개발자 가이드

AWS App Runner

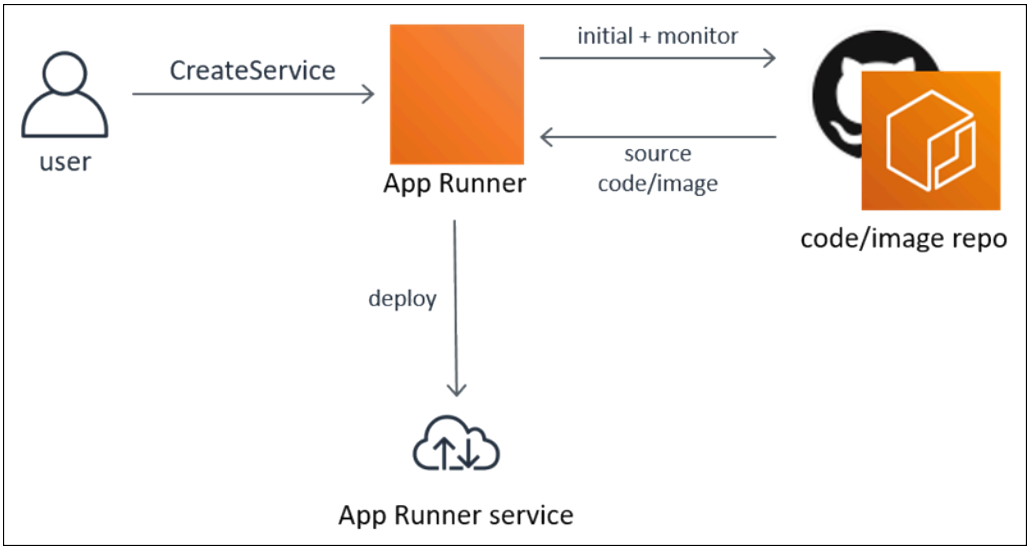
AWS App Runner: 개발자 가이드

Table of Contents

AWS App Runner이란 무엇인가요?

AWS App Runner는AWS서비스를 통해 소스 코드 또는 컨테이너 이미지에서 확장 가능하고 안전한 웹 응용 프로그램에 직접 배포할 수 있는 빠르고 간단하며 비용 효율적인 방법을AWS클라우드 새로운 기술을 배우거나, 사용할 컴퓨팅 서비스를 결정하거나, 프로비저닝 및 구성 방법을 알 필요가 없습니다.AWS있습니다.

앱 러너는 코드 또는 이미지 저장소에 직접 연결합니다. 완전 관리형 운영, 고성능, 확장성 및 보안을 갖춘 자동 통합 및 딜리버리 파이프라인을 제공합니다.



앱 러너는 누구를위한 것입니까?

운영 체제데이터 웨어하우스에서는 App Runner를 사용하여 새 버전의 코드 또는 이미지 리포지토리를 배포하는 프로세스를 간소화할 수 있습니다.

용운영 팀를 사용하면 커밋이 코드 리포지토리에 푸시되거나 새 컨테이너 이미지 버전이 이미지 리포지토리에 푸시될 때마다 App Runner가 자동 배포를 활성화합니다.

앱 러너 액세스

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

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

- AWS명령줄 인터페이스 (AWS CLI)— 다양한 명령어를 제공합니다.AWS서비스를 제공하며 VPC, macOS, Linux를 지원합니다. 자세한 내용은 AWS Command Line Interface 단원을 참조하세요.
- AWSSDK언어별 API를 제공하고, 서명 계산, 요청 재시도 처리 및 오류 처리와 같은 많은 연결 세부 정보를 관리합니다. 자세한 정보는 AWS SDK를 참조하십시오.

앱 러너 요금

응용 프로그램을 실행하는 비용 효율적인 방법을 제공합니다. 앱 러너 서비스에서 소비하는 리소스에 대해서만 요금을 지불합니다. 요청 트래픽이 느려지면 서비스가 더 적은 수의 계산 인스턴스로 축소됩니다. 확장성 설정 (프로비저닝된 인스턴스의 가장 낮거나 가장 높은 수, 인스턴스가 처리하는 최고 로드) 을 제어할 수 있습니다.

App Runner 자동 조정에 대한 자세한 내용은 단원을 참조하십시오.the section called “Auto Scaling”.

요금 정보는 AWS App Runner 요금을 참조하세요.

다음 단계

다음 항목에서 App Runner 를 시작하는 방법을 알아봅니다.

- 설정— App Runner를 사용하기 위한 필수 단계를 완료합니다.
- 시작하기— 앱 러너에 첫 번째 애플리케이션을 배포합니다.

앱 러너 설정

새 인 경우AWS고객의 경우 사용을 시작하기 전에 이 페이지에 나열된 설치 필수 구성 요소를 완료하십시오.AWS App Runner.

