엔드포인트를 생성합니다. App Runner 콘솔에서 들어오는 트래픽에 대해 프라이빗 엔드포인트를 활성화하면 VPC 인터페이스 엔드포인트를 생성합니다. 그러면 App Runner가 자동으로 VPC 인그레스 연결 리소스를 생성하고 VPC 인터페이스 엔드포인트 및 앱 러너 서비스에 연결합니다. 이렇게 하면 선택한 VPC의 트래픽만 App Runner 서비스에 액세스할 수 있도록 하는 비공개 서비스 연결이 생성됩니다.

프라이빗 엔드포인트 관리

다음 방법 중 하나를 사용하여 수신 트래픽의 프라이빗 엔드포인트를 관리하십시오.

- the section called “앱 러너 콘솔”
- the section called “앱 러너 API 또는 AWS CLI”

> **Note**
>
> App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우, WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
> 모범 사례를 포함하여 인프라 보안 및 보안 그룹에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 다음 주제: 보안 그룹을 사용하여 네트워크 트래픽 제어 및 AWS 리소스로의 트래픽 제어를 참조하십시오.

앱 러너 콘솔

App Runner 콘솔을 사용하여 서비스를 만들거나 나중에 구성을 업데이트할 때 수신 트래픽을 구성하도록 선택할 수 있습니다.

수신 트래픽을 구성하려면 다음 중 하나를 선택합니다.

- 퍼블릭 엔드포인트: 인터넷을 통해 모든 서비스에서 서비스에 액세스할 수 있도록 합니다. 기본적으로 퍼블릭 엔드포인트가 선택됩니다.
- 프라이빗 엔드포인트: Amazon VPC 내에서만 App Runner 서비스에 액세스할 수 있도록 하기 위함입니다.

> **Note**
>
> 현재 App Runner는 퍼블릭 엔드포인트에 대해서만 IPv6를 지원합니다. IPv6 엔드포인트는 Amazon VPC (가상 사설 클라우드) 에서 호스팅되는 앱 러너 서비스에 지원되지 않습니다. 듀얼 스택 퍼블릭 엔드포인트를 사용하는 서비스를 프라이빗 엔드포인트로 업데이트하는 경우

App Runner 서비스는 기본적으로 IPv4 엔드포인트의 트래픽만 지원하며 IPv6 엔드포인트의 트래픽은 수신하지 못합니다.

프라이빗 엔드포인트를 활성화합니다.

액세스하려는 Amazon VPC의 VPC 인터페이스 엔드포인트와 연결하여 프라이빗 엔드포인트를 활성화합니다. VPC 인터페이스 엔드포인트를 새로 만들거나 기존 VPC 인터페이스 엔드포인트를 선택할 수 있습니다.

VPC 인터페이스 엔드포인트를 만들려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스 구성의 네트워킹 섹션으로 이동합니다.
3. 들어오는 네트워크 트래픽에 대해 프라이빗 엔드포인트를 선택합니다. VPC 인터페이스 엔드포인트를 사용하여 VCP에 연결하는 옵션이 열립니다.
4. 새 엔드포인트 생성을 선택합니다. 새 VPC 인터페이스 엔드포인트 생성 대화 상자가 열립니다.
5. VPC 인터페이스 엔드포인트의 이름을 입력합니다.
6. 사용 가능한 드롭다운 목록에서 필요한 VPC 인터페이스 엔드포인트를 선택합니다.
7. 드롭다운 목록에서 보안 그룹을 선택합니다. 보안 그룹을 추가하면 VPC 인터페이스 엔드포인트에 추가 보안 계층이 제공됩니다. 두 개 이상의 보안 그룹을 선택하는 것이 좋습니다. 보안 그룹을 선택하지 않는 경우 App Runner는 VPC 인터페이스 엔드포인트에 기본 보안 그룹을 할당합니다. 보안 그룹 규칙이 App Runner 서비스와 통신하려는 리소스를 차단하지 않도록 하세요. 보안 그룹 규칙은 App Runner 서비스와 상호작용하는 리소스를 허용해야 합니다.

> **Note**
>
> App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
> 모범 사례를 포함하여 인프라 보안 및 보안 그룹에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 다음 주제: 보안 그룹을 사용하여 네트워크 트래픽 제어 및 AWS 리소스로의 트래픽 제어를 참조하십시오.

8. 드롭다운 목록에서 필요한 서브넷을 선택합니다. App Runner 서비스에 액세스할 각 가용 영역에 대해 최소 두 개의 서브넷을 선택하는 것이 좋습니다.
9. (선택 사항) Add new tag (새 태그 추가) 를 선택하고 태그 키와 태그 값을 입력합니다.
10. 생성을 선택합니다. 서비스 구성 페이지가 열리고 상단 표시줄에 VPC 인터페이스 엔드포인트가 성공적으로 생성되었다는 메시지가 표시됩니다.

기존 VPC 인터페이스 엔드포인트를 선택하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스 구성의 네트워킹 섹션으로 이동합니다.
3. 들어오는 네트워크 트래픽에 대해 프라이빗 엔드포인트를 선택합니다. VPC 인터페이스 엔드포인트를 사용하여 VPC에 연결하는 옵션이 열립니다. 사용 가능한 VPC 인터페이스 엔드포인트 목록이 표시됩니다.
4. VPC 인터페이스 엔드포인트 아래에 나열된 필수 VPC 인터페이스 엔드포인트를 선택합니다.
5. 다음을 선택하여 서비스를 생성합니다. App Runner는 프라이빗 엔드포인트를 활성화합니다.

> **Note**
>
> 서비스를 생성한 후 필요에 따라 VPC 인터페이스 엔드포인트와 연결된 보안 그룹 및 서브넷을 편집하도록 선택할 수 있습니다.

프라이빗 엔드포인트의 세부 정보를 확인하려면 서비스로 이동하여 구성 탭 아래의 네트워킹 섹션을 확장하십시오. VPC 및 프라이빗 엔드포인트와 연결된 VPC 인터페이스 엔드포인트의 세부 정보를 보여줍니다.

VPC 인터페이스 엔드포인트 업데이트

App Runner 서비스가 생성된 후 프라이빗 엔드포인트와 연결된 VPC 인터페이스 엔드포인트를 편집할 수 있습니다.

> **Note**
>
> 엔드포인트 이름과 VPC 필드는 업데이트할 수 없습니다.

VPC 인터페이스 엔드포인트를 업데이트하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스로 이동하여 왼쪽 패널에서 네트워킹 구성을 선택합니다.
3. 각 서비스와 관련된 VPC 인터페이스 엔드포인트를 보려면 수신 트래픽을 선택합니다.
4. 편집하려는 VPC 인터페이스 엔드포인트를 선택합니다.
5. 편집을 선택합니다. VPC 인터페이스 엔드포인트를 편집하기 위한 대화 상자가 열립니다.
6. 필요한 보안 그룹과 서브넷을 선택하고 업데이트를 클릭합니다. VPC 인터페이스 엔드포인트 세부 정보가 표시된 페이지가 열리고 상단 표시줄에 VPC 인터페이스 엔드포인트가 성공적으로 업데이트되었다는 메시지가 표시됩니다.

> **Note**
>
> App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우, WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
> 모범 사례를 포함하여 인프라 보안 및 보안 그룹에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 다음 주제: 보안 그룹을 사용하여 네트워크 트래픽 제어 및 AWS 리소스로의 트래픽 제어를 참조하십시오.

VPC 인터페이스 엔드포인트 삭제

App Runner 서비스에 비공개로 액세스할 수 없도록 하려면 수신 트래픽을 Public으로 설정하면 됩니다. Public으로 변경하면 Private 엔드포인트가 제거되지만 VPC 인터페이스 엔드포인트는 삭제되지 않습니다.

VPC 인터페이스 엔드포인트를 삭제하려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스로 이동하여 왼쪽 패널에서 네트워킹 구성을 선택합니다.
3. 각 서비스와 관련된 VPC 인터페이스 엔드포인트를 보려면 수신 트래픽을 선택합니다.

> **Note**
>
> VPC 인터페이스 엔드포인트를 삭제하기 전에 서비스를 업데이트하여 연결된 모든 서비스에서 제거하십시오.

4. 삭제를 선택합니다.

   VPC 인터페이스 엔드포인트에 연결된 서비스가 있는 경우 VPC 인터페이스 엔드포인트를 삭제할 수 없음 메시지를 받습니다. VPC 인터페이스 엔드포인트에 연결된 서비스가 없는 경우 삭제를 확인하는 메시지를 받게 됩니다.

5. 삭제를 선택합니다. 들어오는 트래픽에 대한 네트워크 구성 페이지가 열리고 상단 표시줄에 VPC 인터페이스 엔드포인트가 성공적으로 삭제되었다는 메시지가 표시됩니다.

앱 러너 API 또는 AWS CLI

Amazon VPC 내에서만 액세스할 수 있는 애플리케이션을 App Runner에 배포할 수 있습니다.

서비스를 비공개로 설정하는 데 필요한 권한에 대한 자세한 내용은 을 참조하십시오. the section called “권한”

> **Note**
>
> 현재 App Runner는 퍼블릭 엔드포인트에 대해서만 IPv6를 지원합니다. IPv6 엔드포인트는 Amazon VPC (가상 사설 클라우드) 에서 호스팅되는 앱 러너 서비스에 지원되지 않습니다. 듀얼 스택 퍼블릭 엔드포인트를 사용하는 서비스를 프라이빗 엔드포인트로 업데이트하는 경우 App Runner 서비스는 기본적으로 IPv4 엔드포인트의 트래픽만 지원하며 IPv6 엔드포인트의 트래픽은 수신하지 못합니다.

Amazon VPC에 대한 프라이빗 서비스 연결을 생성하려면

1. App Runner에 연결할 AWS PrivateLink 리소스인 VPC 인터페이스 엔드포인트를 생성합니다. 이렇게 하려면 애플리케이션과 연결할 서브넷과 보안 그룹을 지정해야 합니다. 다음은 VPC 인터페이스 엔드포인트를 생성하는 예제입니다.

> **Note**
>
> App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우, WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다. 이는 현재 요청 소스 IP 데이터를 WAF와 연결된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. 따라서 WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다.
> 모범 사례를 포함하여 인프라 보안 및 보안 그룹에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 다음 주제: 보안 그룹을 사용하여 네트워크 트래픽 제어 및 AWS 리소스로의 트래픽 제어를 참조하십시오.

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. CLI를 통해 CreateService또는 UpdateService앱 러너 API 작업을 사용하여 VPC 인터페이스 엔드포인트를 참조합니다. 공개적으로 액세스할 수 없도록 서비스를 구성하세요. `NetworkConfiguration`파라미터 `IsPubliclyAccessible` `False` `IngressConfiguration` 멤버에 로 설정합니다. 다음은 VPC 인터페이스 엔드포인트를 참조하는 예제입니다.

   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. `create-vpc-ingress-connection`API 작업을 호출하여 App Runner용 VPC 인그레스 연결 리소스를 생성하고 이를 이전 단계에서 생성한 VPC 인터페이스 엔드포인트와 연결합니다. 지정

된 VPC에서 서비스에 액세스하는 데 사용되는 도메인 이름을 반환합니다. 다음은 VPC 인그레스 연결 리소스를 생성하는 예제입니다.

Example 요청

```
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 응답

```
{
   "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 인그레스 연결 업데이트

VPC 인그레스 연결 리소스를 업데이트할 수 있습니다. VPC 인그레스 연결이 업데이트되려면 다음 상태 중 하나여야 합니다.

- 사용 가능
- 생성 실패
- 업데이트 실패

다음은 VPC 인그레스 연결 리소스 업데이트 예제입니다.

Example 요청

```
aws apprunner update-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example 응답

```
{
    "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 인그레스 연결 삭제

Amazon VPC에 대한 프라이빗 연결이 더 이상 필요하지 않은 경우 VPC 인그레스 연결 리소스를 삭제할 수 있습니다.

VPC 인그레스 연결을 삭제하려면 다음 상태 중 하나여야 합니다.

- 사용 가능
- 생성 실패
- 업데이트 실패
- 삭제 실패

다음은 VPC 인그레스 연결을 삭제하는 예제입니다.

Example 요청

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example 응답

```
{
   "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>
}
```

다음 App Runner API 작업을 사용하여 서비스의 프라이빗 인바운드 트래픽을 관리하세요.

- CreateVpcIngressConnection— 새 VPC 인그레스 연결 리소스를 생성합니다. App Runner는 App Runner 서비스를 Amazon VPC 엔드포인트에 연결하려는 경우 이 리소스가 필요합니다.
- ListVpcIngressConnections— 계정과 연결된 AWS App Runner VPC 인그레스 연결 엔드포인트 목록을 반환합니다. AWS
- DescribeVpcIngressConnection— AWS App Runner VPC 인그레스 연결 리소스의 전체 설명을 반환합니다.
- UpdateVpcIngressConnection— AWS App Runner VPC 인그레스 연결 리소스를 업데이트합니다.
- DeleteVpcIngressConnection— 앱 러너 서비스와 연결된 앱 러너 VPC 인그레스 연결 리소스를 삭제합니다.

앱 러너 API 사용에 대한 자세한 내용은 앱 러너 API 참조 가이드를 참조하십시오.

퍼블릭 수신 트래픽에 IPv6 활성화

서비스가 IPv6 주소 또는 IPv4 및 IPv6 주소 모두에서 들어오는 네트워크 트래픽을 수신하도록 하려면 퍼블릭 엔드포인트의 이중 스택 주소 유형을 선택하십시오. 새 애플리케이션을 만들 때는 서비스 구성 > 네트워킹 섹션에서 이 설정을 찾을 수 있습니다. 앱 러너 콘솔 또는 앱 러너 API를 사용하여 IPv6를 활성화하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “퍼블릭 엔드포인트의 듀얼 스택 관리”

IPv6 도입에 대한 자세한 내용은 IPv6 on을 참조하십시오. AWSAWS

앱 러너는 퍼블릭 앱 러너 서비스 엔드포인트에 대해서만 이중 스택을 지원합니다. 모든 앱 러너 프라이빗 서비스의 경우 IPv4만 지원됩니다.

> **Note**
>
> IP 주소 유형을 이중 스택으로 설정하고 네트워크 구성을 퍼블릭에서 프라이빗 엔드포인트로 변경하면 App Runner에서 자동으로 주소 유형을 IPv4로 변경합니다. 이는 App Runner가 퍼블릭 엔드포인트에 대해서만 IPv6를 지원하기 때문입니다.

IPv4와 IPv6에 대한 배경 정보를 알아보세요.

인터넷을 통해 네트워크 트래픽을 라우팅하는 데 일반적으로 사용되는 IPv4 네트워크 계층은 32비트 주소 체계를 사용합니다. 이 주소 공간은 제한적이며 네트워크 장치 수가 많으면 고갈될 수 있습니다. 이러한 이유로 네트워크 주소 변환 (NAT) 은 일반적으로 단일 공용 네트워크 주소를 통해 여러 IPv4 주소를 라우팅하는 데 사용됩니다.

인터넷 프로토콜의 최신 버전인 IPv6는 IPv4를 기반으로 하며 128비트 주소 지정 체계를 통해 주소 공간을 확장합니다. IPv6를 사용하면 거의 무제한의 연결 장치로 네트워크를 구축할 수 있습니다. 네트워크 주소가 방대하기 때문에 IPv6에는 NAT가 필요하지 않습니다.

IPv4 엔드포인트는 들어오는 IPv6 트래픽을 수신할 수 없고 그 반대의 경우도 있기 때문에 IPv4 및 IPv6 엔드포인트는 서로 호환되지 않습니다. 듀얼 스택은 IPv4 및 IPv6 네트워크 트래픽을 동시에 지원할 수 있는 편리한 솔루션을 제공합니다.

퍼블릭 수신 트래픽을 위한 듀얼 스택 관리

다음 방법 중 하나를 사용하여 퍼블릭 수신 트래픽의 이중 스택 주소 유형을 관리하십시오.

- the section called “앱 러너 콘솔”
- the section called “앱 러너 API 또는 AWS CLI”

앱 러너 콘솔

App Runner 콘솔을 사용하여 서비스를 만들거나 나중에 구성을 업데이트할 때 들어오는 인터넷 트래픽에 대해 이중 스택 주소 유형을 선택할 수 있습니다.

이중 스택 주소를 활성화하려면 다음을 입력하십시오.

1. 서비스를 만들거나 업데이트할 때 서비스 구성에서 네트워킹 섹션을 확장하십시오.
2. 들어오는 네트워크 트래픽에 대해 퍼블릭 엔드포인트를 선택합니다. 퍼블릭 엔드포인트 IP 주소 유형 옵션이 열립니다.
3. 퍼블릭 엔드포인트 IP 주소 유형을 확장하여 다음 IP 주소 유형을 확인하십시오.
    - IPv4
    - 이중 스택 (IPv4 및 IPv6)

> **Note**
>
> 퍼블릭 엔드포인트 IP 주소 유형을 확장하여 선택하지 않으면 App Runner는 IPv4를 기본 구성으로 할당합니다.

4. 이중 스택 (IPv4 및 IPv6) 을 선택합니다.
5. 서비스를 생성하는 경우 다음을 선택한 다음 생성 및 배포를 선택합니다. 서비스를 업데이트하려면 변경 내용 저장을 선택해도 됩니다.

   서비스가 배포되면 애플리케이션이 IPv4 및 IPv6 엔드포인트 모두에서 네트워크 트래픽을 수신하기 시작합니다.

> **Note**
>
> 현재 App Runner는 퍼블릭 엔드포인트에 대해서만 IPv6을 지원합니다. IPv6 엔드포인트는 Amazon VPC (가상 사설 클라우드) 에서 호스팅되는 앱 러너 서비스에 지원되지 않습니다. 듀얼 스택 퍼블릭 엔드포인트를 사용하는 서비스를 프라이빗 엔드포인트로 업데이트하는 경우 App Runner 서비스는 기본적으로 IPv4 엔드포인트의 트래픽만 지원하며 IPv6 엔드포인트의 트래픽은 수신하지 못합니다.

주소 유형을 변경하려면

1. 단계에 따라 서비스를 업데이트하고 네트워킹으로 이동합니다.
2. 수신 네트워크 트래픽에서 퍼블릭 엔드포인트 IP 주소 유형으로 이동하여 필요한 주소 유형을 선택합니다.
3. 변경 사항 저장를 선택합니다. 선택한 내용으로 서비스가 업데이트됩니다.

앱 러너 API 또는 AWS CLI

CreateService또는 UpdateServiceApp Runner API 작업을 호출할 때는 `NetworkConfiguration` 매개변수의 `IpAddressType` 멤버를 사용하여 주소 유형을 지정합니다. 지정할 수 있는 지원되는 값은 `IPv4` 및 입니다. `DUAL_STACK` 서비스가 IPv4 및 IPv6 엔드포인트에서 인터넷 트래픽을 수신할지 `DUAL_STACK` 여부를 지정합니다. 에 값을 지정하지 않으면 기본적으로 IPv4가 `IpAddressType` 적용됩니다.

다음은 이중 스택을 IP 주소로 사용하여 서비스를 생성하는 예제입니다. 이 예제에서는 `input.json` 파일을 호출합니다.

Example 이중 스택을 지원하는 서비스 생성 요청

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

Example **input.json**의 콘텐츠

```
{
  "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 응답

```
{
  "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**
>
> 현재 App Runner는 퍼블릭 엔드포인트에 대해서만 IPv6를 지원합니다. IPv6 엔드포인트는 Amazon VPC (가상 사설 클라우드) 에서 호스팅되는 앱 러너 서비스에 지원되지 않습니다. 듀얼 스택 퍼블릭 엔드포인트를 사용하는 서비스를 프라이빗 엔드포인트로 업데이트하는 경우

App Runner 서비스는 기본적으로 IPv4 엔드포인트의 트래픽만 지원하며 IPv6 엔드포인트의 트래픽은 수신하지 못합니다.

API 매개변수에 대한 자세한 내용은 을 참조하십시오. NetworkConfiguration

발신 트래픽에 대한 VPC 액세스 활성화

기본적으로 AWS App Runner 애플리케이션은 퍼블릭 엔드포인트로 메시지를 보낼 수 있습니다. 여기에는 자체 솔루션과 기타 모든 공개 웹 사이트 또는 웹 서비스가 포함됩니다. AWS 서비스애플리케이션은 Amazon VPC (가상 사설 클라우드) 에서 VPC에서 실행되는 애플리케이션의 퍼블릭 엔드포인트로 메시지를 보낼 수도 있습니다. 환경을 시작할 때 VPC를 구성하지 않으면 App Runner는 기본 VPC (공개) 를 사용합니다.

사용자 지정 VPC에서 환경을 시작하여 나가는 트래픽에 대한 네트워킹 및 보안 설정을 사용자 지정할 수 있습니다. Amazon VPC (가상 사설 클라우드) 에서 프라이빗 VPC에서 실행되는 애플리케이션에 AWS App Runner 서비스가 액세스할 수 있도록 할 수 있습니다. 이렇게 하면 애플리케이션이 Amazon VPC (Virtual Private Cloud) 에서 호스팅되는 다른 애플리케이션에 연결하여 메시지를 보낼 수 있습니다. Amazon RDS 데이터베이스 ElastiCache, Amazon 및 프라이빗 VPC에서 호스팅되는 기타 프라이빗 서비스를 예로 들 수 있습니다.

VPC 커넥터

앱 러너 콘솔에서 VPC 커넥터라는 VPC 엔드포인트를 생성하여 서비스를 VPC와 연결할 수 있습니다. VPC 커넥터를 만들려면 VPC, 하나 이상의 서브넷, 선택적으로 하나 이상의 보안 그룹을 지정합니다. VPC 커넥터를 구성한 후 하나 이상의 App Runner 서비스와 함께 사용할 수 있습니다.

일회성 지연

아웃바운드 트래픽을 위한 사용자 지정 VPC 커넥터로 App Runner 서비스를 구성하는 경우 2~5분의 일회성 시작 지연 시간이 발생할 수 있습니다. 시작 프로세스는 VPC Connector가 다른 리소스에 연결할 준비가 될 때까지 기다린 후 서비스 상태를 Running으로 설정합니다. 서비스를 처음 생성할 때 사용자 지정 VPC 커넥터로 서비스를 구성하거나 이후에 서비스 업데이트를 수행하여 구성할 수 있습니다.

단, 다른 서비스에 동일한 VPC 커넥터 구성을 재사용하면 지연 시간이 발생하지 않습니다. VPC 커넥터 구성은 보안 그룹과 서브넷 조합을 기반으로 합니다. 지정된 VPC 커넥터 구성에서 지연 시간은 VPC 커넥터 하이퍼플레인 ENI (엘라스틱 네트워크 인터페이스) 를 처음 생성할 때 한 번만 발생합니다.

사용자 지정 VPC 커넥터 및 하이퍼플레인에 대한 자세한 정보 AWS

App Runner의 VPC 커넥터는 Network Load Balancer, NAT Gateway, AWS 등 여러 AWS 리소스를 지원하는 내부 Amazon 네트워크 시스템인 AWS 하이퍼플레인을 기반으로 합니다. PrivateLink AWS 하이퍼플레인 기술은 높은 수준의 공유와 함께 높은 처리량과 짧은 지연 시간을 제공합니다. 하이퍼플레인 ENI는 VPC 커넥터를 생성하여 서비스에 연결할 때 서브넷에 생성됩니다. VPC 커넥터 구성은 보안 그룹과 서브넷 조합을 기반으로 하며, 여러 App Runner 서비스에서 동일한 VPC 커넥터를 참조할 수 있습니다. 따라서 기본 하이퍼플레인 ENI는 앱 러너 서비스 전체에서 공유됩니다. 요청 로드를 처리하는 데 필요한 작업 수를 늘리더라도 이러한 공유가 가능하므로 VPC의 IP 공간을 보다 효율적으로 활용할 수 있습니다. 자세한 내용은 AWS 컨테이너 블로그의 AWS App Runner VPC 네트워킹 심층 분석을 참조하십시오.

서브넷

각 서브넷은 특정 가용 영역에 있습니다. 고가용성을 위해 3개 이상의 가용 영역에서 서브넷을 선택하는 것이 좋습니다. 해당 지역의 가용 영역이 3개 미만인 경우 지원되는 모든 가용 영역에서 서브넷을 선택하는 것이 좋습니다.

VPC의 서브넷을 선택할 때는 퍼블릭 서브넷이 아닌 프라이빗 서브넷을 선택해야 합니다. VPC 커넥터를 만들면 앱 러너 서비스가 각 서브넷에 하이퍼플레인 ENI를 생성하기 때문입니다. 각 하이퍼플레인 ENI에는 사설 IP 주소만 할당되며 키 태그가 지정됩니다. AWSAppRunnerManaged 퍼블릭 서브넷을 선택하면 App Runner 서비스를 실행할 때 오류가 발생합니다. 하지만 서비스가 인터넷 또는 기타 AWS 서비스퍼블릭에 있는 일부 서비스에 액세스해야 하는 경우에는 을 참조하십시오. the section called “서브넷 선택 시 고려 사항 ”

서브넷 선택 시 고려 사항

- 서비스를 VPC에 연결하면 아웃바운드 트래픽이 퍼블릭 인터넷에 액세스할 수 없습니다. 애플리케이션의 모든 아웃바운드 트래픽은 서비스가 연결된 VPC를 통해 전달됩니다. VPC의 모든 네트워킹 규칙은 애플리케이션의 아웃바운드 트래픽에 적용됩니다. 즉, 서비스가 퍼블릭 인터넷 및 AWS API에 액세스할 수 없습니다. 액세스 권한을 얻으려면 다음 중 하나를 수행하십시오.
  - NAT 게이트웨이를 통해 서브넷을 인터넷에 연결합니다.
  - AWS 서비스 액세스하려는 VPC 엔드포인트를 설정합니다. 를 사용하여 Amazon VPC 내에서 서비스를 유지할 수 있습니다. AWS PrivateLink
- 일부의 일부 가용 영역은 App Runner 서비스와 함께 사용할 수 있는 서브넷을 AWS 리전 지원하지 않습니다. 이러한 가용 영역에서 서브넷을 선택하면 서비스가 생성되거나 업데이트되지 않습니다. 이러한 상황에서 App Runner는 지원되지 않는 서브넷 및 가용 영역을 가리키는 자세한 오류 메시지

를 제공합니다. 이 경우 요청에서 지원되지 않는 서브넷을 제거하여 문제를 해결한 다음 다시 시도하십시오.

보안 그룹

App Runner가 지정된 AWS 서브넷에서 액세스하는 데 사용하는 보안 그룹을 선택적으로 지정할 수 있습니다. 보안 그룹을 지정하지 않는 경우 App Runner는 VPC의 기본 보안 그룹을 사용합니다. 기본 보안 그룹은 모든 아웃바운드 트래픽을 허용합니다.

보안 그룹을 추가하면 VCP 커넥터에 추가 보안 계층이 제공되므로 네트워크 트래픽을 더 잘 제어할 수 있습니다. VPC Connector는 애플리케이션으로부터의 아웃바운드 통신에만 사용됩니다. 아웃바운드 규칙을 사용하여 원하는 대상 엔드포인트와 통신할 수 있습니다. 또한 대상 리소스와 연결된 모든 보안 그룹에 적절한 인바운드 규칙이 있는지 확인해야 합니다. 그렇지 않으면 이러한 리소스는 VPC Connector 보안 그룹에서 오는 트래픽을 수락할 수 없습니다.

> **Note**
>
> 서비스를 VPC와 연결할 때 다음 트래픽은 영향을 받지 않습니다.
>
> - 인바운드 트래픽 — 애플리케이션이 수신하는 수신 메시지는 관련 VPC의 영향을 받지 않습니다. 메시지는 서비스와 연결된 퍼블릭 도메인 이름을 통해 라우팅되며 VPC와 상호작용하지 않습니다.
> - App Runner 트래픽 — App Runner는 사용자를 대신하여 소스 코드 및 이미지 가져오기, 로그 푸시, 암호 검색과 같은 여러 작업을 관리합니다. 이러한 작업으로 생성되는 트래픽은 VPC를 통해 라우팅되지 않습니다.

Amazon VPC와 AWS App Runner 통합하는 방법에 대한 자세한 내용은 앱 러너 VPC AWS 네트워킹을 참조하십시오.

> **Note**
>
> 나가는 트래픽의 경우 앱 러너는 현재 IPv4만 지원합니다.

VPC 액세스 관리

> **Note**
>
> 서비스에 대한 아웃바운드 트래픽 VPC 커넥터를 생성하는 경우 이후 서비스 시작 프로세스에서 1회의 지연이 발생합니다. 새 서비스를 생성할 때 또는 이후에 서비스 업데이트를 통해 새 서비스에 대해 이 구성을 설정할 수 있습니다. 자세한 내용은 이 일회성 지연 가이드의 App Runner를 사용한 네트워킹 장을 참조하십시오.

다음 방법 중 하나를 사용하여 App Runner 서비스에 대한 VPC 액세스를 관리합니다.

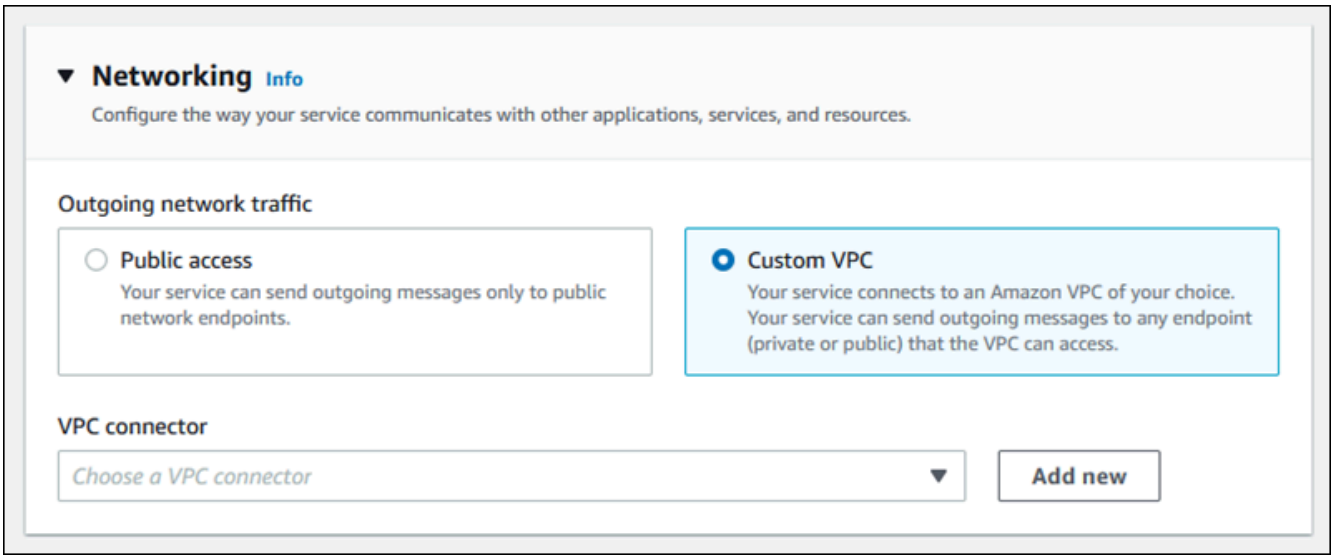
App Runner console

App Runner 콘솔을 사용하여 서비스를 생성하거나 나중에 구성을 업데이트할 때 발신 트래픽을 구성하도록 선택할 수 있습니다. 콘솔 페이지에서 네트워킹 구성 섹션을 찾아보세요. 발신 네트워크 트래픽의 경우 다음 중에서 선택하십시오.

- 퍼블릭 액세스: 서비스를 다른 사람의 퍼블릭 엔드포인트와 연결합니다. AWS 서비스
- 사용자 지정 VPC: 서비스를 Amazon VPC의 VPC와 연결합니다. 애플리케이션은 Amazon VPC에 호스팅된 다른 애플리케이션에 연결하여 메시지를 전송할 수 있습니다.

사용자 지정 VPC를 활성화하려면

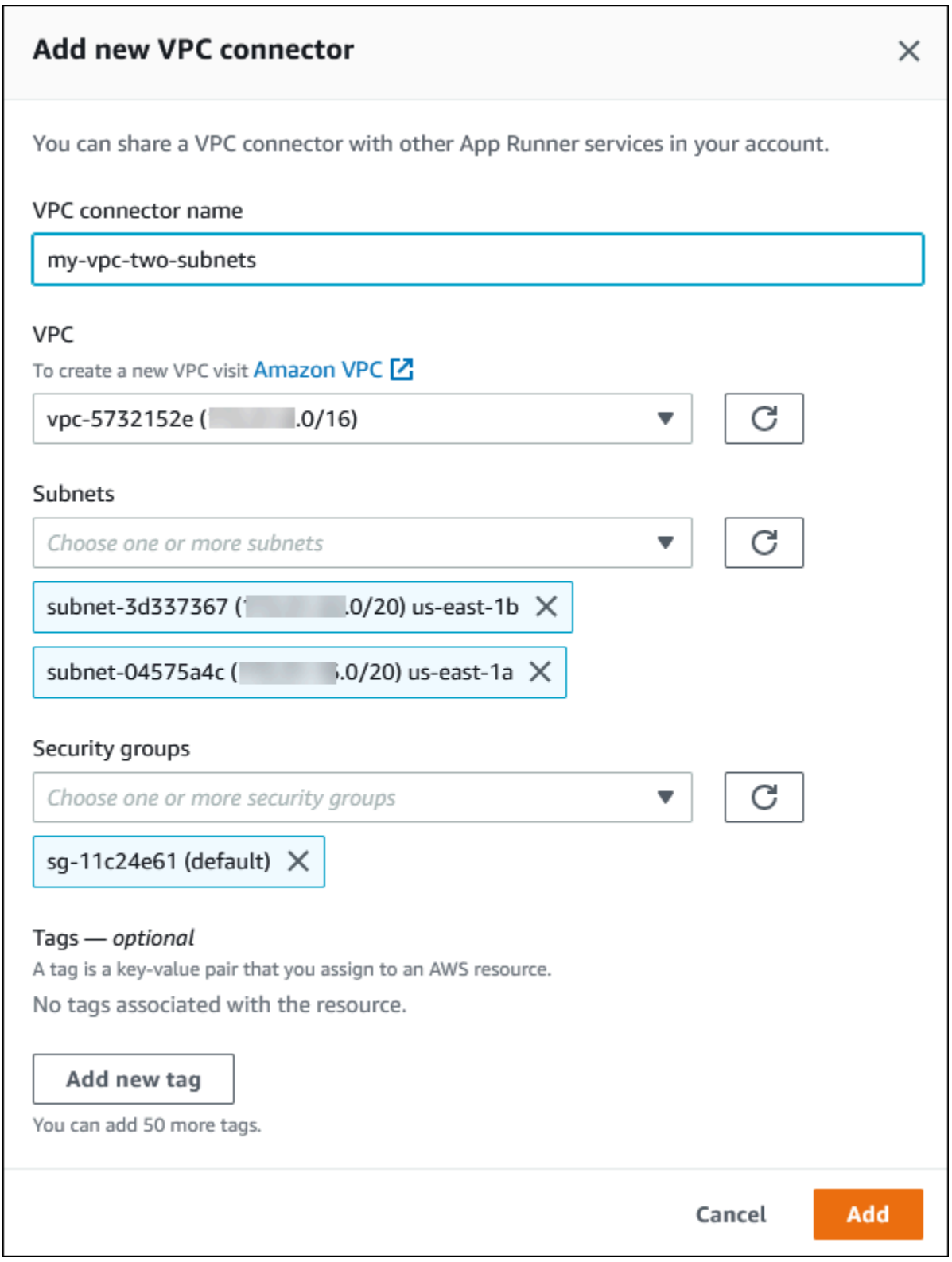
1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스 구성의 네트워킹 섹션으로 이동합니다.



3. 나가는 네트워크 트래픽에 대해 사용자 지정 VPC를 선택합니다.
4. 탐색 창에서 VPC 커넥터를 선택합니다.

   VPC 커넥터를 생성한 경우 콘솔에는 계정의 VPC 커넥터 목록이 표시됩니다. 기존 VPC 커넥터를 선택하고 다음을 선택하여 구성을 검토할 수 있습니다. 그런 다음, 마지막 단계로 넘어가세요. 또는 다음 단계를 사용하여 새 VPC 커넥터를 추가할 수 있습니다.
5. Add new를 선택하여 서비스를 위한 새 VPC 커넥터를 생성합니다.

   그러면 새 VPC 커넥터 추가 대화 상자가 열립니다.



6. VPC 커넥터의 이름을 입력하고 사용 가능한 목록에서 필요한 VPC를 선택합니다.

7. 서브넷의 경우 App Runner 서비스에 액세스하려는 각 가용 영역에 대해 하나의 서브넷을 선택합니다. 가용성을 높이려면 서브넷 3개를 선택하십시오. 또는 서브넷이 3개 미만인 경우 사용 가능한 모든 서브넷을 선택하십시오.

   > **Note**
   >
   > VPC 커넥터에 프라이빗 서브넷을 할당해야 합니다. VPC 커넥터에 퍼블릭 서브넷을 할당하는 경우 업데이트 중에 서비스가 생성되지 않거나 자동으로 롤백되지 않습니다.

8. (선택 사항) 보안 그룹에서 엔드포인트 네트워크 인터페이스와 연결할 보안 그룹을 선택합니다.
9. (선택 사항) 태그를 추가하려면 새 태그 추가를 선택하고 태그 키와 태그 값을 입력합니다.
10. 추가를 선택합니다.

    생성한 VPC 커넥터의 세부 정보가 VPC 커넥터 아래에 표시됩니다.

11. [다음] 을 선택하여 구성을 검토한 다음 [생성 및 배포] 를 선택합니다.

    App Runner는 사용자를 위해 VPC 커넥터 리소스를 만든 다음, 이를 서비스와 연결합니다. 서비스가 성공적으로 생성되면 콘솔에 새 서비스의 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.

App Runner API or AWS CLI

CreateService또는 UpdateServiceApp Runner API 작업을 호출할 때는 `NetworkConfiguration` 파라미터의 `EgressConfiguration` 멤버를 사용하여 서비스의 VPC 커넥터 리소스를 지정합니다.

다음 앱 러너 API 작업을 사용하여 VPC 커넥터 리소스를 관리할 수 있습니다.

- CreateVpcConnector— 새 VPC 커넥터를 생성합니다.
- ListVpcConnectors— 사용자와 연결된 VPC 커넥터 목록을 반환합니다. AWS 계정목록에는 전체 설명이 포함되어 있습니다.
- DescribeVpcConnector— VPC 커넥터의 전체 설명을 반환합니다.
- DeleteVpcConnector— VPC 커넥터를 삭제합니다. VPC 커넥터 할당량에 도달하면 불필요한 VPC 커넥터를 삭제해야 할 수 있습니다. AWS 계정

VPC에 대한 아웃바운드 액세스 권한이 있는 애플리케이션을 App Runner에 배포하려면 먼저 VPC 커넥터를 만들어야 합니다. 애플리케이션과 연결할 서브넷 및 보안 그룹을 하나 이상 지정하여 이 작업을 수행할 수 있습니다. 그런 다음 다음 예제와 같이 Create에서 또는 UpdateServiceCLI를 통해 VPC 커넥터를 참조할 수 있습니다.

```
            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
```

앱 러너 서비스의 옵저버빌리티

AWS App Runner 여러 서비스와 통합되어 App AWS Runner 서비스를 위한 광범위한 옵저버빌리티 도구 모음을 제공합니다. 이 장의 항목에서는 이러한 기능에 대해 설명합니다.

주제

- 앱 러너 서비스 활동 추적
- 로그로 스트리밍된 앱 러너 로그 보기 CloudWatch
- 보고된 App Runner 서비스 지표 보기 CloudWatch
- 에서 앱 러너 이벤트 처리 EventBridge
- 를 사용하여 앱 러너 API 호출 로깅 AWS CloudTrail
- X-Ray를 이용한 앱 러너 애플리케이션 추적

앱 러너 서비스 활동 추적

AWS App Runner 작업 목록을 사용하여 App Runner 서비스의 활동을 추적합니다. 작업은 서비스 생성, 구성 업데이트, 서비스 배포와 같은 API 작업에 대한 비동기 호출을 나타냅니다. 다음 섹션에서는 App Runner 콘솔에서 활동을 추적하고 API를 사용하는 방법을 보여줍니다.

앱 러너 서비스 활동을 추적하세요

다음 방법 중 하나를 사용하여 App Runner 서비스 활동을 추적하세요.

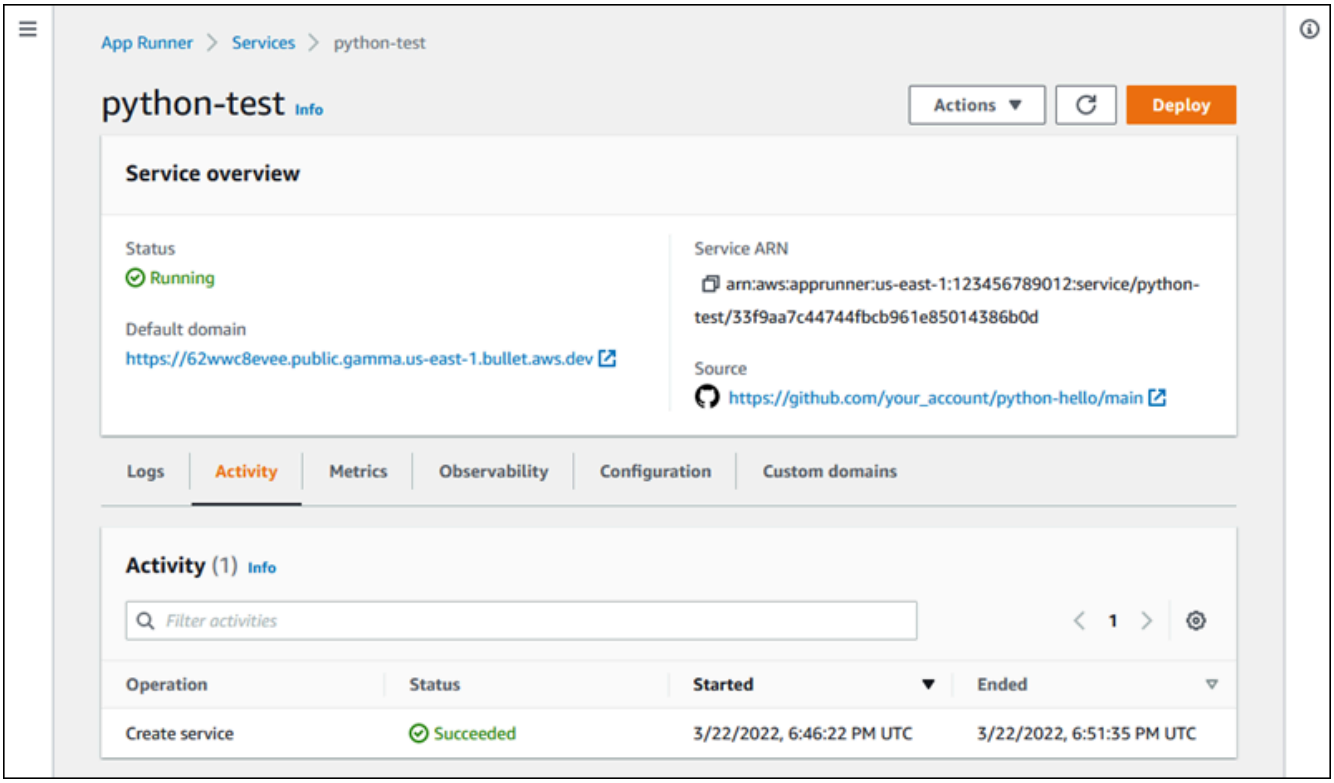
App Runner console

App Runner 콘솔은 App Runner 서비스 활동을 표시하고 운영을 탐색할 수 있는 다양한 방법을 제공합니다.

서비스 활동을 보려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 활동 탭을 선택합니다 (아직 선택하지 않은 경우).

   콘솔에는 작업 목록이 표시됩니다.

4. 특정 작업을 찾으려면 검색어를 입력하여 목록의 범위를 좁히십시오. 테이블에 나타나는 모든 값을 검색할 수 있습니다.

5. 나열된 작업을 선택하여 관련 로그를 보거나 다운로드할 수 있습니다.

App Runner API or AWS CLI

App Runner 서비스의 Amazon 리소스 이름 (ARN) 이 지정된 작업은 이 서비스에서 발생한 작업 목록을 반환합니다. ListOperations 각 목록 항목에는 작업 ID와 일부 추적 세부 정보가 포함되어 있습니다.

로그로 스트리밍된 앱 러너 로그 보기 CloudWatch

Amazon CloudWatch Logs를 사용하여 다양한 AWS 서비스의 리소스가 생성하는 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.

AWS App Runner 애플리케이션 배포 및 활성 서비스의 출력을 수집하여 Logs로 CloudWatch 스트리밍합니다. 다음 섹션에서는 App Runner 로그 스트림을 나열하고 App Runner 콘솔에서 이를 보는 방법을 보여줍니다.

앱 러너 로그 그룹 및 스트림

CloudWatch 로그는 로그 데이터를 로그 스트림에 보관하며, 로그 스트림은 로그 그룹으로 더 정리됩니다. 로그 스트림은 특정 소스에서 발생한 일련의 로그 이벤트입니다. 로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림 그룹입니다.

App Runner는 각 App Runner 서비스에 대해 각각 여러 로그 스트림을 포함하는 두 CloudWatch 개의 로그 로그 그룹을 정의합니다. AWS 계정

서비스 로그

서비스 로그 그룹에는 App Runner 서비스를 관리하고 이에 따라 작업을 수행할 때 App Runner에서 생성된 로깅 출력이 포함됩니다.

로그 그룹 이름	예
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/service`

서비스 로그 그룹 내에서 App Runner는 이벤트 로그 스트림을 생성하여 App Runner 서비스의 라이프사이클 내 활동을 캡처합니다. 예를 들어 이로 인해 애플리케이션이 시작되거나 일시 중지될 수 있습니다.

또한 App Runner는 서비스와 관련된 장기 실행 비동기 작업 각각에 대한 로그 스트림을 생성합니다. 로그 스트림 이름은 작업 유형과 특정 작업 ID를 반영합니다.

배포는 작업 유형입니다. 배포 로그에는 서비스를 만들거나 애플리케이션의 새 버전을 배포할 때 App Runner가 수행하는 빌드 및 배포 단계의 로깅 출력이 포함됩니다. 배포 로그 스트림 이름은 배포를 수행하는 작업의 ID로 `deployment/` 시작하고 끝납니다. 이 작업은 초기 애플리케이션 배포에 대한 CreateService호출이거나 각 추가 배포에 대한 StartDeployment호출입니다.

배포 로그 내에서 각 로그 메시지는 접두사로 시작됩니다.

- `[AppRunner]`— App Runner가 배포 중에 생성하는 출력입니다.

- [Build]— 자체 빌드 스크립트의 출력

로그 스트림 이름	예
events	N/A (고정 이름)
operation-type /*operation-id*	deployment/c2c8eeedea164f45 9cf78f12a8953390

애플리케이션 로그

애플리케이션 로그 그룹에는 실행 중인 애플리케이션 코드의 출력이 포함됩니다.

로그 그룹 이름	예
/aws/apprunner/ *service-n ame* /*service-id* /application	/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application

애플리케이션 로그 그룹 내에서 App Runner는 애플리케이션을 실행하는 각 인스턴스 (스케일링 유닛)에 대한 로그 스트림을 생성합니다.

로그 스트림 이름	예
instance/ *instance-id*	instance/1a80bc9134a84699b7 b3432ebeebb591

콘솔에서 앱 러너 로그 보기

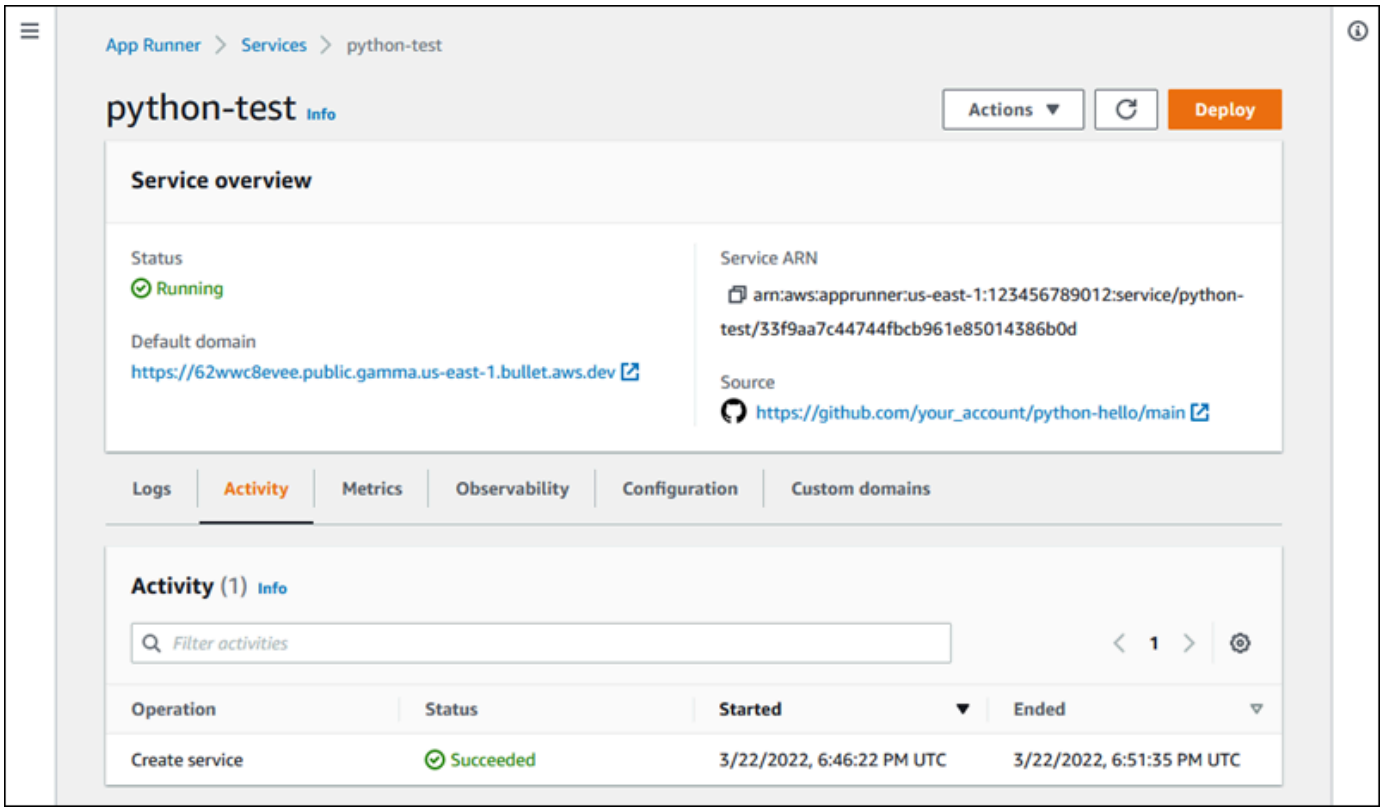
App Runner 콘솔에는 서비스에 대한 모든 로그가 요약되어 표시되며 이를 보고, 탐색하고, 다운로드할 수 있습니다.

서비스 로그를 보려면

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전

2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

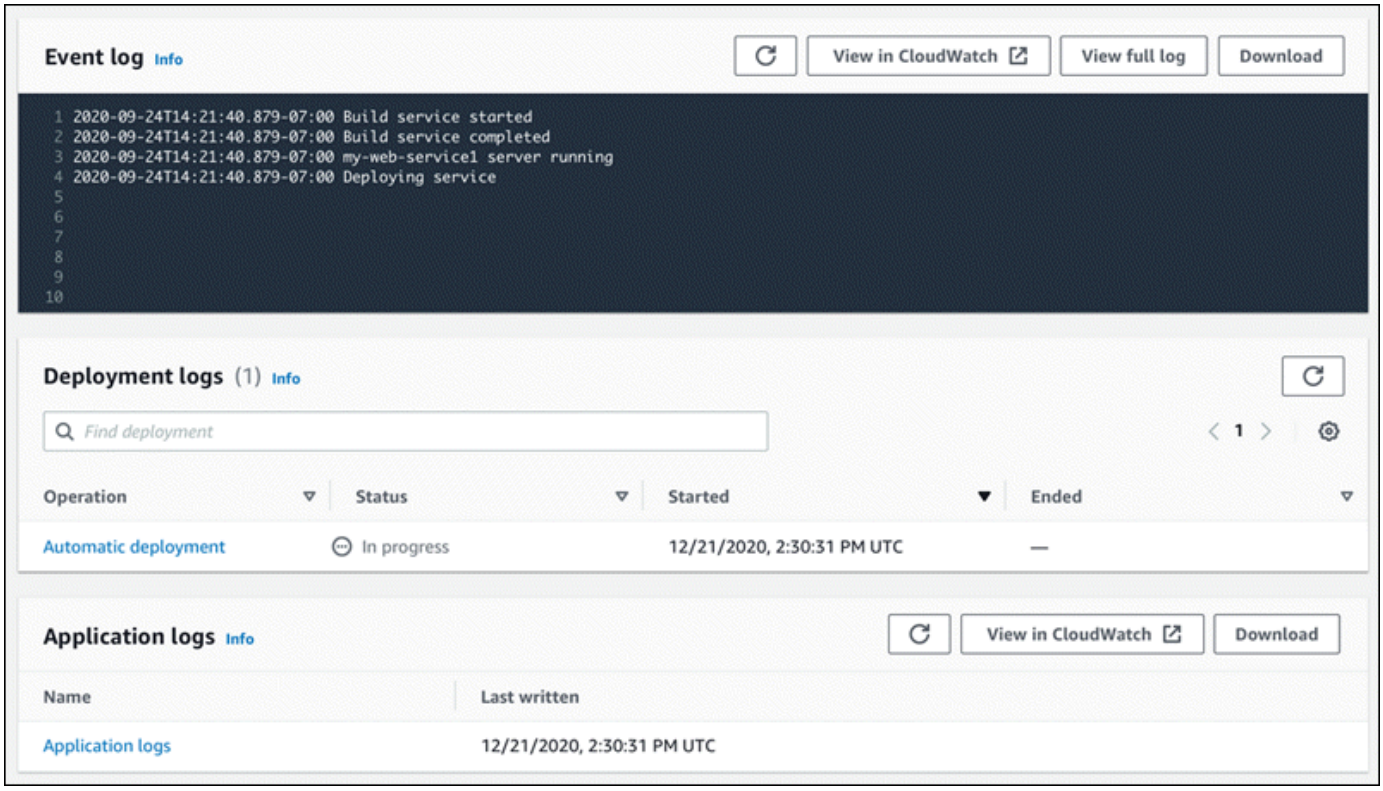
   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 로그 탭을 선택합니다.

   콘솔에는 여러 섹션에 몇 가지 유형의 로그가 표시됩니다.

   - 이벤트 로그 - App Runner 서비스의 라이프사이클 내 활동입니다. 콘솔에는 최신 이벤트가 표시됩니다.
   - 배포 로그 - App Runner 서비스에 대한 소스 리포지토리 배포입니다. 콘솔에는 각 배포에 대한 별도의 로그 스트림이 표시됩니다.
   - 애플리케이션 로그 - App Runner 서비스에 배포된 웹 애플리케이션의 출력입니다. 콘솔은 실행 중인 모든 인스턴스의 출력을 단일 로그 스트림으로 결합합니다.



4. 특정 배포를 찾으려면 검색어를 입력하여 배포 로그 목록의 범위를 좁히십시오. 표에 나타나는 모든 값을 검색할 수 있습니다.
5. 로그 내용을 보려면 전체 로그 보기 (이벤트 로그) 또는 로그 스트림 이름 (배포 및 애플리케이션 로그) 을 선택합니다.
6. [Download] 를 선택하여 로그를 다운로드합니다. 배포 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.
7. View CloudWatch in을 선택하여 CloudWatch 콘솔을 열고 콘솔의 모든 기능을 사용하여 App Runner 서비스 로그를 탐색하세요. 배포 로그 스트림의 경우 먼저 로그 스트림을 선택하십시오.

> **Note**
>
> CloudWatch 콘솔은 결합된 애플리케이션 로그 대신 특정 인스턴스의 애플리케이션 로그를 보려는 경우에 특히 유용합니다.

보고된 App Runner 서비스 지표 보기 CloudWatch

Amazon은 Amazon Web Services (AWS) 리소스와 실행 중인 애플리케이션을 AWS 실시간으로 CloudWatch 모니터링합니다. 리소스와 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적하는 데 사용할 CloudWatch 수 있습니다. 또한 이를 사용하여 지표를 감시하는 경보를 만들 수도 있습니다. 특정 임계값에 도달하면 알림을 CloudWatch 보내거나 모니터링된 리소스를 자동으로 변경합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

AWS App Runner App Runner 서비스의 사용, 성능 및 가용성에 대한 가시성을 높이는 다양한 지표를 수집합니다. 일부 지표는 웹 서비스를 실행하는 개별 인스턴스를 추적하는 반면, 다른 지표는 전체 서비스 수준에서 추적합니다. 다음 섹션에는 App Runner 측정항목이 나열되어 있으며 App Runner 콘솔에서 이를 보는 방법을 보여줍니다.

앱 러너 측정항목

App Runner는 서비스와 관련된 다음 지표를 수집하여 CloudWatch 네임스페이스에 게시합니다. `AWS/AppRunner`

> **Note**
>
> 2023년 8월 23일 이전에는 CPU 사용률 및 메모리 사용률 지표가 오늘 계산된 사용률 퍼센트 대신 vCPU 단위 및 메모리 사용률 메가바이트를 기준으로 했습니다. 이 날짜 이전에 App Runner에서 애플리케이션을 실행한 후 App Runner 또는 CloudWatch 콘솔에서 이 날짜의 측정치를 보기로 선택한 경우, 지표가 두 단위 모두에 표시되고 그 결과 일부 불규칙성이 나타납니다.

> **Important**
>
> 2023년 8월 23일 이전의 CPU 사용률 및 메모리 사용률 측정값을 기반으로 하는 모든 CloudWatch 경보를 업데이트해야 합니다. vCPU 또는 메가바이트가 아닌 사용률을 기준으로 트리거하도록 경보를 업데이트하십시오. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

인스턴스 수준 지표는 각 인스턴스 (규모 조정 단위) 에 대해 개별적으로 수집됩니다.

무엇을 측정했나요?	지표	설명
CPU utilization	`CPUUtilization`	서비스 구성에서 예약한 총 CPU 사용량 중 1분 동안의 평균 CPU 사용량 비율입니다.
Memory utilization	`MemoryUtilization`	서비스 구성에서 예약한 총 메모리 중 1분 동안의 평균 메모리 사용량 비율입니다.

서비스 수준 지표는 전체 서비스에 대해 수집됩니다.

무엇을 측정했나요?	지표	설명
CPU utilization	`CPUUtilization`	서비스 구성에서 예약한 총 CPU 사용량 중 1분 동안 모든 인스턴스에서 집계된 CPU 사용량의 비율입니다.
Memory utilization	`MemoryUtilization`	서비스 구성에서 예약한 총 메모리 중 1분 동안 모든 인스턴스에서 집계된 메모리 사용량의 비율입니다.
Concurrency	`Concurrency`	서비스에서 처리 중인 대략적인 동시 요청 수입니다.
HTTP request count	`Requests`	서비스가 수신한 HTTP 요청 수

HTTP status counts	`2xxStatusResponses`
	`4xxStatusResponses`
	`5xxStatusResponses`
		범주별로 그룹화된 각 응답 상태를 반환한 HTTP 요청 수 (2XX, 4XX, 5XX).

무엇을 측정했나요?	지표	설명
HTTP request latency	`RequestLatency`	웹 서비스가 HTTP 요청을 처리하는 데 걸린 시간(밀리초)
Instance counts	`ActiveInstances`	서비스에 대한 HTTP 요청을 처리 중인 인스턴스의 수. **Note** `ActiveInstances` 지표가 0으로 표시되면 서비스에 대한 요청이 없음을 의미합니다. 그렇다고 서비스 인스턴스 수가 0인 것은 아닙니다.

콘솔에서 App Runner 측정항목 보기

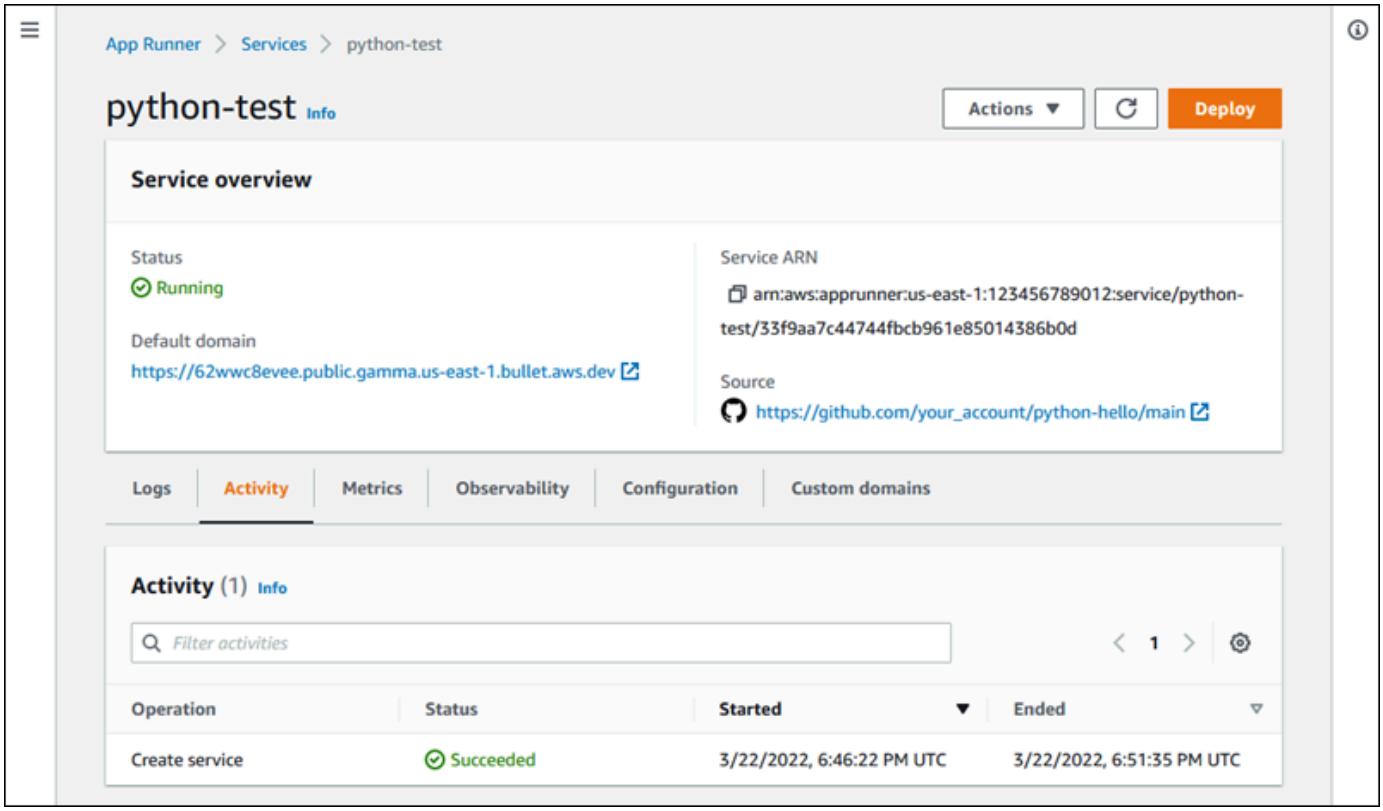
App Runner 콘솔은 App Runner가 서비스에 대해 수집하는 지표를 그래픽으로 표시하고 이를 탐색할 수 있는 다양한 방법을 제공합니다.

> **Note**
> 현재 콘솔에는 서비스 지표만 표시됩니다. 인스턴스 지표를 보려면 CloudWatch 콘솔을 사용하십시오.

서비스에 대한 로그를 보려면

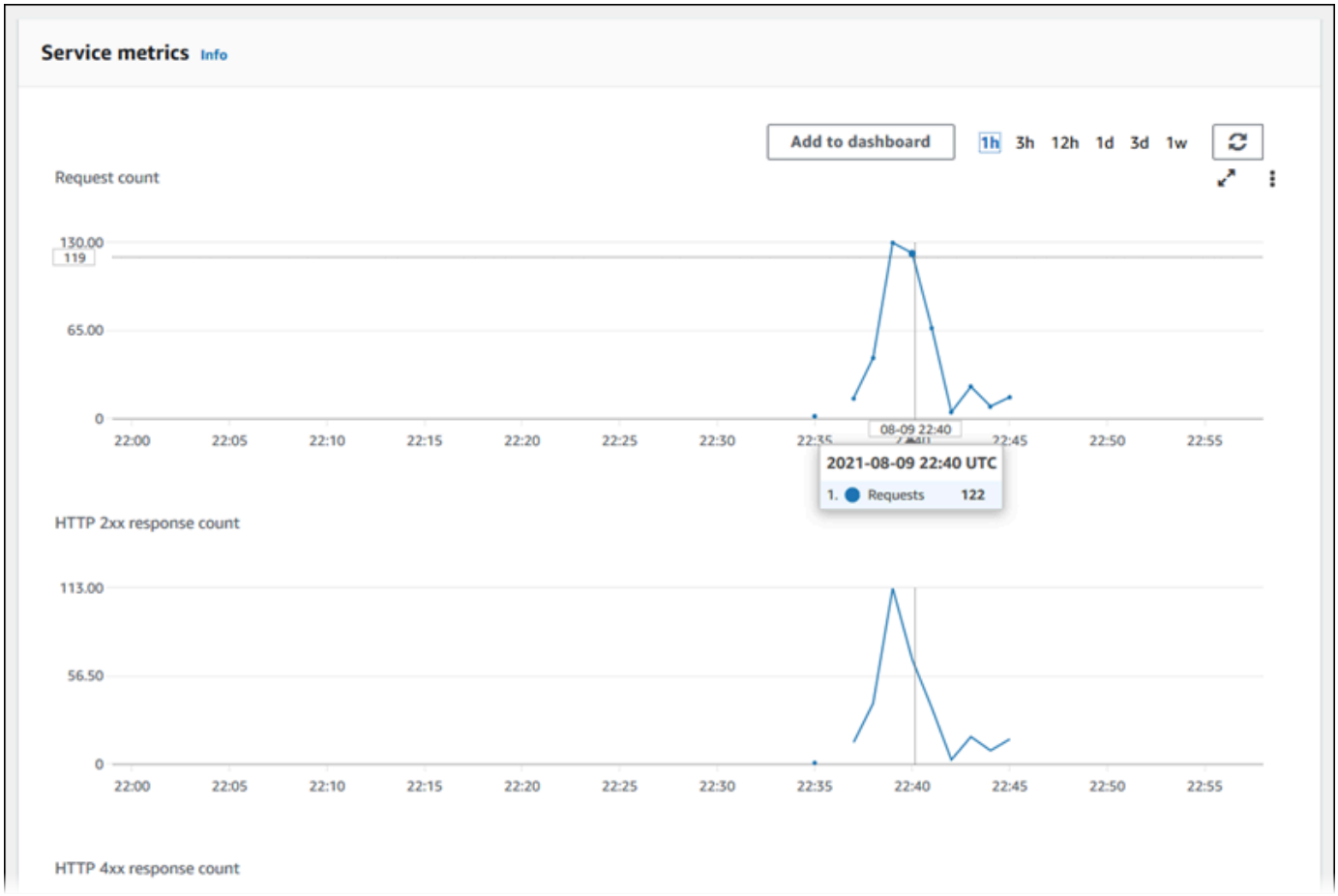
1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 탐색 창에서 서비스를 선택한 다음 App Runner 서비스를 선택합니다.

   콘솔에는 서비스 개요가 포함된 서비스 대시보드가 표시됩니다.



3. 서비스 대시보드 페이지에서 지표 탭을 선택합니다.

   콘솔에는 일련의 지표 그래프가 표시됩니다.



4. 기간 (예: 12시간) 을 선택하여 해당 기간의 최근 기간으로 지표 그래프의 범위를 지정합니다.
5. 그래프 섹션 중 하나의 상단에서 대시보드에 추가를 선택하거나, 모든 그래프의 메뉴를 사용하여 CloudWatch 콘솔의 대시보드에 관련 지표를 추가하여 추가 조사를 진행할 수 있습니다.

에서 앱 러너 이벤트 처리 EventBridge

EventBridgeAmazon을 사용하면 특정 패턴에 대해 AWS App Runner 서비스의 실시간 데이터 스트림을 모니터링하는 이벤트 기반 규칙을 설정할 수 있습니다. 규칙의 패턴이 일치하면 Amazon ECS 및 Amazon SNS와 같은 AWS Lambda대상에서 작업을 EventBridge 시작합니다. AWS Batch예를 들어, 서비스 배포가 실패할 때마다 Amazon SNS 주제에 신호를 보내 이메일 알림을 보내는 규칙을 설정할 수 있습니다. 또는 서비스 업데이트가 실패할 때마다 Lambda 함수를 설정하여 Slack 채널에 알릴 수 있습니다. 에 대한 EventBridge 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하십시오.

App Runner는 다음 이벤트 유형을 다음으로 전송합니다. EventBridge

- 서비스 상태 변경 — App Runner 서비스의 상태 변경입니다. 예를 들어, 서비스 상태가 로 변경되었습니다. `DELETE_FAILED`

- 서비스 작업 상태 변경 — App Runner 서비스의 긴 비동기 작업 상태 변경입니다. 예를 들어, 서비스 생성이 시작되었거나, 서비스 업데이트가 성공적으로 완료되었거나, 서비스 배포가 완료되어 오류가 발생했습니다.

App Runner 이벤트에 EventBridge 적용되는 규칙 생성

EventBridge 이벤트는 소스 AWS 서비스, 세부 정보 (이벤트) 유형과 같은 일부 표준 EventBridge 필드와 이벤트 세부 정보가 포함된 이벤트별 필드 세트를 정의하는 객체입니다. EventBridge 규칙을 만들려면 EventBridge 콘솔을 사용하여 이벤트 패턴 (추적해야 하는 이벤트) 을 정의하고 대상 조치 (경기 시 수행해야 하는 작업) 를 지정합니다. 이벤트 패턴은 일치하는 이벤트와 비슷합니다. 일치시킬 필드의 하위 집합을 지정하고 각 필드에 대해 가능한 값 목록을 지정합니다. 이 항목에서는 App Runner 이벤트 및 이벤트 패턴의 예를 제공합니다.

EventBridge 규칙 생성에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 AWS 서비스 규칙 생성을 참조하십시오.

> **Note**
>
> 일부 서비스는 에서 EventBridge사전 정의된 패턴을 지원합니다. 이를 통해 이벤트 패턴이 생성되는 방법이 간소화됩니다. 양식에서 필드 값을 선택하고 자동으로 패턴을 EventBridge 생성합니다. 현재 App Runner는 사전 정의된 패턴을 지원하지 않습니다. 패턴을 JSON 개체로 입력해야 합니다. 이 항목의 예제를 출발점으로 사용할 수 있습니다.

앱 러너 이벤트 예제

다음은 App Runner가 보내는 이벤트의 몇 가지 예입니다. EventBridge

- 서비스 상태 변경 이벤트. 구체적으로 말하자면, 에서 `OPERATION_IN_PROGRESS` `RUNNING` 상태로 변경된 서비스입니다.

```
{
  "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"
  }
}
```

- 작업 상태 변경 이벤트입니다. 구체적으로 말하자면, 성공적으로 완료된 `UpdateService` 작업입니다.

```
{
  "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"
  }
}
```

앱 러너 이벤트 패턴 예제

다음 예시는 EventBridge 규칙에서 하나 이상의 App Runner 이벤트를 매칭하는 데 사용할 수 있는 이벤트 패턴을 보여줍니다. 이벤트 패턴은 이벤트와 비슷합니다. 일치시키려는 필드만 포함하고 각 필드에 스칼라 대신 목록을 제공하십시오.

- 서비스가 더 이상 상태가 아닌 특정 계정의 서비스에 대한 모든 서비스 RUNNING 상태 변경 이벤트를 매칭하십시오.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- 작업이 실패한 특정 계정의 서비스에 대한 모든 운영 상태 변경 이벤트를 일치시킵니다.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```

앱 러너 이벤트 참조

서비스 상태 변경

서비스 상태 변경 이벤트가 로 `detail-type` 설정되었습니다`AppRunner Service Status Change`. 여기에는 다음과 같은 세부 정보 필드와 값이 있습니다.

```
"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"
```

작업 상태 변경

작업 상태 변경 이벤트가 로 `detail-type` 설정되었습니다`AppRunner Service Operation Status Change`. 여기에는 다음과 같은 세부 정보 필드와 값이 있습니다.

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

다음 표에는 가능한 모든 상태 코드 및 관련 메시지가 나열되어 있습니다.

상태 표시기	메시지
`CreateServiceStarted`	서비스 생성이 시작되었습니다.
`CreateServiceCompletedSuccessfully`	서비스 생성이 성공적으로 완료되었습니다.
`CreateServiceFailed`	서비스 생성에 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
`DeleteServiceStarted`	서비스 삭제가 시작되었습니다.

상태 표시기	메시지
DeleteServiceCompletedSuccessfully	서비스 삭제가 성공적으로 완료되었습니다.
DeleteServiceFailed	서비스 삭제에 실패했습니다.
UpdateServiceStarted	
UpdateServiceCompletedSuccessfully	서비스 업데이트가 성공적으로 완료되었습니다. 새 애플리케이션 및 구성이 배포되었습니다.
	서비스 업데이트가 성공적으로 완료되었습니다. 새 구성이 배포되었습니다.
UpdateServiceFailed	서비스 업데이트에 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
DeploymentStarted	배포가 시작되었습니다.
DeploymentCompletedSuccessfully	배포가 성공적으로 완료되었습니다.
DeploymentFailed	배포에 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
PauseServiceStarted	서비스 일시 중지가 시작되었습니다.
PauseServiceCompletedSuccessfully	서비스 일시 중지가 성공적으로 완료되었습니다.
PauseServiceFailed	서비스 일시 중지에 실패했습니다.
ResumeServiceStarted	서비스 재개가 시작되었습니다.
ResumeServiceCompletedSuccessfully	서비스 재개가 성공적으로 완료되었습니다.
ResumeServiceFailed	서비스 재개에 실패했습니다.

를 사용하여 앱 러너 API 호출 로깅 AWS CloudTrail

App Runner는 App Runner에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스인 서비스와 통합되어 있습니다. CloudTrail앱 러너에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처된 호출에는 App Runner 콘솔에서의 호출과 App Runner API 작업에 대한 코드 호출이 포함됩니다. 트레일을 생성하면 App Runner용 CloudTrail 이벤트를 포함하여 Amazon S3 버킷에 이벤트를 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 App Runner에 이루어진 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

자세히 CloudTrail 알아보려면 AWS CloudTrail 사용 설명서를 참조하십시오.

앱 러너 정보는 CloudTrail

CloudTrail 계정을 만들 AWS 계정 때 활성화됩니다. App Runner에서 활동이 발생하면 해당 활동이 CloudTrail 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. 에서 최근 이벤트를 보고, 검색하고, 다운로드할 수 있습니다. AWS 계정자세한 내용은 이벤트 기록으로 CloudTrail 이벤트 보기를 참조하십시오.

App Runner용 이벤트를 AWS 계정포함하여 내 이벤트의 진행 중인 기록을 보려면 트레일을 생성하세요. 트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 트레일은 AWS 파티션에 있는 모든 지역의 이벤트를 기록하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

모든 App Runner 작업은 AWS App Runner API 참조에 의해 CloudTrail 기록되고 문서화됩니다. 예를 들어, `CreateServiceDeleteConnection`, 및 `StartDeployment` 작업에 대한 호출은 CloudTrail 로그 파일에 항목을 생성합니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부.

자세한 내용은 CloudTrail 사용자 ID 요소를 참조하십시오.

앱 러너 로그 파일 항목 이해

트레일은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜 및 시간, 요청 매개 변수에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제는 `CreateService` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

> **Note**
>
> 보안상의 이유로 로그에서 일부 속성 값이 수정되고 텍스트로 대체됩니다. `HIDDEN_DUE_TO_SECURITY_REASONS` 이렇게 하면 의도하지 않은 비밀 정보 노출을 방지할 수 있습니다. 하지만 여전히 이러한 속성이 요청에서 전달되었거나 응답에서 반환된 것을 확인할 수 있습니다.

`CreateService`App Runner 작업에 대한 예제 CloudTrail 로그 항목


```
{
  "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",
```

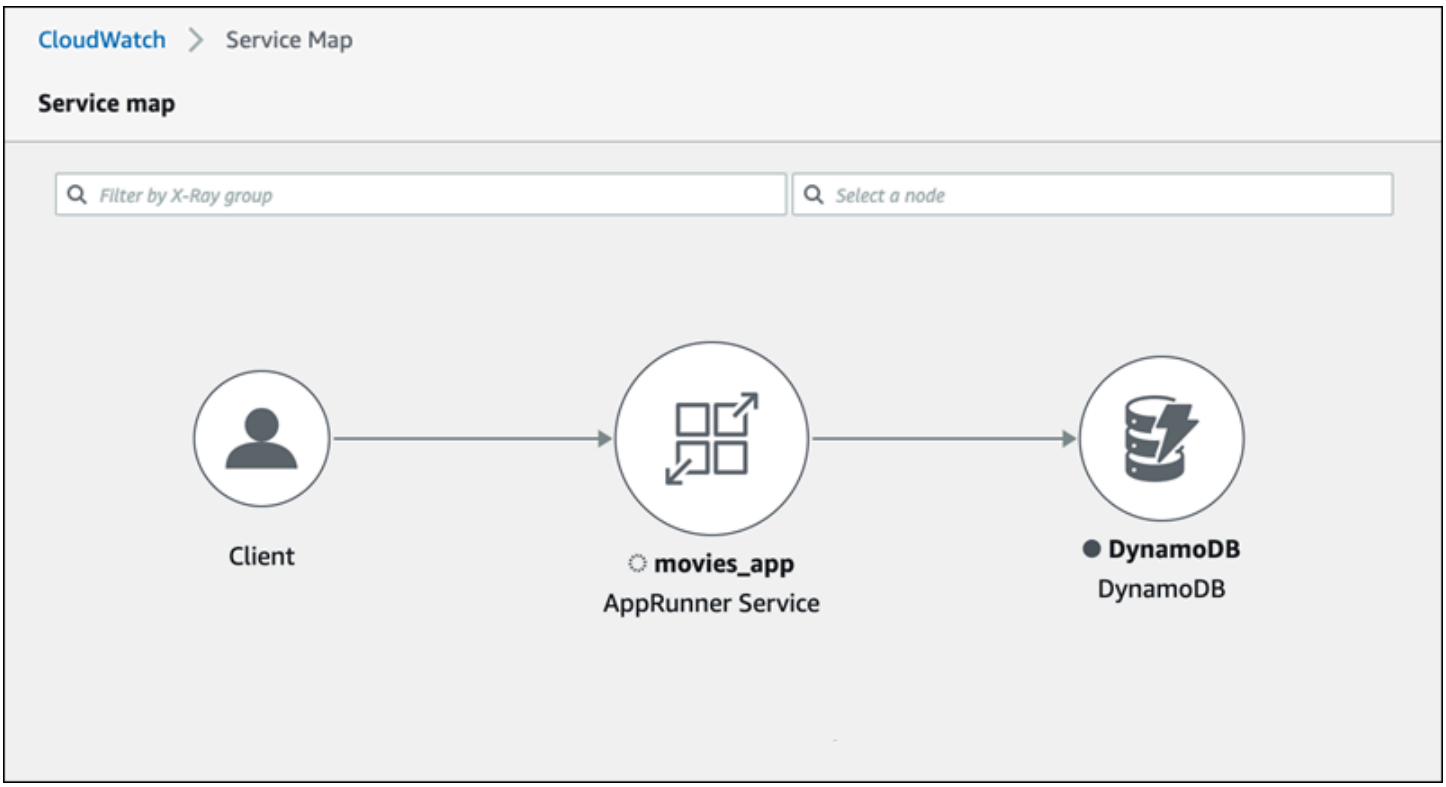
```
            "autoScalingConfigurationRevision": 1
        }
    }
},
  "requestID": "1a60af60-ecf5-4280-aa8f-64538319ba0a",
  "eventID": "e1a3f623-4d24-4390-a70b-bf08a0e24669",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```


X-Ray를 이용한 앱 러너 애플리케이션 추적

AWS X-Ray 애플리케이션에서 처리하는 요청에 대한 데이터를 수집하고, 해당 데이터를 보고, 필터링하고, 통찰력을 확보하여 최적화를 위한 문제와 기회를 식별하는 데 사용할 수 있는 도구를 제공하는 서비스입니다. 애플리케이션에 대한 추적된 요청의 경우 요청 및 응답뿐만 아니라 애플리케이션이 다운스트림 AWS 리소스, 마이크로서비스, 데이터베이스 및 HTTP 웹 API에 대해 수행하는 호출에 대한 세부 정보도 볼 수 있습니다.

X-Ray는 클라우드 애플리케이션을 지원하는 AWS 리소스의 추적 데이터를 사용하여 상세한 서비스 그래프를 생성합니다. 서비스 그래프는 프런트 엔드 서비스가 요청을 처리하고 데이터를 유지하기 위해 호출하는 클라이언트, 프런트 엔드 서비스 및 백엔드 서비스를 보여줍니다. 서비스 그래프를 사용하여 병목 현상, 지연 시간 급증 및 기타 해결해야 할 문제를 식별하여 애플리케이션의 성능을 개선하십시오.

X-Ray에 대한 자세한 내용은 AWS X-Ray 개발자 안내서를 참조하세요.



추적을 위해 애플리케이션을 계측하십시오.

휴대용 텔레메트리 사양인 을 사용하여 OpenTelemetry추적할 수 있도록 App Runner 서비스 애플리케이션을 계측하십시오. 현재 App Runner는 서비스를 사용하여 원격 측정 정보를 수집하고 표시하는 OpenTelemetry 구현인 AWS Distro for OpenTelemetry (ADOT) 를 지원합니다. AWS X-Ray는 추적 구성 요소를 구현합니다.

애플리케이션에서 사용하는 특정 ADOT SDK에 따라 ADOT는 자동 및 수동이라는 최대 두 가지 계측 접근 방식을 지원합니다. SDK를 사용한 계측에 대한 자세한 내용은 ADOT 설명서를 참조하고 탐색 창에서 SDK를 선택하십시오.

런타임 설정

다음은 App Runner 서비스 애플리케이션을 추적할 수 있도록 계측하기 위한 일반적인 런타임 설정 지침입니다.

런타임에 대한 추적 설정하기

1. AWS Distro for OpenTelemetry (ADOT) 의 런타임에 제공된 지침을 따라 애플리케이션을 계측하십시오.

2. 소스 코드 리포지토리를 사용하는 경우 `apprunner.yaml` 파일 `build` 섹션에, 컨테이너 이미지를 사용하는 경우 Dockerfile에 필요한 `OTEL` 종속성을 설치합니다.
3. 소스 코드 리포지토리를 사용하는 경우 `apprunner.yaml` 파일에 환경 변수를 설정하고 컨테이너 이미지를 사용하는 경우 Dockerfile에서 환경 변수를 설정합니다.

   Example 환경 변수

   > **Note**
   >
   > 다음 예제는 파일에 추가할 중요한 환경 변수를 나열합니다. `apprunner.yaml` 컨테이너 이미지를 사용하는 경우 이러한 환경 변수를 Dockerfile에 추가하세요. 그러나 각 런타임에는 고유한 특성이 있을 수 있으므로 다음 목록에 더 많은 환경 변수를 추가해야 할 수도 있습니다. 런타임별 지침 및 런타임에 맞게 애플리케이션을 설정하는 방법에 대한 예제에 대한 자세한 내용은 시작하기에서 런타임용AWS 배포판 OpenTelemetry 및 실행으로 이동을 참조하십시오.

```
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`App Runner Otel 컬렉터는 메트릭 로깅을 허용하지 않으므로 App Runner의 중요한 환경 변수입니다. 메트릭 추적만 허용합니다.

런타임 설정 예제

다음 예제는 ADOT Python SDK를 사용하여 애플리케이션을 자동 계측하는 방법을 보여줍니다. SDK는 Python 코드를 한 줄도 추가하지 않고도 애플리케이션의 Python 프레임워크에서 사용하는 값을 설

명하는 원격 측정 데이터가 포함된 범위를 자동으로 생성합니다. 두 소스 파일에 몇 줄만 추가하거나 수정하면 됩니다.

먼저 다음 예제와 같이 종속성을 몇 개 추가합니다.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

그런 다음 애플리케이션을 계측하십시오. 이를 수행하는 방법은 서비스 소스 (소스 이미지 또는 소스 코드) 에 따라 다릅니다.

Source image

서비스 소스가 이미지인 경우 컨테이너 이미지 빌드와 이미지에서 애플리케이션 실행을 제어하는 Dockerfile을 직접 계측할 수 있습니다. 다음 예제는 Python 애플리케이션을 위한 인스트루먼트된 Dockerfile을 보여줍니다. 계측 추가 사항은 굵게 강조 표시되어 있습니다.

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

서비스 소스가 애플리케이션 소스가 포함된 리포지토리인 경우 App Runner 구성 파일 설정을 사용하여 이미지를 간접적으로 계측합니다. 이러한 설정은 App Runner가 생성하여 애플리케이션용 이미지를 빌드하는 데 사용하는 Dockerfile을 제어합니다. 다음 예제는 Python 애플리케이션을 위한

인스트루먼트된 App Runner 구성 파일을 보여줍니다. 계측 추가 사항은 굵게 강조 표시되어 있습니다.

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'
```

App Runner 서비스 인스턴스 역할에 X-Ray 권한 추가

App Runner 서비스에서 X-Ray 추적을 사용하려면 서비스 인스턴스에 X-Ray 서비스와 상호작용할 수 있는 권한을 제공해야 합니다. 이를 위해서는 인스턴스 역할을 서비스에 연결하고 X-Ray 권한이 있는 관리형 정책을 추가하면 됩니다. App Runner 인스턴스 역할에 대한 자세한 내용은 을 참조하십시오. the section called “인스턴스 역할” 인스턴스 역할에 `AWSXRayDaemonWriteAccess` 관리형 정책을 추가하고 생성 과정에서 서비스에 할당하세요.

App Runner 서비스에 X-Ray 추적 기능을 활성화하세요

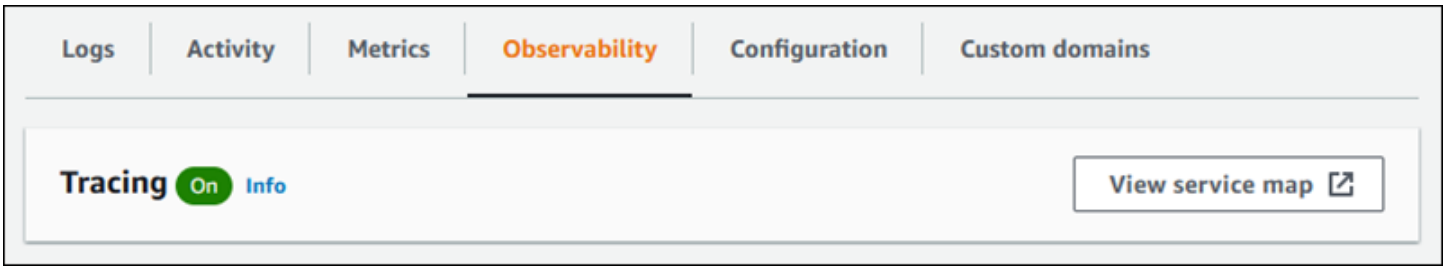
서비스를 만들면 App Runner는 기본적으로 추적을 비활성화합니다. 옵저버빌리티 구성의 일환으로 서비스에 대해 X-Ray Tracing을 활성화할 수 있습니다. 자세한 정보는 the section called “옵저버빌리티 관리”을 참조하세요.

App Runner API 또는 를 사용하는 경우 ObservabilityConfiguration리소스 TraceConfiguration개체 내의 개체에 추적 설정이 포함됩니다. AWS CLI추적을 계속 비활성화하려면 객체를 지정하지 마세요. `TraceConfiguration`

콘솔과 API의 경우 모두 이전 섹션에서 설명한 인스턴스 역할을 App Runner 서비스와 연결해야 합니다.

App Runner 서비스의 X-Ray 추적 데이터 보기

App Runner 콘솔의 서비스 대시보드 페이지에 있는 옵저버빌리티 탭에서 서비스 맵 보기를 선택하여 Amazon CloudWatch 콘솔로 이동합니다.



Amazon CloudWatch 콘솔을 사용하여 애플리케이션이 제공하는 요청에 대한 서비스 맵과 추적을 볼 수 있습니다. 서비스 맵에는 요청 지연 시간, 다른 애플리케이션 및 AWS 서비스와의 상호 작용과 같은 정보가 표시됩니다. 코드에 추가한 사용자 지정 주석을 사용하면 트레이스를 쉽게 검색할 수 있습니다. 자세한 내용은 Amazon 사용 CloudWatch 설명서의 애플리케이션 상태 ServiceLens 모니터링을 위한 사용을 참조하십시오.

AWS WAF 웹 ACL을 서비스와 연결

AWS WAF App Runner 서비스를 보호하는 데 사용할 수 있는 웹 애플리케이션 방화벽입니다. AWS WAF 웹 ACL (액세스 제어 목록) 을 사용하면 일반적인 웹 악용 및 원치 않는 봇으로부터 App Runner 서비스 엔드포인트를 보호할 수 있습니다.

웹 ACL을 사용하면 App Runner 서비스로 들어오는 모든 웹 요청을 세밀하게 제어할 수 있습니다. 웹 ACL에서 규칙을 정의하여 웹 트래픽을 허용, 차단 또는 모니터링하여 승인되고 합법적인 요청만 웹 애플리케이션 및 API에 도달하도록 할 수 있습니다. 특정 비즈니스 및 보안 요구 사항에 따라 웹 ACL 규칙을 사용자 지정할 수 있습니다. 인프라 보안 및 네트워크 ACL 적용 모범 사례에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 네트워크 트래픽 제어를 참조하십시오.

> **Important**
>
> WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다. 이는 현재 요청 소스 IP 데이터를 WAF와 관련된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다.

수신 웹 요청 흐름

AWS WAF 웹 ACL이 App Runner 서비스와 연결된 경우 들어오는 웹 요청은 다음 프로세스를 거칩니다.

1. App Runner는 원본 요청의 콘텐츠를 에 전달합니다. AWS WAF
2. AWS WAF 요청을 검사하고 해당 콘텐츠를 웹 ACL에서 지정한 규칙과 비교합니다.
3. 검사에 따라 App AWS WAF Runner에 `allow` or `block` 응답을 반환합니다.
   - `allow`응답이 반환되면 App Runner는 요청을 애플리케이션에 전달합니다.
   - `block`응답이 반환되면 App Runner는 요청이 웹 애플리케이션에 도달하지 못하도록 차단합니다. 의 `block` 응답을 AWS WAF 애플리케이션에 전달합니다.

> **Note**
>
> 기본적으로 App Runner는 응답이 반환되지 않으면 요청을 차단합니다. AWS WAF

AWS WAF 웹 ACL에 대한 자세한 내용은 개발자 안내서의 웹 액세스 제어 목록 (웹 ACL) 을 참조하십시오.AWS WAF

> **Note**
>
> 표준 AWS WAF 가격을 지불합니다. App Runner 서비스에 AWS WAF 웹 ACL을 사용하는 데에는 추가 비용이 발생하지 않습니다. 요금에 대한 자세한 내용은 요금을 참조하십시오.AWS WAF

WAF 웹 ACL을 앱 러너 서비스에 연결

다음은 AWS WAF 웹 ACL을 App Runner 서비스에 연결하는 고급 프로세스입니다.

1. 콘솔에서 웹 ACL을 생성합니다. AWS WAF 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 생성을 참조하십시오.
2. 에 대한 AWS Identity and Access Management (IAM) 권한을 업데이트하십시오. AWS WAF자세한 내용은 Permissions 단원을 참조하십시오.
3. 다음 방법 중 하나를 사용하여 웹 ACL을 App Runner 서비스에 연결합니다.
   - 앱 러너 콘솔: 앱 러너 서비스를 만들거나 업데이트할 때 앱 러너 콘솔을 사용하여 기존 웹 ACL을 연결합니다. 지침은 웹 ACL 관리를 참조하십시오. AWS WAF
   - AWS WAF 콘솔: 기존 App Runner 서비스의 AWS WAF 콘솔을 사용하여 웹 ACL을 연결합니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL과 AWS 리소스 연결 또는 연결 해제를 참조하십시오.
   - AWS CLI: 공개 API를 사용하여 웹 ACL을 AWS WAF 연결합니다. AWS WAF 퍼블릭 API에 대한 자세한 내용은 API 참조 안내서의 AssociateWebACL을AWS WAF 참조하십시오.

고려 사항

- WAF 웹 ACL과 연결된 앱 러너 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다. 이는 현재 요청 소스 IP 데이터를 WAF와 관련된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다.
- App Runner 서비스는 하나의 웹 ACL에만 연결할 수 있습니다. 하지만 하나의 웹 ACL을 여러 App Runner 서비스 및 여러 리소스와 연결할 수 있습니다. AWS Amazon Cognito 사용자 풀과 애플리케이션 로드 밸런서 리소스를 예로 들 수 있습니다.

- 웹 ACL을 생성하면 웹 ACL이 완전히 전파되어 App Runner에서 사용할 수 있게 되기까지 약간의 시간이 소요됩니다. 전파 시간은 몇 초에서 몇 분까지 걸릴 수 있습니다. AWS WAF 완전히 전파되기 전에 웹 ACL을 연결하려고 `WAFUnavailableEntityException` 하면 a를 반환합니다.

  웹 ACL이 완전히 전파되기 전에 브라우저를 새로 고치거나 App Runner 콘솔에서 다른 곳으로 이동하면 연결이 발생하지 않습니다. 하지만 App Runner 콘솔 내에서는 탐색할 수 있습니다.
- AWS WAF 유효하지 않은 상태인 App Runner 서비스에 대해 다음 AWS WAF API 중 하나를 호출하면 `WAFNonexistentItemException` 오류가 반환됩니다.
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  App Runner 서비스의 잘못된 상태는 다음과 같습니다.
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`
  - `OPERATION_IN_PROGRESS`

  > **Note**
  >
  > `OPERATION_IN_PROGRESS`상태는 App Runner 서비스가 삭제되는 경우에만 유효하지 않습니다.
- 요청으로 인해 페이로드가 검사할 수 있는 한도보다 클 AWS WAF 수 있습니다. App Runner의 크기 초과 요청을 AWS WAF 처리하는 방법에 대한 자세한 내용은 AWS WAF 개발자 안내서의 크기 초과 요청 구성 요소 처리를 참조하여 App Runner의 크기 초과 요청을 AWS WAF 처리하는 방법을 알아보세요.
- 적절한 규칙을 설정하지 않거나 트래픽 패턴이 변경되면 웹 ACL이 애플리케이션 보안에 그다지 효과적이지 않을 수 있습니다.

권한

에서 AWS App Runner웹 ACL을 사용하려면 다음 IAM 권한을 추가하십시오. AWS WAF

- `apprunner:ListAssociatedServicesForWebAcl`

- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

IAM 권한에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 정책 및 권한을 참조하십시오.

다음은 에 대한 업데이트된 IAM 정책의 예입니다. AWS WAF이 IAM 정책에는 App Runner 서비스를 사용하는 데 필요한 권한이 포함되어 있습니다.

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**
>
> IAM 권한을 부여해야 하지만 나열된 작업은 권한 전용이며 API 작업에 해당하지 않습니다.

AWS WAF 웹 ACL 관리

다음 방법 중 하나를 사용하여 App Runner 서비스의 AWS WAF 웹 ACL을 관리하세요.

- the section called “앱 러너 콘솔”
- the section called “AWS CLI”

앱 러너 콘솔

App Runner 콘솔에서 서비스를 만들거나 기존 서비스를 업데이트할 때 웹 ACL을 연결하거나 연결을 끊을 수 있습니다. AWS WAF

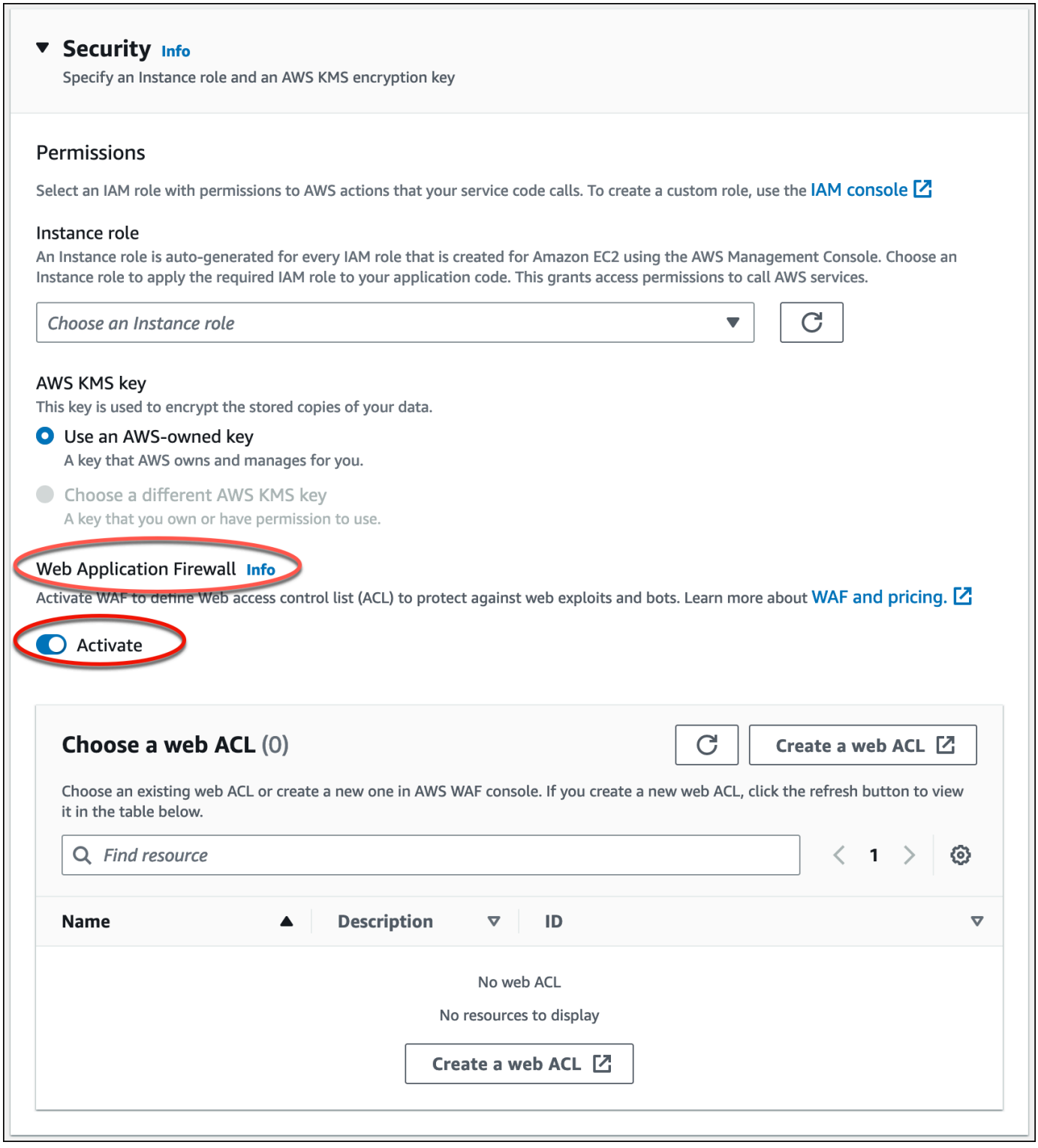
> **Note**
>
> - 앱 러너 서비스는 하나의 웹 ACL에만 연결할 수 있습니다. 하지만 하나의 웹 ACL을 다른 리소스 외에도 둘 이상의 App Runner 서비스와 연결할 수 있습니다. AWS
> - 웹 ACL을 연결하기 전에 에 대한 IAM 권한을 업데이트해야 합니다. AWS WAF자세한 내용은 Permissions 단원을 참조하십시오.

웹 ACL 연결 AWS WAF

> **Important**
>
> WAF 웹 ACL과 연결된 앱 러너 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다. 이는 현재 요청 소스 IP 데이터를 WAF와 관련된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다.

웹 ACL을 연결하려면 AWS WAF

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 서비스를 생성 중인지 업데이트 중인지에 따라 다음 단계 중 하나를 수행하십시오.
   - 새 서비스를 생성하는 경우 App Runner 서비스 생성을 선택하고 서비스 구성으로 이동합니다.
   - 기존 서비스를 업데이트하려면 구성 탭을 선택한 다음 서비스 구성에서 편집을 선택합니다.
3. 보안에서 웹 애플리케이션 방화벽으로 이동합니다.
4. 활성화 토글 버튼을 선택하여 옵션을 확인합니다.



5. 다음 단계 중 하나를 수행하세요.

   - 기존 웹 ACL을 연결하려면: 앱 러너 서비스에 연결할 웹 ACL 선택 테이블에서 필요한 웹 ACL을 선택합니다.

- 새 웹 ACL을 만들려면: 웹 ACL 생성을 선택하여 콘솔을 사용하여 새 웹 ACL을 생성합니다. AWS WAF 자세한 내용은 개발자 안내서의 웹 ACL 생성을 참조하십시오.AWS WAF

  1. 새로 고침 버튼을 선택하면 웹 ACL 선택 테이블에서 새로 만든 웹 ACL을 볼 수 있습니다.
  2. 필요한 웹 ACL을 선택합니다.

6. 새 서비스를 만들려면 다음을 선택하고, 기존 서비스를 업데이트하려면 변경 내용 저장을 선택합니다. 선택한 웹 ACL은 App Runner 서비스와 연결됩니다.
7. 웹 ACL 연결을 확인하려면 서비스의 구성 탭을 선택하고 서비스 구성으로 이동합니다. 보안에서 웹 응용 프로그램 방화벽으로 스크롤하여 서비스와 관련된 웹 ACL의 세부 정보를 확인합니다.

> **Note**
>
> 웹 ACL을 만들면 웹 ACL이 완전히 전파되어 App Runner에서 사용할 수 있게 되기까지 약간의 시간이 걸립니다. 전파 시간은 몇 초에서 몇 분까지 걸릴 수 있습니다. AWS WAF 완전히 전파되기 전에 웹 ACL을 연결하려고 `WAFUnavailableEntityException` 하면 a를 반환합니다.
> 웹 ACL이 완전히 전파되기 전에 브라우저를 새로 고치거나 App Runner 콘솔에서 다른 곳으로 이동하면 연결이 발생하지 않습니다. 하지만 App Runner 콘솔 내에서는 탐색할 수 있습니다.

웹 ACL 연결 끊기 AWS WAF

App Runner 서비스를 업데이트하여 더 이상 필요하지 않은 AWS WAF 웹 ACL의 연결을 끊을 수 있습니다.

웹 ACL의 연결을 끊으려면 AWS WAF

1. App Runner 콘솔을 열고 지역 목록에서 해당 콘솔을 선택합니다. AWS 리전
2. 업데이트하려는 서비스의 구성 탭으로 이동한 다음 서비스 구성에서 편집을 선택합니다.
3. 보안에서 웹 애플리케이션 방화벽으로 이동합니다.
4. 활성화 토글 버튼을 비활성화합니다. 삭제를 확인하는 메시지가 나타납니다.
5. 확인을 선택합니다. 웹 ACL이 앱 러너 서비스와 연결이 끊겼습니다.

> **Note**
>
> - 서비스를 다른 웹 ACL과 연결하려면 웹 ACL 선택 테이블에서 웹 ACL을 선택합니다. App Runner는 현재 웹 ACL의 연결을 해제하고 선택한 웹 ACL과 연결하는 프로세스를 시작합니다.
> - 다른 App Runner 서비스나 리소스에서 연결이 끊긴 웹 ACL을 사용하지 않는 경우 웹 ACL을 삭제해 보세요. 그렇지 않으면 비용이 계속 발생합니다. 요금에 대한 자세한 내용은 AWS WAF 요금 부분을 참조하세요. 웹 ACL을 삭제하는 방법에 대한 지침은 API 참조의 DeleteWeb AWS WAF ACL을 참조하십시오.
> - 다른 활성 App Runner 서비스 또는 기타 리소스와 연결된 웹 ACL은 삭제할 수 없습니다.

AWS CLI

퍼블릭 API를 사용하여 AWS WAF 웹 ACL을 연결하거나 연결을 끊을 수 있습니다. AWS WAF 웹 ACL을 연결하거나 연결을 끊으려는 App Runner 서비스는 유효한 상태여야 합니다.

AWS WAF 잘못된 상태인 App Runner 서비스에 대해 다음 AWS WAF API 중 하나를 호출하면 `WAFNonexistentItemException` 오류가 반환됩니다.

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

App Runner 서비스의 잘못된 상태는 다음과 같습니다.

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`상태는 App Runner 서비스가 삭제되는 경우에만 유효하지 않습니다.

AWS WAF 퍼블릭 API에 대한 자세한 내용은 AWS WAF API 참조 가이드를 참조하십시오.

> **Note**
>
> 에 대한 IAM 권한을 업데이트하십시오. AWS WAF자세한 내용은 Permissions 단원을 참조하십시오.

를 AWS WAF 사용하여 웹 ACL 연결 AWS CLI

> **Important**
>
> WAF 웹 ACL과 연결된 앱 러너 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다. 이는 현재 요청 소스 IP 데이터를 WAF와 관련된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다.

웹 ACL을 연결하려면 AWS WAF

1. 서비스에 대해 선호하는 규칙 작업 세트 `Allow` 또는 `Block` 서비스에 대한 AWS WAF 웹 요청을 사용하여 서비스에 대한 웹 ACL을 생성하세요. AWS WAF API에 대한 자세한 내용은 AWS WAF API 참조 안내서의 CreateWebACL을 참조하십시오.

   Example 웹 ACL 생성 - 요청

   ```
   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. 퍼블릭 API를 사용하여 생성한 웹 ACL을 앱 러너 서비스에 연결합니다. `associate-web-acl` AWS WAF AWS WAF API에 대한 자세한 내용은 API 참조 가이드의 AssociateWeb AWS WAF ACL을 참조하십시오.

> **Note**
>
> 웹 ACL을 만들면 웹 ACL이 완전히 전파되어 App Runner에서 사용할 수 있게 되기까지 약간의 시간이 걸립니다. 전파 시간은 몇 초에서 몇 분까지 걸릴 수 있습니다. AWS WAF 완전히 전파되기 전에 웹 ACL을 연결하려고 `WAFUnavailableEntityException` 하면 a를 반환합니다.
> 웹 ACL이 완전히 전파되기 전에 브라우저를 새로 고치거나 App Runner 콘솔에서 다른 곳으로 이동하면 연결이 발생하지 않습니다. 하지만 App Runner 콘솔 내에서는 탐색할 수 있습니다.

Example 웹 ACL 연결 - 요청

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. 퍼블릭 API를 사용하여 웹 ACL이 앱 러너 서비스와 연결되어 있는지 확인하세요. `get-web-acl-for-resource` AWS WAF AWS WAF API에 대한 자세한 내용은 API 참조 ForResource 가이드의 GetWeb AWS WAF ACL을 참조하십시오.

Example 리소스용 웹 ACL 확인 - 요청

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

서비스와 연결된 웹 ACL이 없는 경우 빈 응답을 받게 됩니다.

를 사용하여 AWS WAF 웹 ACL 삭제 AWS CLI

앱 러너 서비스와 연결된 AWS WAF 웹 ACL은 삭제할 수 없습니다.

웹 ACL을 AWS WAF 삭제하려면

1. 퍼블릭 API를 사용하여 앱 러너 서비스에서 웹 ACL을 분리하십시오. `disassociate-web-acl` AWS WAF AWS WAF API에 대한 자세한 내용은 API 참조 가이드의 DisassociateWebACL을 참조하십시오.AWS WAF

   Example 웹 ACL 연결 해제 - 요청

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. 퍼블릭 API를 사용하여 앱 러너 서비스에서 웹 ACL의 연결이 끊겼는지 확인하세요. `get-web-acl-for-resource` AWS WAF

   Example 웹 ACL의 연결이 끊겼는지 확인 - 요청

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   연결이 해제된 웹 ACL은 App Runner 서비스에 나열되지 않습니다. 서비스와 연결된 웹 ACL이 없는 경우 빈 응답을 받게 됩니다.

3. 공개 API를 사용하여 연결이 끊긴 웹 ACL을 `delete-web-acl` AWS WAF 삭제합니다. AWS WAF API에 대한 자세한 내용은 API 참조 안내서의 DeleteWebACL을AWS WAF 참조하십시오.

   Example 웹 ACL 삭제 - 요청

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   --id <web_acl_id>
   --lock-token <web_acl_lock_token>
   --region <region>
   ```

4. `list-web-acl` AWS WAF 퍼블릭 API를 사용하여 웹 ACL이 삭제되었는지 확인합니다. AWS WAF API에 대한 자세한 내용은 AWS WAF API 참조 안내서의 ListWebACL을 참조하십시오.

Example 웹 ACL이 삭제되었는지 확인 - 요청

```
aws wafv2 list-web-acls
--scope REGIONAL
--region <region>
```

삭제된 웹 ACL은 더 이상 목록에 표시되지 않습니다.

> **Note**
>
> 웹 ACL이 다른 활성 App Runner 서비스 또는 Amazon Cognito 사용자 풀과 같은 다른 리소스와 연결되어 있는 경우 웹 ACL을 삭제할 수 없습니다.

웹 ACL과 연결된 앱 러너 서비스 목록

웹 ACL을 여러 App Runner 서비스 및 기타 리소스와 연결할 수 있습니다. 공개 API를 사용하여 웹 ACL과 연결된 앱 러너 서비스를 나열합니다. `list-resources-for-web-acl` AWS WAF AWS WAF API에 대한 자세한 내용은 API 참조 안내서의 ListResourcesForWebACL을AWS WAF 참조하십시오.

Example 웹 ACL과 관련된 앱 러너 서비스 목록 - 요청

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example 웹 ACL과 관련된 앱 러너 서비스 목록 - 응답

다음 예는 웹 ACL과 연결된 App Runner 서비스가 없을 때의 응답을 보여줍니다.

```
{
  "ResourceArns": []
}
```

Example 웹 ACL과 연결된 앱 러너 서비스 목록 - 응답

다음 예는 웹 ACL과 연결된 App Runner 서비스가 있을 때의 응답을 보여줍니다.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

웹 ACL 테스트 및 로깅 AWS WAF

웹 ACL에서 규칙 작업을 카운트로 설정하면 규칙과 일치하는 요청 수에 요청이 AWS WAF 추가됩니다. App Runner 서비스로 웹 ACL을 테스트하려면 규칙 동작을 Count로 설정하고 각 규칙과 일치하는 요청의 양을 고려하세요. 예를 들어 정상 사용자 트래픽이라고 판단되는 많은 요청과 일치하는 `Block` 작업에 대한 규칙을 설정합니다. 이 경우 규칙을 다시 구성해야 할 수 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 AWS WAF 보호 기능 테스트 및 조정을 참조하십시오.

Amazon CloudWatch Logs 로그 그룹, Amazon Simple Storage Service (Amazon S3) 버킷 또는 Amazon 데이터 파이어호스에 요청 헤더를 AWS WAF 기록하도록 구성할 수도 있습니다. 자세한 내용은 AWS WAF 개발자 안내서의 웹 ACL 트래픽 로깅을 참조하세요.

App Runner 서비스와 관련된 웹 ACL과 관련된 로그에 액세스하려면 다음 로그 필드를 참조하십시오.

- `httpSourceName`: 포함 `APPRUNNER`
- `httpSourceId`: 포함 `customeraccountid-apprunnerserviceid`

자세한 내용은 AWS WAF 개발자 안내서의 로그 예제를 참조하십시오.

> **Important**
>
> WAF 웹 ACL과 연결된 App Runner 프라이빗 서비스의 소스 IP 규칙은 IP 기반 규칙을 준수하지 않습니다. 이는 현재 요청 소스 IP 데이터를 WAF와 관련된 App Runner 프라이빗 서비스에 전달하는 것을 지원하지 않기 때문입니다. App Runner 애플리케이션에 소스 IP/CIDR 수신 트래픽 제어 규칙이 필요한 경우 WAF 웹 ACL 대신 프라이빗 엔드포인트에 대한 보안 그룹 규칙을 사용해야 합니다.

구성 파일을 사용하여 App Runner 서비스 옵션 설정

> **Note**
>
> 구성 파일은 소스 코드를 기반으로 하는 서비스에만 적용됩니다. 이미지 기반 서비스에는 구성 파일을 사용할 수 없습니다.

소스 코드 리포지토리를 사용하여 AWS App Runner 서비스를 생성하는 경우 서비스 빌드 및 시작에 대한 정보가 AWS App Runner 필요합니다. App Runner 콘솔 또는 API를 사용하여 서비스를 만들 때마다 이 정보를 제공할 수 있습니다. 또는 구성 파일을 사용하여 서비스 옵션을 설정할 수 있습니다. 파일에 지정하는 옵션은 소스 리포지토리의 일부가 되며, 이러한 옵션에 대한 모든 변경 사항은 소스 코드의 변경 내용을 추적하는 방식과 유사하게 추적됩니다. App Runner 구성 파일을 사용하여 API가 지원하는 것보다 더 많은 옵션을 지정할 수 있습니다. API가 지원하는 기본 옵션만 필요한 경우 구성 파일을 제공하지 않아도 됩니다.

App Runner 구성 파일은 애플리케이션 저장소의 소스 `apprunner.yaml` 디렉터리에 이름이 지정된 YAML 파일입니다. 서비스를 위한 빌드 및 런타임 옵션을 제공합니다. 이 파일의 값은 App Runner에 서비스를 빌드하고 시작하는 방법을 지시하고 네트워크 설정 및 환경 변수와 같은 런타임 컨텍스트를 제공합니다.

App Runner 구성 파일에는 CPU 및 메모리와 같은 운영 설정이 포함되어 있지 않습니다.

App Runner 구성 파일의 예는 을 참조하십시오. the section called “예제” 전체 참조 가이드는 을 참조하십시오the section called “레퍼런스”.

주제

- 앱 러너 구성 파일 예제
- 앱 러너 구성 파일 참조

앱 러너 구성 파일 예제

> **Note**
>
> 구성 파일은 소스 코드를 기반으로 하는 서비스에만 적용됩니다. 이미지 기반 서비스에는 구성 파일을 사용할 수 없습니다.

다음 예제는 AWS App Runner 구성 파일을 보여줍니다. 일부는 최소한이며 필수 설정만 포함합니다. 모든 구성 파일 섹션을 포함한 다른 섹션은 완성되었습니다. App Runner 구성 파일에 대한 개요는 을 참조하십시오앱 러너 구성 파일.

구성 파일 예제

최소 구성 파일

App Runner는 최소 구성 파일을 사용하여 다음과 같은 가정을 합니다.

- 빌드 또는 실행 중에는 사용자 지정 환경 변수가 필요하지 않습니다.
- 최신 런타임 버전이 사용됩니다.
- 기본 포트 번호와 포트 환경 변수가 사용됩니다.

Example apprunner.yaml

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

전체 구성 파일

이 예제는 관리되는 런타임에서 `apprunner.yaml` 원래 형식의 모든 구성 키를 사용하는 방법을 보여줍니다.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-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"
```

전체 구성 파일 — (수정된 빌드 사용)

이 예제에서는 관리형 `apprunner.yaml` 런타임과 함께 에서 모든 구성 키를 사용하는 방법을 보여줍니다.

이 `pre-run` 매개변수는 수정된 App Runner 빌드에서만 지원됩니다. 애플리케이션이 원본 App Runner 빌드에서 지원하는 런타임 버전을 사용하는 경우 구성 파일에 이 매개변수를 삽입하지 마세요. 자세한 정보는 관리형 런타임 버전 및 앱 러너 빌드을 참조하세요.

> **Note**
>
> 이 예제는 Python 3.11용이므로 `pip3` 및 `python3` 명령을 사용합니다. 자세한 내용은 Python 플랫폼 주제를 참조하십시오특정 런타임 버전에 대한 콜아웃.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-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"
```

특정 관리형 런타임 구성 파일의 예는 의 특정 런타임 하위 항목을 참조하십시오. 코드 기반 서비스

앱 러너 구성 파일 참조

> **Note**
>
> 구성 파일은 소스 코드를 기반으로 하는 서비스에만 적용됩니다. 이미지 기반 서비스에는 구성 파일을 사용할 수 없습니다.

이 항목은 구성 파일의 구문 및 의미에 대한 포괄적인 참조 안내서입니다. AWS App Runner App Runner 구성 파일에 대한 개요는 를 참조하십시오. 앱 러너 구성 파일

앱 러너 구성 파일은 YAML 파일입니다. 이름을 지정하고 애플리케이션 저장소의 소스 디렉터리에 배치합니다. `apprunner.yaml`

구조 개요

앱 러너 구성 파일은 YAML 파일입니다. 이름을 지정하고 애플리케이션 저장소의 소스 디렉터리에 배치합니다. `apprunner.yaml`

App Runner 구성 파일에는 다음과 같은 주요 부분이 포함되어 있습니다.

- 상위 섹션 — 최상위 키 포함
- 빌드 섹션 - 빌드 단계를 구성합니다.
- 실행 섹션 - 런타임 단계를 구성합니다.

상단 섹션

파일 상단의 키는 파일 및 서비스 런타임에 대한 일반 정보를 제공합니다. 다음과 같은 키를 사용할 수 있습니다.

- `version`— 필수. 앱 러너 구성 파일 버전입니다. 가장 좋은 방법은 최신 버전을 사용하는 것입니다.

  구문

```
version: version
```

Example

```
version: 1.0
```

- `runtime`— 필수. 애플리케이션에서 사용하는 런타임의 이름. App Runner에서 제공하는 다양한 프로그래밍 플랫폼에서 사용 가능한 런타임에 대해 알아보려면 을 참조하십시오. 코드 기반 서비스

> **Note**
>
> 관리형 런타임의 명명 규칙은 다음과 같습니다. <language-name><major-version>

구문

```
runtime: runtime-name
```

Example

```
runtime: python3
```

빌드 섹션

빌드 섹션에서는 App Runner 서비스 배포의 빌드 단계를 구성합니다. 빌드 명령과 환경 변수를 지정할 수 있습니다. 빌드 명령이 필요합니다.

섹션은 `build:` 키로 시작하며 다음과 같은 하위 키로 구성됩니다.

- `commands`— 필수. App Runner가 다양한 빌드 단계에서 실행하는 명령을 지정합니다. 다음과 같은 하위 키가 포함됩니다.
  - `pre-build`— 선택 사항. App Runner가 빌드 전에 실행하는 명령 예를 들어 npm 종속 항목 또는 테스트 라이브러리를 설치할 수 있습니다.
  - `build`— 필수. App Runner가 애플리케이션을 빌드하기 위해 실행하는 명령입니다. 예를 들어, 를 사용하세요pipenv.
  - `post-build`— 선택 사항. 앱 러너가 빌드 후에 실행하는 명령어 예를 들어 Maven을 사용하여 빌드 아티팩트를 JAR 또는 WAR 파일로 패키징하거나 테스트를 실행할 수 있습니다.

구문

```
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— 선택 사항입니다. 빌드 단계의 사용자 지정 환경 변수를 지정합니다. 이름-값 스칼라 매핑으로 정의됩니다. 빌드 명령에서 이러한 변수를 이름으로 참조할 수 있습니다.

> **Note**
>
> 이 구성 파일에는 서로 다른 두 위치에 서로 다른 두 개의 env 항목이 있습니다. 한 세트는 Build 섹션에 있고 다른 세트는 Run 섹션에 있습니다.
>
> - 빌드 프로세스 중에pre-build, buildpost-build, 및 pre-run 명령으로 빌드 섹션의 env 세트를 참조할 수 있습니다.
>
> 중요 - pre-run 명령어는 빌드 섹션에 정의된 환경 변수에만 액세스할 수 있지만 이 파일의 Run 섹션에 있다는 점에 유의하십시오.
>
> - 실행 섹션의 env 세트는 런타임 환경의 run 명령으로 참조할 수 있습니다.

구문

```
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"
```

실행 섹션

실행 섹션은 App Runner 애플리케이션 배포의 컨테이너 실행 단계를 구성합니다. 런타임 버전, 사전 실행 명령 (수정된 형식만 해당), 시작 명령, 네트워크 포트 및 환경 변수를 지정할 수 있습니다.

섹션은 `run:` 키로 시작하며 다음과 같은 하위 키가 있습니다.

- `runtime-version`— 선택 사항. App Runner 서비스에 대해 잠그려는 런타임 버전을 지정합니다.

  기본적으로 메이저 버전만 잠깁니다. App Runner는 모든 배포 또는 서비스 업데이트에서 런타임에 사용할 수 있는 최신 마이너 및 패치 버전을 사용합니다. 메이저 버전과 마이너 버전을 지정하면 둘 다 잠기고 App Runner는 패치 버전만 업데이트합니다. 메이저, 마이너, 패치 버전을 지정하면 서비스가 특정 런타임 버전에서만 잠기며 App Runner는 해당 버전을 업데이트하지 않습니다.

  구문

```
run:
  runtime-version: major[.minor[.patch]]
```

> **Note**
>
> 일부 플랫폼의 런타임은 버전 구성요소가 다릅니다. 자세한 내용은 특정 플랫폼 주제를 참조하십시오.

Example

```
runtime: python3
run:
  runtime-version: 3.7
```

- `pre-run`— 선택 사항. 빌드 사용만 수정되었습니다. 빌드 이미지에서 실행 이미지로 애플리케이션을 복사한 후 App Runner가 실행하는 명령을 지정합니다. 여기에 명령을 입력하여 `/app` 디렉터리 외부에서 실행 이미지를 수정할 수 있습니다. 예를 들어 `/app` 디렉터리 외부에 있는 추가 글로벌 종속성을 설치해야 하는 경우 이 하위 섹션에 필요한 명령을 입력하여 설치하십시오. App Runner 빌드 프로세스에 대한 자세한 내용은 을 참조하십시오. 관리형 런타임 버전 및 앱 러너 빌드

> **Note**
>
> - 중요 — `pre-run` 명령이 Run 섹션에 나열되어 있더라도 이 구성 파일의 Build 섹션에 정의된 환경 변수만 참조할 수 있습니다. 이 실행 섹션에 정의된 환경 변수는 참조할 수 없습니다.
> - 이 `pre-run` 매개변수는 수정된 App Runner 빌드에서만 지원됩니다. 애플리케이션이 원본 App Runner 빌드에서 지원하는 런타임 버전을 사용하는 경우 구성 파일에 이 매개변수를 삽입하지 마세요. 자세한 정보는 관리형 런타임 버전 및 앱 러너 빌드을 참조하세요.

구문

```
run:
  pre-run:
      - command
      - …
```

- `command`— 필수. App Runner가 애플리케이션 빌드를 완료한 후 애플리케이션을 실행하는 데 사용하는 명령입니다.

구문

```
run:
  command: command
```

- network— 선택 사항입니다. 애플리케이션이 수신하는 포트를 지정합니다. 여기에는 다음이 포함됩니다.
  - port— 선택 사항입니다. 지정된 경우 애플리케이션에서 수신하는 포트 번호입니다. 기본값은 8080입니다.
  - env— 선택 사항입니다. 지정된 경우 App Runner는 기본 환경 변수에서 동일한 포트 번호를 전달하는 것 외에도 이 환경 변수의 컨테이너에 포트 번호를 전달합니다 (대신 전달하지 않음). PORT 즉env, 지정하는 경우 App Runner는 두 환경 변수에 포트 번호를 전달합니다.

  구문

  ```
  run:
    network:
      port: port-number
      env: env-variable-name
  ```

  Example

  ```
  run:
    network:
      port: 8000
      env: MY_APP_PORT
  ```

- env— 선택 사항입니다. 실행 단계에 대한 사용자 지정 환경 변수 정의 이름-값 스칼라 매핑으로 정의됩니다. 런타임 환경에서 이러한 변수를 이름으로 참조할 수 있습니다.

  > **Note**
  >
  > 이 구성 파일에는 서로 다른 두 위치에 서로 다른 두 개의 env 항목이 있습니다. 한 세트는 Build 섹션에 있고 다른 세트는 Run 섹션에 있습니다.
  >
  > - 빌드 프로세스 중에pre-build, buildpost-build, 및 pre-run 명령으로 빌드 섹션의 env 세트를 참조할 수 있습니다.

중요 - pre-run 명령어는 빌드 섹션에 정의된 환경 변수에만 액세스할 수 있지만 이 파일의 Run 섹션에 있다는 점에 유의하십시오.

- 실행 섹션의 env 세트는 런타임 환경의 run 명령으로 참조할 수 있습니다.

구문

```
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"
```

앱 러너 API

API (AWS App Runner 애플리케이션 프로그래밍 인터페이스) 는 앱 러너 서비스에 요청하기 위한 RESTful API입니다. API를 사용하여 앱 러너 리소스를 만들고, 나열하고, 설명하고, 업데이트하고, 삭제할 수 있습니다. AWS 계정

애플리케이션 코드에서 직접 API를 호출하거나 AWS SDK 중 하나를 사용할 수 있습니다.

전체 API 참조 정보는 AWS App Runner API 참조를 참조하십시오.

AWS 개발자 도구에 대한 자세한 내용은 빌드 기반 도구를 참조하십시오 AWS.

주제

- 를 사용하여 App Runner와 함께 AWS CLI 작업하기
- AWS CloudShell 작업에 사용 AWS App Runner

를 사용하여 App Runner와 함께 AWS CLI 작업하기

명령줄 스크립트의 경우 AWS CLI를 사용하여 App Runner 서비스를 호출합니다. 전체 AWS CLI 참조 정보는 명령 참조의 aprunner를AWS CLI 참조하십시오.

AWS CloudShell 개발 환경에 를 설치하지 않고 대신 AWS CLI 에서 사용할 수 있습니다. AWS Management Console 설치를 피할 수 있을 뿐 아니라 자격 증명을 구성할 필요도 없고 지역을 지정할 필요도 없습니다. AWS Management Console 세션에서는 이 컨텍스트를 에 제공합니다 AWS CLI. 에 대한 자세한 CloudShell 내용 및 사용 예는 을 참조하십시오the section called “사용 AWS CloudShell”.

AWS CloudShell 작업에 사용 AWS App Runner

AWS CloudShell 는 에서 직접 실행할 수 있는 브라우저 기반의 사전 인증된 셸입니다. AWS Management Console선호하는 셸 (Bash 또는 Z 셸 AWS App Runner) 을 사용하여 AWS 서비스 (포함) 에 대해 AWS CLI 명령을 실행할 수 있습니다. PowerShell 또한 명령줄 도구를 다운로드하거나 설치할 필요 없이 이 작업을 수행할 수 있습니다.

AWS CloudShell 에서 실행하면 콘솔에 AWS Management Console 로그인하는 데 사용한 AWS 자격 증명이 새 셸 세션에서 자동으로 사용할 수 있습니다. 이러한 AWS CloudShell 사용자 사전 인증을 통해 AWS CLI 버전 2 (셸의 컴퓨팅 환경에 사전 설치됨) 를 사용하여 App Runner와 같은 AWS 서비스와 상호 작용할 때 자격 증명 구성을 건너뛸 수 있습니다.

주제

IAM 권한 획득: AWS CloudShell

에서 제공하는 AWS Identity and Access Management액세스 관리 리소스를 사용하여 관리자는 IAM 사용자에게 환경 기능에 AWS CloudShell 액세스하고 사용할 수 있는 권한을 부여할 수 있습니다.

관리자가 사용자에게 액세스 권한을 부여하는 가장 빠른 방법은 AWS 관리형 정책을 사용하는 것입니다. AWS 관리형 정책은 AWS에서 생성 및 관리하는 독립 실행형 정책입니다. 다음과 같은 AWS 관리형 정책을 IAM ID에 연결할 CloudShell 수 있습니다.

- `AWSCloudShellFullAccess`: 모든 기능에 대한 전체 액세스 AWS CloudShell 권한과 함께 사용할 수 있는 권한을 부여합니다.

IAM 사용자가 수행할 수 있는 작업의 범위를 제한하려면 `AWSCloudShellFullAccess` 관리형 정책을 템플릿으로 AWS CloudShell사용하는 사용자 지정 정책을 생성할 수 있습니다. 에서 CloudShell 사용자가 수행할 수 있는 작업을 제한하는 방법에 대한 자세한 내용은 사용 설명서의 IAM 정책을 통한 AWS CloudShell 액세스 및 사용 관리를 참조하십시오.AWS CloudShell

> **Note**
>
> 또한 IAM ID에는 App Runner를 호출할 수 있는 권한을 부여하는 정책이 필요합니다. 자세한 정보는 the section called “앱 러너 및 IAM”을 참조하세요.

를 사용하여 앱 러너와 상호작용하기 AWS CloudShell

AWS CloudShell 에서 실행한 후에는 명령줄 인터페이스를 사용하여 즉시 App Runner와 상호 작용을 시작할 수 있습니다. AWS Management Console

다음 예제에서는 in을 사용하여 App Runner 서비스 중 하나에 대한 정보를 검색합니다 AWS CLI . CloudShell

> **Note**
>
> AWS CLI AWS CloudShell in을 사용하면 추가 리소스를 다운로드하거나 설치할 필요가 없습니다. 또한 셸 내에서 이미 인증되었기 때문에 직접 호출을 하기 전에 보안 인증을 구성하지 않아도 됩니다.

Example 를 사용하여 App Runner 서비스 정보를 검색하는 중입니다. AWS CloudShell

1. 에서 AWS Management Console탐색 표시줄에 있는 다음 옵션을 CloudShell 선택하여 시작할 수 있습니다.

   - CloudShell 아이콘을 선택합니다.
   - 검색 **`cloudshell`** 상자에 입력을 시작한 다음 검색 결과에 해당 CloudShell옵션이 표시되면 해당 옵션을 선택합니다.

2. 콘솔 세션 AWS 지역의 AWS 계정에 있는 현재 App Runner 서비스를 모두 나열하려면 명령줄에 다음 명령을 입력합니다. CloudShell

   ```
   $ aws apprunner list-services
   ```

   출력에는 서비스에 대한 요약 정보가 나열됩니다.

   ```
   {
     "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. 특정 App Runner 서비스에 대한 자세한 설명을 보려면 이전 단계에서 검색한 ARN 중 하나를 사용하여 CloudShell 명령줄에 다음 명령을 입력합니다.

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

출력에는 지정한 서비스에 대한 자세한 설명이 나열됩니다.

```
{
  "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
    }
  }
}
```

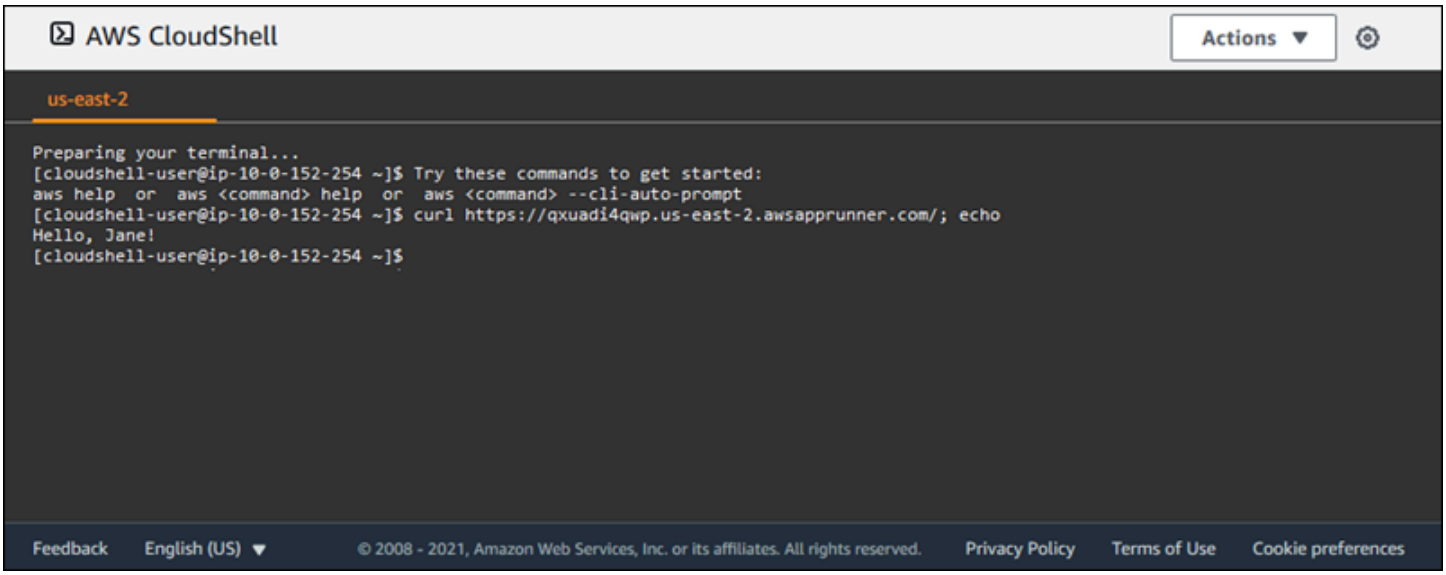
를 사용하여 App Runner 서비스를 확인합니다. AWS CloudShell

App Runner 서비스를 만들면 App Runner는 서비스 웹사이트의 기본 도메인을 생성하여 콘솔에 표시하거나 API 호출 결과에 반환합니다. 를 CloudShell 사용하여 웹 사이트를 호출하고 제대로 작동하는지 확인할 수 있습니다.

예를 들어 에 설명된 대로 App Runner 서비스를 만든 후 다음 명령어를 실행하십시오. 시작하기 CloudShell

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

출력에는 예상 페이지 콘텐츠가 표시되어야 합니다.



문제 해결

이 장에서는 AWS App Runner 서비스를 사용하는 동안 발생할 수 있는 일반적인 오류 및 문제에 대한 문제 해결 단계를 제공합니다. 오류 메시지는 콘솔, API 또는 서비스 페이지의 로그 탭에 표시될 수 있습니다.

문제 해결 조언과 일반적인 지원 질문에 대한 답변은 지식 센터를 참조하세요.

주제

서비스 생성에 실패한 경우

App Runner 서비스를 만들려는 시도가 실패하면 서비스가 `CREATE_FAILED` 상태로 전환됩니다. 이 상태는 콘솔에서 생성 실패로 표시됩니다. 다음 중 하나 이상과 관련된 문제로 인해 서비스를 만들지 못할 수 있습니다.

- 애플리케이션 코드
- 빌드 프로세스
- 구성
- 리소스 할당량
- 서비스가 AWS 서비스 사용하는 기반과 관련된 일시적인 문제

서비스 생성 실패 문제를 해결하려면 다음과 같이 하는 것이 좋습니다.

1. 서비스 이벤트 및 로그를 읽고 서비스 생성에 실패한 원인을 알아보세요.
2. 코드나 구성을 필요에 따라 변경하십시오.
3. 서비스 할당량에 도달한 경우 서비스를 하나 이상 삭제하세요.
4. 다른 리소스 할당량에 도달한 경우 조정 가능한 경우 할당량을 늘릴 수 있습니다.
5. 위 단계를 모두 완료한 후 서비스를 다시 빌드해 보세요. 서비스를 다시 빌드하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “실패한 서비스 재구축”

> **Note**
>
> 문제를 일으킬 수 있는 조정 가능한 리소스 할당량 중 하나는 Fargate 온디맨드 vCPU 리소스입니다.
> vCPU 리소스 수는 App Runner가 서비스에 제공할 수 있는 인스턴스 수를 결정합니다.
> 서비스에 있는 Fargate 온디맨드 vCPU 리소스 수에 대한 조정 가능한 할당량 값입니다.
> AWS Fargate (Fargate) 계정의 vCPU 할당량 설정을 보거나 할당량 증가를 요청하려면 의 Service Quotas 콘솔을 사용하십시오. AWS Management Console자세한 내용은 Amazon Elastic 컨테이너AWS Fargate 서비스 개발자 안내서의 서비스 할당량을 참조하십시오.

> **Important**
>
> 실패한 서비스에 대한 초기 생성 시도 이후에는 추가 요금이 발생하지 않습니다. 실패한 서비스는 사용할 수 없더라도 여전히 서비스 할당량에 포함됩니다. App Runner는 실패한 서비스를 자동으로 삭제하지 않으므로 장애 분석을 완료한 후에는 반드시 삭제해야 합니다.

사용자 지정 도메인 이름

이 섹션에서는 커스텀 도메인에 연결하는 동안 발생할 수 있는 다양한 오류를 해결하고 해결하는 방법을 다룹니다.

> **Note**
>
> 앱 러너 애플리케이션의 보안을 강화하기 위해*.awsapprunner.com 도메인은 공개 접미사 목록 (PSL) 에 등록되어 있습니다. 보안 강화를 위해 App Runner 애플리케이션의 기본 도메인 이름에 민감한 쿠키를 설정해야 하는 경우 `__Host-` 접두사가 있는 쿠키를 사용하는 것이 좋습니다. 이렇게 쿠키를 설정하면 교차 사이트 요청 위조 시도(CSRF)로부터 도메인을 보호하는 데 도움이 됩니다. 자세한 내용은 Mozilla 개발자 네트워크의 Set-Cookie 페이지를 참조하십시오.

사용자 지정 도메인의 Create Fail 오류가 발생했습니다.

- 이 오류가 CAA 레코드 문제 때문인지 확인하십시오. DNS 트리에 CAA 레코드가 없는 경우 메시지를 `fail open` 수신하고 AWS Certificate Manager 인증서를 발급하여 사용자 지정 도메인을 확인합니다. 이렇게 하면 App Runner가 사용자 지정 도메인을 수락할 수 있습니다. DNS 레코드에서 CAA 인증을 사용하는 경우 하나 이상의 도메인의 CAA 레코드에 다음이 포함되는지 확인하세요. `amazon.com` 그렇지 않으면 ACM이 인증서를 발급하지 못합니다. 따라서 App Runner의 사용자 지정 도메인이 생성되지 않습니다.

  다음 예시에서는 DNS 조회 도구인 DiG를 사용하여 필수 항목이 누락된 CAA 레코드를 표시합니다. 이 예에서는 사용자 지정 `example.com` 도메인으로 사용합니다. 예제에서 다음 명령을 실행하여 CAA 레코드를 확인합니다.

```
...
;; 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...
```

- 도메인 레코드를 수정하고 하나 이상의 CAA 레코드에 포함된 내용이 있는지 확인하십시오. `amazon.com`
- 다시 시도하여 사용자 지정 도메인을 App Runner와 연결해 보세요.

CAA 오류를 해결하는 방법에 대한 지침은 다음을 참조하십시오.

- CAA (인증 기관 인증) 문제
- ACM 인증서 발급 또는 갱신 시 발생하는 CAA 오류를 해결하려면 어떻게 해야 합니까?

사용자 지정 도메인의 DNS 인증서 검증 보류 중 오류가 발생했습니다.

- 커스텀 도메인 설정의 중요한 단계를 건너뛰었는지 확인하세요. 또한 DiG와 같은 DNS 조회 도구를 사용하여 DNS 레코드를 잘못 구성했는지 확인하십시오. 특히 다음과 같은 실수가 있는지 확인하세요.
  - 놓친 모든 단계.

  - DNS 레코드의 큰따옴표와 같은 지원되지 않는 문자.
- 실수를 바로 잡으세요.
- 사용자 지정 도메인을 App Runner와 다시 연결해 보세요.

CAA 유효성 검사 오류를 해결하는 방법에 대한 지침은 다음을 참조하십시오.

- DNS 검증
- the section called “사용자 지정 도메인 이름”

기본 문제 해결 명령

- 서비스를 찾을 수 있는지 확인합니다.

```
aws apprunner list-services
```

- 서비스를 설명하고 상태를 확인하세요.

```
aws apprunner describe-service --service-arn
```

- 커스텀 도메인의 상태를 확인하세요.

```
aws apprunner describe-custom-domains --service-arn
```

- 진행 중인 모든 작업을 나열하세요.

```
aws apprunner list-operations --service-arn
```

커스텀 도메인 인증서 갱신

서비스에 커스텀 도메인을 추가하면 App Runner는 DNS 서버에 추가하는 CNAME 레코드 세트를 제공합니다. 이러한 CNAME 레코드에는 인증서 레코드가 포함됩니다. 앱 러너는 AWS Certificate Manager (ACM) 을 사용하여 도메인을 확인합니다. App Runner는 이 도메인의 지속적인 소유권을 보장하기 위해 이러한 DNS 레코드를 검증합니다. DNS 영역에서 CNAME 레코드를 제거하면 App Runner에서 더 이상 DNS 레코드를 검증할 수 없으며 사용자 지정 도메인 인증서가 자동으로 갱신되지 않습니다.

이 섹션에서는 다음과 같은 사용자 지정 도메인 인증서 갱신 문제를 해결하는 방법을 다룹니다.

- the section called “CNAME은 DNS 서버에서 제거됩니다.”.
- the section called “인증서가 만료되었습니다.”.

CNAME은 DNS 서버에서 제거됩니다.

- DescribeCustomDomainsAPI를 사용하거나 앱 러너 콘솔의 사용자 지정 도메인 설정에서 CNAME 레코드를 검색하세요. 저장된 CNAME에 대한 자세한 내용은 을 참조하십시오. CertificateValidationRecords
- DNS 서버에 인증서 검증 CNAME 레코드를 추가합니다. 그러면 App Runner에서 도메인을 소유하고 있는지 확인할 수 있습니다. CNAME 레코드를 추가한 후 DNS 레코드가 전파되는 데 최대 30분이 걸릴 수 있습니다. 또한 App Runner와 ACM이 인증서 갱신 프로세스를 재시도하는 데 몇 시간이 걸릴 수 있습니다. CNAME 레코드를 추가하는 방법에 대한 지침은 을 참조하십시오. the section called “커스텀 도메인 관리”

인증서가 만료되었습니다.

- App Runner 콘솔 또는 API를 사용하여 App Runner 서비스의 사용자 지정 도메인을 연결 해제 (연결 해제) 한 다음 연결 (연결) 하십시오. 앱 러너는 새 인증서 검증 CNAME 레코드를 생성합니다.
- 새 인증서 검증 CNAME 레코드를 DNS 서버에 추가합니다.

  사용자 지정 도메인의 연결을 끊고 (연결 해제) 및 연결 (연결) 하는 방법에 대한 지침은 을 참조하십시오. the section called “커스텀 도메인 관리”

인증서가 성공적으로 갱신되었는지 어떻게 확인할 수 있나요?

인증서 레코드의 상태를 확인하여 인증서가 성공적으로 갱신되었는지 확인할 수 있습니다. curl과 같은 도구를 사용하여 인증서 상태를 확인할 수 있습니다.

인증서 갱신에 대한 자세한 내용은 다음 링크를 참조하십시오.

- 내 ACM 인증서가 갱신 부적격으로 표시되는 이유는 무엇입니까?
- ACM 인증서의 관리형 갱신
- DNS 검증

HTTP/HTTPS 요청 라우팅 오류

이 섹션에서는 HTTP/HTTPS 트래픽을 App Runner 서비스 엔드포인트로 라우팅할 때 발생할 수 있는 오류를 해결하고 해결하는 방법을 다룹니다.

404 HTTP/HTTPS 트래픽을 App Runner 서비스 엔드포인트로 전송할 때 발생하는 오류를 찾을 수 없음

- App `Host Header` Runner가 호스트 헤더 정보를 사용하여 요청을 라우팅하므로 이 HTTP 요청의 서비스 URL을 가리키는지 확인하세요. 와 같은 cURL 대부분의 클라이언트와 웹 브라우저는 자동으로 호스트 헤더를 서비스 URL을 가리킵니다. 클라이언트가 서비스 URL을 로 설정하지 않으면 `404 Not Found` 오류가 발생합니다. `Host Header`

  Example 잘못된 호스트 헤더

  ```
  $ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
  HTTP/1.1 404 Not Found
  transfer-encoding: chunked
  ```

  Example 호스트 헤더 수정

  ```
  $ ~ 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
  ```

- 클라이언트가 공용 또는 사설 서비스로 라우팅되는 요청에 대해 서버 이름 표시기 (SNI) 를 올바르게 설정하고 있는지 확인하십시오. TLS 종료 및 요청 라우팅의 경우 앱 러너는 HTTPS 연결에 설정된 SNI를 사용합니다.

서비스가 Amazon RDS 또는 다운스트림 서비스에 연결되지 않는 경우

Amazon RDS 데이터베이스 또는 기타 다운스트림 애플리케이션 또는 서비스에 연결하지 못하면 서비스에 네트워크 구성 문제가 있을 수 있습니다. 이 주제에서는 네트워크 구성에 문제가 있는지 확인하는 몇 가지 단계와 이를 해결하기 위한 옵션을 안내합니다. App Runner의 아웃바운드 트래픽 구성에 대한 자세한 내용은 을 참조하십시오. 발신 트래픽에 대한 VPC 액세스 활성화

> **Note**
>
> VPC 커넥터 구성을 보려면 App Runner 콘솔 왼쪽 탐색 창에서 네트워크 구성을 선택합니다. 그런 다음 발신 트래픽 탭을 선택합니다. VPC 커넥터를 선택합니다. 다음 페이지에는 VPC 커넥터에 대한 세부 정보가 표시됩니다. 이 페이지에서 VPC를 사용하는 서브넷, 보안 그룹, App Runner 서비스를 보고 자세히 살펴볼 수 있습니다.

애플리케이션이 다른 다운스트림 서비스에 연결할 수 없는 원인을 좁히려면

1. VPC 커넥터에서 사용되는 서브넷이 프라이빗 서브넷인지 확인하십시오. 커넥터가 퍼블릭 서브넷으로 구성된 경우 각 서브넷의 기본 하이퍼플레인 ENI (엘라스틱 네트워크 인터페이스) 에 퍼블릭 IP 공간이 없기 때문에 서비스에 오류가 발생합니다.

   VPC 커넥터가 퍼블릭 서브넷을 사용하는 경우 다음 옵션을 사용하여 이 구성을 수정할 수 있습니다.

   a. 새 프라이빗 서브넷을 만들고 VPC Connector의 퍼블릭 서브넷 대신 사용합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC용 서브넷을 참조하십시오.

   b. NAT 게이트웨이를 통해 기존 퍼블릭 서브넷을 라우팅합니다. 자세한 내용은 Amazon VPC 사용 설명서의 NAT 게이트웨이를 참조하십시오.

2. VPC 커넥터의 보안 그룹 수신 및 송신 규칙이 올바른지 확인합니다. App Runner 콘솔 왼쪽 탐색 창에서 네트워크 구성 > 발신 트래픽을 선택합니다. 목록에서 VPC 커넥터를 선택합니다. 다음 페이지에는 검사하도록 선택할 수 있는 보안 그룹이 나열되어 있습니다.

3. 연결을 시도하는 RDS 인스턴스 또는 기타 다운스트림 서비스에 대한 보안 그룹 인바운드 및 아웃바운드 규칙이 올바른지 확인하십시오. 자세한 내용은 App Runner 애플리케이션이 연결하려는 다운스트림 서비스의 서비스 가이드를 참조하십시오.
4. App Runner 구성 외에 다른 유형의 네트워크 설정 문제가 없는지 확인하려면 App Runner 외부의 RDS 또는 다운스트림 서비스에 연결해 보세요.

   a. 동일한 VPC의 Amazon EC2 인스턴스에서 RDS 인스턴스 또는 서비스에 연결해 보십시오.

   b. 서비스 VPC 엔드포인트에 연결하려는 경우 동일한 VPC의 EC2 인스턴스에서 동일한 엔드포인트에 액세스하여 연결을 확인합니다.
5. 4단계의 연결 테스트 중 하나에 실패할 경우 계정 내 다른 리소스가 있는 App Runner 구성 이외의 문제가 있을 가능성이 큽니다. AWS다른 네트워크 구성과 관련된 문제를 추가로 격리하고 수정하려면 AWS Support에 문의하세요.
6. 4단계의 지침에 따라 RDS 인스턴스 또는 다운스트림 서비스에 성공적으로 연결했다면 이 단계의 지침을 따르십시오. 하이퍼플레인 ENI 흐름 로그를 활성화하고 검사하여 트래픽이 ENI로 들어오는지 확인합니다.

   > **Note**
   >
   > 이 단계를 완료하고 필요한 ENI 흐름 로그 정보를 얻으려면 App Runner 서비스가 성공적으로 시작된 후에 RDS 또는 다운스트림 서비스에 대한 연결 시도가 이루어져야 합니다. 애플리케이션이 Running 상태일 때 RDS 또는 다운스트림 서비스에 대한 연결 작업을 수행해야 합니다. 그렇지 않으면 App Runner의 롤백 워크플로의 일부로 ENI를 정리할 수 있습니다. 이렇게 하면 ENI를 추가 조사에 계속 사용할 수 있습니다.

   a. AWS 콘솔에서 EC2 콘솔을 시작합니다.

   b. 왼쪽 탐색 창의 네트워크 및 보안 그룹에서 네트워크 인터페이스를 선택합니다.

   c. 인터페이스 유형 및 설명 열로 스크롤하여 VPC 커넥터와 연결된 서브넷의 ENI를 찾습니다. 이름 지정 패턴은 다음과 같습니다.

      - 인터페이스 유형: 파게이트
      - 설명: 다음으로 시작 AWSAppRunner ENI(예: AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)

   d. 행 앞에 있는 확인란을 사용하여 적용되는 ENI를 선택합니다.

   e. 작업 메뉴에서 흐름 로그 만들기를 선택합니다.

f. 프롬프트에 정보를 입력하고 페이지 하단에서 플로그 만들기를 선택합니다.

g. 생성된 흐름 로그를 검사합니다.

- 연결을 테스트할 때 트래픽이 ENI로 들어왔다면 문제는 ENI 설정과 관련이 없습니다. App Runner 서비스 외에 AWS 계정의 다른 리소스에 네트워크 구성 문제가 있을 수 있습니다. AWS Support에 문의하여 추가 지원을 받으십시오.
- 연결을 테스트할 때 ENI에 트래픽이 유입되지 않았다면 AWS Support에 문의하여 Fargate 서비스에 알려진 문제가 있는지 확인하는 것이 좋습니다.

h. 네트워크 도달성 분석기 도구를 사용하십시오. 이 도구를 사용하면 가상 네트워크 경로의 소스에 연결할 수 없는 경우 차단 구성 요소를 식별하여 네트워크 구성 오류를 확인할 수 있습니다. 자세한 내용은 Reachability Analyzer란 무엇입니까? 를 참조하십시오. Amazon VPC 접근성 분석기 가이드에서 확인할 수 있습니다.

   앱 러너 ENI를 소스로 입력하고 RDS ENI를 대상으로 입력합니다.

7. 문제의 범위를 더 좁힐 수 없거나 이전 단계를 완료한 후에도 여전히 RDS 또는 다운스트림 서비스에 연결할 수 없는 경우 Support에 문의하여 추가 AWS 지원을 받는 것이 좋습니다.

앱 러너의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처를 활용할 수 있습니다.

보안은 기업과 기업 간의 공동 책임입니다. AWS 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오. AWS App Runner
- 클라우드에서의 보안 — 사용하는 AWS 서비스에 따라 책임이 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 App Runner를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 App Runner를 구성하는 방법을 보여줍니다. 또한 App Runner 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

앱 러너의 데이터 보호

AWS 공동 책임 모델 의 데이터 보호에 적용됩니다 AWS App Runner. 이 모델에 설명된 대로 AWS 는 모든 데이터를 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은AWS 보안 블로그에서 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API 또는 SDK를 AWS 서비스 사용하여 App Runner 또는 기타 작업을 수행하는 경우가 포함됩니다. AWS CLI AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

주제

- 암호화를 사용하여 데이터 보호
- 인터네트워크 트래픽 개인 정보 보호

암호화를 사용하여 데이터 보호

AWS App Runner 지정한 저장소에서 애플리케이션 소스 (소스 이미지 또는 소스 코드) 를 읽고 서비스에 배포하기 위해 저장합니다. 자세한 정보는 아키텍처 및 개념을 참조하세요.

데이터 보호란 전송 중 (App Runner를 오가는 데이터) 및 유휴 상태 (AWS 데이터 센터에 저장되는 동안) 를 보호하는 것을 말합니다.

데이터 보호에 대한 자세한 내용은 을 참조하십시오. the section called “데이터 보호”

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

전송 중 암호화

전송 중에 데이터를 보호하는 방법에는 두 가지가 있습니다. 하나는 전송 계층 보안 (TLS) 을 사용하여 연결을 암호화하는 방법과 클라이언트 측 암호화 (객체를 전송하기 전에 암호화하는 방법) 를 사용하는 것입니다. 두 방법 모두 애플리케이션 데이터를 보호하는 데 유효합니다 연결을 보호하려면 응용 프로그램, 개발자 및 관리자, 최종 사용자가 객체를 보내거나 받을 때마다 TLS를 사용하여 암호화하십시오. App Runner는 TLS를 통해 트래픽을 수신하도록 애플리케이션을 설정합니다.

클라이언트 측 암호화는 배포를 위해 App Runner에 제공하는 소스 이미지 또는 코드를 보호하는 데 사용할 수 있는 유효한 방법이 아닙니다. App Runner는 애플리케이션 소스에 대한 액세스 권한이 필요하므로 암호화할 수 없습니다. 따라서 개발 또는 배포 환경과 App Runner 간의 연결을 보호해야 합니다.

저장 중 암호화 및 키 관리

애플리케이션 데이터를 보호하기 위해 App Runner는 애플리케이션 소스 이미지 또는 소스 번들의 저장된 모든 사본을 암호화합니다. App Runner 서비스를 생성할 때 다음을 제공할 수 있습니다. AWS KMS key앱을 제공하면 App Runner는 제공된 키를 사용하여 소스를 암호화합니다. 키를 제공하지 않으면 App Runner에서 대신 키를 사용합니다. AWS 관리형 키

App Runner 서비스 생성 매개변수에 대한 자세한 내용은 을 참조하십시오. CreateService AWS Key Management Service (AWS KMS) 에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하십시오.

인터네트워크 트래픽 개인 정보 보호

App Runner는 Amazon VPC (Virtual Private Cloud) 를 사용하여 앱 러너 애플리케이션의 리소스 간에 경계를 만들고 리소스, 온프레미스 네트워크 및 인터넷 간의 트래픽을 제어합니다. Amazon VPC 보안에 대한 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC의 인터네트워크 트래픽 개인 정보 보호를 참조하십시오.

App Runner 애플리케이션을 사용자 지정 Amazon VPC와 연결하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “발신 트래픽”

VPC 엔드포인트를 사용하여 App Runner에 대한 요청을 보호하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “VPC 엔드포인트”

데이터 보호에 대한 자세한 내용은 을 참조하십시오. the section called “데이터 보호”

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

앱 러너의 ID 및 액세스 관리

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. AWS IAM 관리자는 App Runner 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 AWS 서비스 없이 사용할 수 있습니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

주제

- 고객
- 자격 증명을 통한 인증
- 정책을 사용한 액세스 관리
- 앱 러너가 IAM과 함께 작동하는 방식
- App Runner ID 기반 정책 예제
- App Runner의 서비스 연결 역할 사용
- AWS 에 대한 관리형 정책 AWS App Runner
- App Runner ID 및 액세스 문제 해결

고객

앱 러너에서 수행하는 작업에 따라 AWS Identity and Access Management (IAM) 사용 방식이 다릅니다.

서비스 사용자 - App Runner 서비스를 사용하여 작업을 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 App Runner 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할

수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. App Runner의 기능에 액세스할 수 없는 경우 을 참조하십시오. App Runner ID 및 액세스 문제 해결

서비스 관리자 — 회사에서 App Runner 리소스를 담당하는 경우 App Runner에 대한 전체 액세스 권한이 있을 것입니다. 서비스 사용자가 액세스해야 하는 App Runner 기능과 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 회사에서 App Runner와 함께 IAM을 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. 앱 러너가 IAM과 함께 작동하는 방식

IAM 관리자 — IAM 관리자라면 App Runner에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대해 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 App Runner ID 기반 정책의 예를 보려면 을 참조하십시오. App Runner ID 기반 정책 예제

자격 증명을 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용자 설명서의 AWS에서 다중 인증(MFA) 사용을 참조합니다.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메

일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 가진 사용자 내의 자격 증명입니다. AWS 계정 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 아이덴티티에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 자격 증명이 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.

- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조합니다.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용자 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조합니다.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACLs)

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티에 대한 권한 경계를 참조합니다.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 항목을 포함하여 구성원 계정의 엔티티에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 연합된 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용자 설명서의 세션 정책을 참조합니다.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

앱 러너가 IAM과 함께 작동하는 방식

IAM을 사용하여 액세스를 관리하기 전에 App AWS App Runner Runner에서 사용할 수 있는 IAM 기능이 무엇인지 이해해야 합니다. App Runner 및 기타 AWS 서비스가 IAM과 연동되는 방식을 자세히 알아보려면 IAM 사용 설명서의 IAM과 연동되는AWS 서비스를 참조하십시오.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

주제

- 앱 러너 ID 기반 정책
- 앱 러너 리소스 기반 정책
- 앱 러너 태그를 기반으로 한 권한 부여
- 앱 러너 사용자 권한
- 앱 러너 IAM 역할

앱 러너 ID 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. App Runner는 특정 작업, 리소스, 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

App Runner의 정책 작업은 작업 앞에 다음 접두사를 사용합니다. `apprunner:` 예를 들어 누군가에게 Amazon EC2 `RunInstances` API 작업을 통해 Amazon EC2 인스턴스를 실행할 권한을 부여하려면 해당 정책에 `ec2:RunInstances` 작업을 포함하세요. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. App Runner는 이 서비스로 수행할 수 있는 작업을 설명하는 자체 작업 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "apprunner:Describe*"
```

App Runner 작업 목록을 보려면 서비스 권한 부여 AWS App Runner 참조에 정의된 작업을 참조하십시오.

리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

App Runner 리소스의 ARN 구조는 다음과 같습니다.

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

ARN 형식에 대한 자세한 내용은 의 Amazon 리소스 이름 (ARN) 및 AWS 서비스 네임스페이스를 참조하십시오. AWS 일반 참조

예를 들어 명세서에서 `my-service` 서비스를 지정하려면 다음 ARN을 사용하십시오.

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

특정 계정에 속하는 모든 서비스를 지정하려면 와일드카드 (*) 를 사용합니다.

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

리소스 생성 작업과 같은 일부 App Runner 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

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

App Runner 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 AWS App Runner참조에 정의된 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS App Runner가 정의한 작업을 참조하십시오.

조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS 는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

App Runner는 일부 글로벌 조건 키 사용을 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 설명서의AWS 글로벌 조건 컨텍스트 키를 참조하십시오.

App Runner는 서비스별 조건 키 세트를 정의합니다. 또한 App Runner는 조건 키를 사용하여 구현되는 태그 기반 액세스 제어를 지원합니다. 자세한 내용은 the section called “앱 러너 태그를 기반으로 한 권한 부여”단원을 참조하세요.

App Runner 조건 키 목록을 보려면 서비스 권한 부여 참조의 조건 키를 참조하십시오 AWS App Runner. 조건 키를 사용할 수 있는 작업 및 리소스에 대해 알아보려면 작업 정의 기준을 참조하십시오. AWS App Runner

예제

App Runner ID 기반 정책의 예를 보려면 을 참조하십시오. App Runner ID 기반 정책 예제

앱 러너 리소스 기반 정책

App Runner는 리소스 기반 정책을 지원하지 않습니다.

앱 러너 태그를 기반으로 한 권한 부여

App Runner 리소스에 태그를 첨부하거나 요청의 태그를 App Runner에 전달할 수 있습니다. 태그를 기반으로 액세스를 제어하려면 `apprunner:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. App Runner 리소스에 태그를 지정하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “구성”

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예제는 태그를 기반으로 App Runner 서비스에 대한 액세스를 제어합니다. 섹션에서 확인할 수 있습니다.

앱 러너 사용자 권한

앱 러너를 사용하려면 IAM 사용자에게 앱 러너 작업에 대한 권한이 필요합니다. 사용자에게 권한을 부여하는 일반적인 방법은 IAM 사용자 또는 그룹에 정책을 연결하는 것입니다. 사용자 권한 관리에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 권한 변경을 참조하십시오.

App Runner는 사용자에게 연결할 수 있는 두 가지 관리형 정책을 제공합니다.

- `AWSAppRunnerReadOnlyAccess`— App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있는 권한을 부여합니다.

- `AWSAppRunnerFullAccess`— 모든 앱 러너 작업에 권한을 부여합니다.

사용자 권한을 보다 세밀하게 제어하기 위해 사용자 지정 정책을 생성하여 사용자에게 연결할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

사용자 정책의 예는 을 참조하십시오. the section called “사용자 정책”

AWSAppRunnerReadOnlyAccess

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

앱 러너 IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 엔티티입니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수 없습니다.

App Runner는 서비스 연결 역할을 지원합니다. App Runner 서비스 연결 역할을 만들거나 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. the section called “서비스 링크 역할 사용”

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 사용자는 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

App Runner는 몇 가지 서비스 역할을 지원합니다.

액세스 역할

액세스 역할은 앱 러너가 사용자 계정의 Amazon Elastic Container Registry (Amazon ECR)에 있는 이미지에 액세스하는 데 사용하는 역할입니다. Amazon ECR에 있는 이미지에 액세스하는 데 필요하지만 Amazon ECR Public에서는 필요하지 않습니다. Amazon ECR에서 이미지를 기반으로 서비스를 생성하기 전에 IAM을 사용하여 서비스 역할을 생성하고 이 역할의 `AWSAppRunnerServicePolicyForECRAccess` 관리형 정책을 사용하십시오. 그러면 SourceConfiguration파라미터 AuthenticationConfiguration멤버에서 CreateServiceAPI를 호출하거나 App Runner 콘솔을 사용하여 서비스를 생성할 때 이 역할을 App Runner에 전달할 수 있습니다.

AWSAppRunnerServicePolicyForECRAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 액세스 역할에 대한 사용자 지정 정책을 직접 만드는 경우 작업에 대한 사용자 지정 정책을 `"Resource": "*"` 지정해야 합니다`ecr:GetAuthorizationToken`. 토큰은 액세스 권한이 있는 모든 Amazon ECR 레지스트리에 액세스하는 데 사용할 수 있습니다.

액세스 역할을 생성할 때는 App Runner 서비스 보안 주체를 신뢰할 수 있는 `build.apprunner.amazonaws.com` 개체로 선언하는 신뢰 정책을 추가해야 합니다.

액세스 역할에 대한 신뢰 정책

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

App Runner 콘솔을 사용하여 서비스를 만드는 경우 콘솔에서 자동으로 액세스 역할을 생성하고 새 서비스에 사용할 액세스 역할을 선택할 수 있습니다. 콘솔에는 계정의 다른 역할도 나열되며 원하는 경우 다른 역할을 선택할 수 있습니다.

인스턴스 역할

인스턴스 역할은 App Runner가 서비스의 컴퓨팅 인스턴스에 필요한 AWS 서비스 작업에 대한 권한을 제공하는 데 사용하는 선택적 역할입니다. 애플리케이션 코드가 AWS 작업 (API) 을 호출하는 경우 App Runner에 인스턴스 역할을 제공해야 합니다. 필요한 권한을 인스턴스 역할에 내장하거나, 사용자 지정 정책을 만들어 인스턴스 역할에서 사용하십시오. 코드에서 어떤 호출을 사용하는지 예측할 방법이 없습니다. 따라서 이러한 용도의 관리형 정책은 제공하지 않습니다.

App Runner 서비스를 생성하기 전에 IAM을 사용하여 필요한 사용자 지정 또는 내장된 정책을 포함하는 서비스 역할을 생성하십시오. 그런 다음 InstanceConfiguration파라미터 `InstanceRoleArn` 멤버에서 CreateServiceAPI를 호출하거나 App Runner 콘솔을 사용하여 서비스를 생성할 때 이 역할을 인스턴스 역할로 App Runner에 전달할 수 있습니다.

인스턴스 역할을 만들 때는 App Runner 서비스 보안 주체를 신뢰할 수 있는 `tasks.apprunner.amazonaws.com` 개체로 선언하는 신뢰 정책을 추가해야 합니다.

인스턴스 역할에 대한 신뢰 정책

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tasks.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

App Runner 콘솔을 사용하여 서비스를 만드는 경우 콘솔에는 계정의 역할이 나열되며 이 용도로 만든 역할을 선택할 수 있습니다.

서비스 생성에 대한 자세한 내용은 을 참조하십시오the section called “생성”.

App Runner ID 기반 정책 예제

기본적으로 IAM 사용자 및 역할에는 리소스를 생성하거나 수정할 권한이 없습니다. AWS App Runner 또한 AWS Management Console AWS CLI, 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

주제

- 정책 모범 사례
- 사용자 정책
- 태그를 기반으로 App Runner 서비스에 대한 액세스를 제어합니다.

정책 모범 사례

ID 기반 정책은 누군가가 계정에서 App Runner 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하고 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS managed policies(관리형 정책) 또는 AWS managed policies for job functions(직무에 대한 관리형 정책)를 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 Policies and permissions in IAM(IAM의 정책 및 권한)을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하tpdy.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 Configuring MFA-protected API access(MFA 보호 API 액세스 구성)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

사용자 정책

App Runner 콘솔에 액세스하려면 IAM 사용자에게 최소한의 권한이 있어야 합니다. 이러한 권한을 통해 내 App Runner 리소스의 세부 정보를 나열하고 볼 수 있어야 합니다. AWS 계정필요한 최소 권한보다 더 제한적인 ID 기반 정책을 만들면 해당 정책을 사용하는 사용자에게 콘솔이 의도한 대로 작동하지 않습니다.

App Runner는 사용자에게 연결할 수 있는 두 가지 관리형 정책을 제공합니다.

- `AWSAppRunnerReadOnlyAccess`— App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있는 권한을 부여합니다.

- `AWSAppRunnerFullAccess`— 모든 앱 러너 작업에 권한을 부여합니다.

사용자가 App Runner 콘솔을 사용할 수 있도록 하려면 최소한 `AWSAppRunnerReadOnlyAccess` 관리형 정책을 사용자에게 첨부하십시오. 대신 `AWSAppRunnerFullAccess` 관리형 정책을 연결하거나 특정 추가 권한을 추가하여 사용자가 리소스를 생성, 수정 및 삭제하도록 허용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 대신, 사용자가 수행하도록 허용하려는 API 작업과 일치하는 작업에만 액세스를 허용하세요.

다음 예는 사용자 지정 사용자 정책을 보여줍니다. 이를 출발점으로 삼아 사용자 지정 사용자 정책을 정의할 수 있습니다. 예제를 복사하거나, 작업을 제거하고, 리소스의 범위를 좁히고, 조건을 추가하세요.

예: 콘솔 및 연결 관리 사용자 정책

이 예제 정책은 콘솔 액세스를 활성화하고 연결 생성 및 관리를 허용합니다. App Runner 서비스 생성 및 관리는 허용되지 않습니다. 소스 코드 자산에 대한 App Runner 서비스 액세스를 관리하는 역할을 하는 사용자에게 첨부할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner:DeleteConnection"
      ],
      "Resource": "*"
    }
  ]
}
```

예: 조건 키를 사용하는 사용자 정책

이 섹션의 예는 일부 리소스 속성 또는 작업 매개변수에 따라 달라지는 조건부 권한을 보여줍니다.

이 예제 정책은 App Runner 서비스를 생성할 수 있도록 허용하지만 이름이 지정된 연결 사용은 거부합니다. `prod`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
      "Effect": "Allow",
      "Action": "apprunner:CreateService",
      "Resource": "*",
      "Condition": {
        "ArnNotLike": {
          "apprunner:ConnectionArn": "arn:aws:apprunner:*:*:connection/prod/*"
        }
      }
    }
  ]
}
```

이 예제 정책은 이름이 지정된 Auto Scaling `preprod` 구성으로만 이름이 지정된 App Runner 서비스를 업데이트할 수 있도록 합니다. `preprod`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
      "Effect": "Allow",
      "Action": "apprunner:UpdateService",
      "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
      "Condition": {
        "ArnLike": {
          "apprunner:AutoScalingConfigurationArn":
 "arn:aws:apprunner:*:*:autoscalingconfiguration/preprod/*"
        }
      }
    }
  ]
}
```

태그를 기반으로 App Runner 서비스에 대한 액세스를 제어합니다.

ID 기반 정책의 조건을 사용하여 태그를 기반으로 App Runner 리소스에 대한 액세스를 제어할 수 있습니다. 이 예제는 App Runner 서비스 삭제를 허용하는 정책을 만드는 방법을 보여줍니다. 하지만

Owner 서비스 태그가 해당 사용자의 사용자 이름 값을 가지고 있는 경우에만 권한이 부여됩니다. 이 정책은 콘솔에서 이 태스크를 완료하는 데 필요한 권한도 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:*:*:service/*",
      "Condition": {
        "StringEquals": {"apprunner:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

이 정책을 계정의 IAM 사용자에게 연결할 수 있습니다. 라는 사용자가 App Runner 서비스를 richard-roe 삭제하려고 하면 서비스에 또는 태그를 Owner=richard-roe 지정해야 합니다. owner=richard-roe 그렇지 않으면 액세스가 거부됩니다. 조건 키 이름은 대소문자를 구분하지 않기 때문에 태그 키 Owner은(는) Owner 및 owner 모두와 일치합니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

App Runner의 서비스 연결 역할 사용

AWS App Runner AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 App Runner에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 App Runner에서 미리 정의하며 서비스가 사용자를 대신하여 다른 서비스를 호출하는 데 필요한 모든 권한을 포함합니다. AWS

주제

- 관리를 위한 역할 사용
- 네트워킹을 위한 역할 사용

관리를 위한 역할 사용

AWS App Runner AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 App Runner에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 App Runner에서 미리 정의하며 서비스가 사용자를 대신하여 다른 서비스를 호출하는 데 필요한 모든 권한을 포함합니다. AWS

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 App Runner를 더 쉽게 설정할 수 있습니다. App Runner는 서비스 연결 역할의 권한을 정의하며, 달리 정의되지 않는 한 App Runner만 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 제거할 수 없으므로 App Runner 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

App Runner에 대한 서비스 연결 역할 권한

App Runner는 이름이 지정된 서비스 연결 역할을 사용합니다. AWSServiceRoleForAppRunner

이 역할을 통해 App Runner는 다음 작업을 수행할 수 있습니다.

- CloudWatch 로그를 Amazon Logs 로그 그룹에 푸시합니다.
- Amazon CloudWatch Events 규칙을 생성하여 Amazon Elastic Container 레지스트리 (Amazon ECR) 이미지 푸시를 구독하십시오.
- 로 추적 정보를 전송하십시오. AWS X-Ray

AWSServiceRoleForAppRunner 서비스 연결 역할은 다음 서비스를 신뢰하여 역할을 수임합니다.

- `apprunner.amazonaws.com`

AWSServiceRoleForAppRunner 서비스 연결 역할의 권한 정책에는 App Runner가 사용자를 대신하여 작업을 완료하는 데 필요한 모든 권한이 포함되어 있습니다.

AppRunnerServiceRolePolicy 관리형 정책

```
{
```

```
  "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*"
    }
  ]
}
```

X-Ray 추적 정책

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 링크 역할 권한을 참조하세요.

App Runner를 위한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console AWS CLI, 또는 AWS API에서 App Runner 서비스를 만들면 App Runner가 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. App Runner 서비스를 만들면 App Runner에서 서비스 연결 역할을 다시 생성합니다.

App Runner의 서비스 연결 역할 편집

App Runner에서는 서비스 연결 역할을 편집할 수 없습니다. AWSServiceRoleForAppRunner 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

App Runner의 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

App Runner에서 이는 계정의 모든 App Runner 서비스를 삭제하는 것을 의미합니다. App Runner 서비스 삭제에 대한 자세한 내용은 을 참조하십시오. the section called “삭제”

> **Note**
>
> 리소스를 삭제하려고 할 때 App Runner 서비스가 역할을 사용하는 경우 삭제가 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

수동으로 서비스 연결 역할 삭제

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AWSServiceRoleForAppRunner 서비스 연결 역할을 삭제하십시오. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

App Runner 서비스 연결 역할이 지원되는 지역

App Runner는 서비스를 사용할 수 있는 모든 지역에서 서비스 연결 역할을 사용할 수 있도록 지원합니다. 자세한 내용은AWS 일반 참조에서 AWS App Runner 엔드포인트 및 할당량을 참조하세요.

네트워킹을 위한 역할 사용

AWS App Runner AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 App Runner에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 App Runner에서 미리 정의하며 서비스가 사용자를 대신하여 다른 서비스를 호출하는 데 필요한 모든 권한을 포함합니다. AWS

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 App Runner를 더 쉽게 설정할 수 있습니다. App Runner는 서비스 연결 역할의 권한을 정의하며, 달리 정의되지 않는 한 App Runner만 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 제거할 수 없으므로 App Runner 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

App Runner에 대한 서비스 연결 역할 권한

App Runner는 이름이 지정된 서비스 연결 역할을 사용합니다.
AWSServiceRoleForAppRunnerNetworking

이 역할을 통해 App Runner는 다음 작업을 수행할 수 있습니다.

- VPC를 App Runner 서비스에 연결하고 네트워크 인터페이스를 관리합니다.

AWSServiceRoleForAppRunnerNetworking 서비스 연결 역할은 다음 서비스를 신뢰하여 역할을 맡습니다.

- `networking.apprunner.amazonaws.com`

지정된 역할 권한 정책에는 App Runner가 사용자를 대신하여 작업을 완료하는 데 필요한 모든 권한이 AppRunnerNetworkingServiceRolePolicy 포함되어 있습니다.

AppRunnerNetworkingServiceRolePolicy

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

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 링크 역할 권한을 참조하세요.

App Runner를 위한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console AWS CLI, 또는 AWS API에서 VPC 커넥터를 만들면 App Runner가 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. VPC 커넥터를 만들면 App Runner가 서비스 연결 역할을 다시 생성합니다.

App Runner의 서비스 연결 역할 편집

App Runner에서는 서비스 연결 역할을 편집할 수 없습니다. AWSServiceRoleForAppRunnerNetworking 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

App Runner의 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

App Runner에서 이는 계정의 모든 앱 러너 서비스에서 VPC 커넥터를 분리하고 VPC 커넥터를 삭제하는 것을 의미합니다. 자세한 정보는 the section called “발신 트래픽”을 참조하세요.

> **Note**
>
> App Runner 서비스가 리소스를 삭제하려고 할 때 역할을 사용하는 경우 삭제가 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

수동으로 서비스 연결 역할 삭제

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AWSServiceRoleForAppRunnerNetworking 서비스 연결 역할을 삭제하십시오. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

App Runner 서비스 연결 역할이 지원되는 지역

App Runner는 서비스를 사용할 수 있는 모든 지역에서 서비스 연결 역할을 사용할 수 있도록 지원합니다. 자세한 내용은AWS 일반 참조에서 AWS App Runner 엔드포인트 및 할당량을 참조하세요.

AWS 에 대한 관리형 정책 AWS App Runner

AWS 관리형 정책은 에서 만들고 관리하는 독립 실행형 정책입니다. AWS AWS 관리형 정책은 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었으므로 사용자, 그룹 및 역할에 권한을 할당하기 시작할 수 있습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수도 있다는 점에 유의하세요. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

관리형 정책에 정의된 권한은 변경할 수 없습니다. AWS AWS 관리형 정책에 정의된 권한을 업데이트하는 경우 AWS 해당 업데이트는 정책이 연결된 모든 주체 ID (사용자, 그룹, 역할) 에 영향을 미칩니다. AWS 새 API 작업이 시작되거나 기존 서비스에 새 AWS 서비스 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 가장 높습니다.

자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책을 참조하세요.

App Runner가 AWS 관리형 정책을 업데이트합니다.

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 App Runner의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하세요. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 App Runner 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
AWSAppRunnerReadOnlyAccess - 새 정책	App Runner는 사용자가 App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용하는 새 정책을 추가했습니다.	2022년 2월 24일
AWSAppRunnerFullAccess-기존 정책 업데이트	App Runner는 `AWSServiceRoleForAppRunnerNetworking` 서비스 연결 역할을 생성할 수 있도록 `iam:CreateServiceLinkedRole` 작업에 대한 리소스 목록을 업데이트했습니다.	2022년 2월 8일

변경 사항	설명	날짜
AppRunnerNetworkingServiceRolePolicy - 새 정책	App Runner는 App Runner가 Amazon Virtual Private Cloud를 호출하여 앱 러너 서비스에 VPC를 연결하고 앱 러너 서비스를 대신하여 네트워크 인터페이스를 관리할 수 있도록 허용하는 새 정책을 추가했습니다. 정책은 서비스 연결 역할에 사용됩니다. `AWSServiceRoleForAppRunnerNetworking`	2022년 2월 8일
AWSAppRunnerFullAccess - 새 정책	App Runner는 사용자가 모든 앱 러너 작업을 수행할 수 있도록 허용하는 새 정책을 추가했습니다.	2022년 1월 10일
AppRunnerServiceRolePolicy - 새 정책	앱 러너는 앱 러너 서비스를 대신하여 앱 러너가 Amazon CloudWatch Logs 및 Amazon CloudWatch Events를 호출할 수 있도록 허용하는 새 정책을 추가했습니다. 정책은 서비스 연결 역할에 사용됩니다. `AWSServiceRoleForAppRunner`	2021년 3월 1일
AWSAppRunnerServicePolicyForECRAccess - 새 정책	App Runner는 App Runner가 사용자 계정의 Amazon Elastic Container Registry (Amazon ECR) 이미지에 액세스할 수 있도록 허용하는 새 정책을 추가했습니다.	2021년 3월 1일
앱 러너가 변경 사항 추적을 시작했습니다.	App Runner는 AWS 관리형 정책의 변경 사항을 추적하기 시작했습니다.	2021년 3월 1일

App Runner ID 및 액세스 문제 해결

다음 정보를 사용하면 및 IAM을 사용할 때 발생할 수 있는 일반적인 문제를 AWS App Runner 진단하고 해결하는 데 도움이 됩니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

주제

- 저는 App Runner에서 작업을 수행할 권한이 없습니다.
- 외부 사용자가 내 App Runner AWS 계정 리소스에 액세스할 수 있도록 허용하고 싶습니다.

저는 App Runner에서 작업을 수행할 권한이 없습니다.

작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 도움을 요청하세요. 관리자는 사용자에게 AWS 로그인 자격 증명을 제공한 사람입니다.

다음 예제 오류는 이라는 IAM 사용자가 콘솔을 사용하여 App Runner 서비스에 대한 세부 정보를 `marymajor` 보려고 하지만 권한이 없는 경우 발생합니다. `apprunner:DescribeService`

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

이 경우 Mary는 관리자에게 작업을 사용하여 *my-example-service* 리소스에 액세스할 수 있도록 정책을 업데이트해 달라고 요청합니다`apprunner:DescribeService`.

외부 사용자가 내 App Runner AWS 계정 리소스에 액세스할 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- App Runner가 이러한 기능을 지원하는지 알아보려면 을 참조하십시오. 앱 러너가 IAM과 함께 작동하는 방식
- 소유한 리소스에 대한 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서의 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공을 참조하십시오. AWS 계정
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

앱 러너에서의 로깅 및 모니터링

모니터링은 AWS App Runner 서비스의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS 솔루션의 모든 부분에서 모니터링 데이터를 수집하면 장애가 발생할 경우 장애를 더 쉽게 디버깅할 수 있습니다. App Runner는 App Runner 서비스를 모니터링하고 잠재적 사고에 대응하기 위한 여러 AWS 도구와 통합됩니다.

아마존 CloudWatch 알람

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 서비스 메트릭을 관찰할 수 있습니다. 지표가 지정된 기간 동안 지정된 임계값을 초과하는 경우 알림을 받게 됩니다.

App Runner는 전체 서비스와 웹 서비스를 실행하는 인스턴스 (확장 단위) 에 대한 다양한 측정치를 수집합니다. 자세한 정보는 지표 () CloudWatch을 참조하세요.

애플리케이션 로그

App Runner는 애플리케이션 코드의 출력을 수집하여 Amazon CloudWatch Logs로 스트리밍합니다. 이 출력의 내용은 사용자에게 달려 있습니다. 예를 들어, 웹 서비스에 대한 요청의 세부 기록을 포함할 수 있습니다. 이러한 로그 레코드는 보안 및 액세스 감사에 유용할 수 있습니다. 자세한 정보는 로그 (로그) CloudWatch 을 참조하세요.

AWS CloudTrail 작업 로그

App Runner는 App Runner에서 사용자, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합됩니다. AWS CloudTrail CloudTrail 앱 러너에 대한 모든 API 호출을 이벤트로 캡처합니다. CloudTrail 콘솔에서 가장 최근 이벤트를 볼 수 있으며, Amazon Simple Storage Service (Amazon S3) 버킷으로 CloudTrail 이벤트를 지속적으로 전송할 수 있는 트레일을 생성할 수 있습니다. 자세한 내용은 API 작업 () CloudTrail을(를) 참조하세요.

앱 러너에 대한 규정 준수 검증

제3자 감사자는 여러 규정 AWS 준수 프로그램의 AWS App Runner 일환으로 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의 범위별, 규정 준수AWS 서비스 프로그램별 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

앱 러너의 레질리언스

AWS 글로벌 인프라는 가용 영역을 중심으로 구축됩니다 AWS 리전 . AWS 리전 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

가용 영역에 대한 AWS 리전 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

AWS App Runner 사용자를 대신하여 AWS 글로벌 인프라 사용을 관리하고 자동화합니다. App Runner를 사용하면 제공되는 AWS 가용성 및 내결함성 메커니즘을 활용할 수 있습니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

의 인프라 보안 AWS App Runner

관리형 서비스로서 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차에 따라 보호됩니다. AWS App Runner

AWS 게시된 API 호출을 사용하여 네트워크를 통해 App Runner에 액세스할 수 있습니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오. 보안

VPC 엔드포인트와 함께 앱 러너 사용

AWS 애플리케이션은 Amazon VPC (가상 사설 클라우드) 의 VPC에서 AWS 서비스 실행되는 다른 AWS App Runner 서비스와 통합할 수 있습니다. 애플리케이션의 일부가 VPC 내에서 App Runner에 요청을 보낼 수 있습니다. 예를 들어 App Runner 서비스에 지속적으로 배포하는 AWS CodePipeline

데 사용할 수 있습니다. 애플리케이션의 보안을 개선하는 한 가지 방법은 VPC 엔드포인트를 통해 이러한 App Runner 요청 (및 다른 요청에 대한 요청 AWS 서비스) 을 보내는 것입니다.

VPC 엔드포인트를 사용하면 VPC를 에서 지원하는 지원 서비스 및 AWS 서비스 VPC 엔드포인트 서비스에 비공개로 연결할 수 있습니다. AWS PrivateLink인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 연결이 필요하지 않습니다. AWS Direct Connect

VPC의 리소스는 퍼블릭 IP 주소를 사용하여 App Runner 리소스와 상호작용하지 않습니다. VPC와 앱 러너 사이의 트래픽은 Amazon 네트워크를 벗어나지 않습니다. VPC 엔드포인트에 대한 자세한 내용은 가이드의 VPC 엔드포인트를 참조하십시오.AWS PrivateLink

> **Note**
>
> 기본적으로 App Runner 서비스의 웹 애플리케이션은 App Runner가 제공하고 구성하는 VPC에서 실행됩니다. 이 VPC는 퍼블릭 VPC입니다. 인터넷에 연결되어 있다는 뜻입니다. 애플리케이션을 사용자 지정 VPC와 연결할 수도 있습니다. 자세한 정보는 the section called “발신 트래픽”을 참조하세요.
> 서비스가 VPC에 연결되어 있는 경우에도 AWS API를 포함한 인터넷에 액세스하도록 서비스를 구성할 수 있습니다. VPC 아웃바운드 트래픽에 대한 퍼블릭 인터넷 액세스를 활성화하는 방법에 대한 지침은 을 참조하십시오. the section called “서브넷 선택 시 고려 사항 ”
> App Runner는 애플리케이션을 위한 VPC 엔드포인트 생성을 지원하지 않습니다.

앱 러너용 VPC 엔드포인트 설정

VPC에서 App Runner 서비스를 위한 인터페이스 VPC 엔드포인트를 만들려면 가이드의 인터페이스 엔드포인트 생성 절차를 따르십시오.AWS PrivateLink 서비스 이름에서 `com.amazonaws.region.apprunner`를 선택합니다.

VPC 네트워크 개인 정보 보호 고려 사항

> **Important**
>
> App Runner용 VPC 엔드포인트를 사용한다고 해서 VPC의 모든 트래픽이 인터넷을 차단할 수는 없습니다. VPC는 퍼블릭일 수 있습니다. 또한 솔루션의 일부에서는 VPC 엔드포인트를 사용하여 API를 호출하지 않을 수 있습니다. AWS 예를 AWS 서비스 들어 퍼블릭 엔드포인트를 사용하여 다른 서비스를 호출할 수 있습니다. VPC의 솔루션에 트래픽 프라이버시가 필요한 경우 이 섹션을 읽어보세요.

VPC의 네트워크 트래픽 프라이버시를 보장하려면 다음 사항을 고려하십시오.

- DNS 이름 활성화 — 애플리케이션의 일부가 여전히 `apprunner.region.amazonaws.com` 퍼블릭 엔드포인트를 사용하여 인터넷을 통해 App Runner에 요청을 보낼 수 있습니다. VPC가 인터넷 액세스를 지원하도록 구성된 경우 이러한 요청은 아무런 알림 없이 성공합니다. 엔드포인트를 생성할 때 DNS 이름 활성화가 활성화되도록 하여 이를 방지할 수 있습니다. 기본적으로 이 값은 true로 설정됩니다. 그러면 퍼블릭 서비스 엔드포인트를 인터페이스 VPC 엔드포인트에 매핑하는 DNS 항목이 VPC에 추가됩니다.
- 추가 서비스를 위한 VPC 엔드포인트 구성 — 솔루션이 다른 사람에게 요청을 보낼 수 있습니다. AWS 서비스예를 AWS CodePipeline 들어, 에 요청을 보낼 수 있습니다. AWS CodeBuild이러한 서비스에 대한 VPC 엔드포인트를 구성하고 이러한 엔드포인트에서 DNS 이름을 활성화합니다.
- 프라이빗 VPC 구성 - 가능하면 (솔루션에 인터넷 액세스가 전혀 필요하지 않은 경우) VPC를 비공개로 설정합니다. 즉, 인터넷에 연결되어 있지 않습니다. 이렇게 하면 누락된 VPC 엔드포인트로 인해 눈에 보이는 오류가 발생하므로 누락된 엔드포인트를 추가할 수 있습니다.

엔드포인트 정책을 사용하여 VPC 엔드포인트로 액세스 제어

VPC 엔드포인트 정책은 앱 러너에 지원되지 않습니다. 기본적으로 인터페이스 엔드포인트를 통해 App Runner에 대한 전체 액세스가 허용됩니다. 또는 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하여 인터페이스 엔드포인트를 통해 App Runner로 들어오는 트래픽을 제어할 수 있습니다.

인터페이스 엔드포인트와 통합

App Runner는 App Runner에 대한 비공개 연결을 제공하고 트래픽이 인터넷에 노출되지 않도록 지원합니다 AWS PrivateLink. 애플리케이션이 를 사용하여 App Runner에 요청을 보낼 수 있게 하려면 AWS PrivateLink인터페이스 엔드포인트라고 하는 VPC 엔드포인트 유형을 구성하십시오. 자세한 내용은 AWS PrivateLink 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하십시오.

앱 러너의 구성 및 취약성 분석

AWS 또한 고객은 높은 수준의 소프트웨어 구성 요소 보안 및 규정 준수를 달성할 책임을 공유합니다. 자세한 내용은 AWS 공동 책임 모델을 참조하십시오.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

앱 러너의 보안 베스트 프랙티스

AWS App Runner 자체 보안 정책을 개발하고 구현할 때 고려해야 할 몇 가지 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주십시오.

다른 App Runner 보안 주제에 대해서는 을 참조하십시오보안.

예방 보안 모범 사례

예방적 보안 통제는 사고가 발생하기 전에 사고를 예방하려고 시도합니다.

최소 권한 액세스 구현

App Runner는 IAM 사용자 및 액세스 역할에 대한 AWS Identity and Access Management (IAM) 관리형 정책을 제공합니다. 이러한 관리형 정책은 App Runner 서비스를 올바르게 운영하는 데 필요할 수 있는 모든 권한을 지정합니다.

애플리케이션에 우리의 관리형 정책의 모든 권한이 필요한 것은 아닙니다. 이를 사용자 지정하고 사용자와 App Runner 서비스가 작업을 수행하는 데 필요한 권한만 부여할 수 있습니다. 이는 다른 사용자 역할이 다른 권한 요구를 가질 수 있는 사용자 정책과 특히 관련이 있습니다. 최소 권한 액세스를 구현하는 것이 오류 또는 악의적인 의도로 인해 발생할 수 있는 보안 위험과 영향을 최소화할 수 있는 근본적인 방법입니다.

탐지 보안 모범 사례

탐지 보안 통제는 보안 위반이 발생한 후 이를 식별합니다. 잠재적인 보안 위협이나 사고를 탐지하는 데 도움이 됩니다.

모니터링 구현

모니터링은 App Runner 솔루션의 안정성, 보안, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. AWS 서비스를 모니터링하는 데 도움이 되는 여러 도구와 AWS 서비스를 제공합니다.

다음은 모니터링 대상 항목의 예입니다:

- App Runner용 Amazon CloudWatch 지표 — 주요 App Runner 지표와 애플리케이션의 사용자 지정 지표에 대한 경보를 설정합니다. 자세한 내용은 지표 () CloudWatch단원을 참조하세요.
- AWS CloudTrail 항목 — 또는 같이 가용성에 영향을 미칠 수 있는 작업을 추적합니다. `PauseService` `DeleteConnection` 자세한 내용은 API 작업 () CloudTrail 단원을 참조하십시오.

AWS 용어집

최신 AWS 용어는 참조의 AWS 용어집을 참조하십시오.AWS 용어집

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.