이러한 설치 절차의 경우AWS Identity and Access Management(IAM) 서비스를 사용할 수 있습니다. IAM에 대한 자세한 내용은 다음 참조 자료를 참조하십시오.

- AWS Identity and Access Management (IAM)
- IAM User Guide

AWS 계정 생성

당신은 가입 할 때AWS를 사용하면 모든 서비스에 액세스 할 수있는 계정 번호를 얻을 수 있습니다.AWS제안AWS App Runner.

이미 를 가지고 있는 경우AWS 계정에서 다음 사전 조건으로 건너뜁니다.

AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성하십시오.

AWS 계정에 가입하려면

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드를 사용하여 확인 코드를 입력하는 과정이 있습니다.

IAM 사용자 생성

에 액세스하려면AWS서비스를 사용하는 경우 자격 증명을 제공합니다. 이러한 자격 증명을 통해인증(당신이 누구인지) 그리고authorization(작업을 수행하는 데 필요한 사용 권한AWS리소스) 를 참조하십시오.

Amazon Web Services (를 처음 생성할 때)AWS) 계정을 사용하는 경우 단일 로그인 자격 증명으로 시작합니다. 이 자격 증명은 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한을 갖습니다. 이 자격 증명을 호출하려면AWSaccount루트 사용자. 로그인할 때 계정을 만드는 데 사용한 이메일 주소와 암호를 입력합니다.

> **Important**
>
> 일상적인 작업, 심지어 관리 작업의 경우에도 루트 사용자를 사용하지 마실 것을 강력히 권장합니다. 대신, IAM 사용자를 처음 생성할 때만 루트 사용자를 사용하는 모범 사례를 준수합니다. 그런 다음 루트 사용자 자격 증명을 안전하게 보관하고 몇 가지 계정 및 서비스 관리 작업을 수행할 때만 사용합니다. 루트 사용자로 로그인해야 하는 작업을 보려면루트 사용자 자격 증명을 필요로 하는 작업.

루트 사용자 및 IAM 사용자 자격 증명에 대한 자세한 내용은 단원을 참조하십시오.AWS 계정루트 사용자 자격 증명 및 IAM 사용자 자격 증명의AWS일반 참조.

관리자 사용자를 직접 생성하여 관리자 그룹에 추가하려면(콘솔)

1. 에 로그인합니다.IAM 콘솔를 선택하여 계정 소유자로루트 사용자를 입력하고AWS계정 이메일 주소입니다. 다음 페이지에서 암호를 입력합니다.

   > **Note**
   >
   > 사용을 위한 모범 사례를 준수하는 것이 좋습니다.**Administrator**루트 사용자 자격 증명을 준수하고 안전하게 보관해 두는 IAM 사용자입니다. 몇 가지 계정 및 서비스 관리 태스크를 수행하려면 반드시 루트 사용자로 로그인해야 합니다.

2. 탐색 창에서 사용자와 사용자 추가를 차례로 선택합니다.
3. 사용자 이름에 **Administrator**를 입력합니다.
4. AWS Management Console 액세스 옆의 확인란을 선택합니다. 그런 다음 Custom password(사용자 지정 암호)를 선택하고 텍스트 상자에 새 암호를 입력합니다.
5. (선택 사항) 기본적으로 AWS에서는 새 사용자가 처음 로그인할 때 새 암호를 생성해야 합니다. User must create a new password at next sign-in(사용자가 다음에 로그인할 때 새 암호를 생성해야 합니다) 옆에 있는 확인란의 선택을 취소하면 새 사용자가 로그인한 후 암호를 재설정할 수 있습니다.
6. [다음: 권한(Next: Permissions)]을 선택합니다.
7. 권한 설정 아래에서 그룹에 사용자 추가를 선택합니다.
8. 그룹 생성을 선택합니다.
9. 그룹 생성 대화 상자의 그룹 이름에 **Administrators**를 입력합니다.
10. 선택정책 필터링을 선택한 다음AWS관리 - 작업 기능를 클릭하여 테이블 내용을 필터링합니다.

11. 정책 목록에서 AdministratorAccess 확인란을 선택합니다. 그런 다음 Create group을 선택합니다.

> **Note**
>
> 결제에 대한 IAM 사용자 및 역할 액세스를 활성화해야AdministratorAccess에 액세스할 수 있는 권한AWS Billing and Cost Management콘솔에 로그인합니다. 이를 위해 결제 콘솔에 액세스를 위임하기 위한 자습서 1단계의 지침을 따르십시오.

12. 그룹 목록으로 돌아가 새로 만든 그룹 옆의 확인란을 선택합니다. 목록에서 그룹을 확인하기 위해 필요한 경우 Refresh(새로 고침)를 선택합니다.
13. [다음: 권한(Next: Tags)]를 선택합니다.
14. (선택 사항) 태그를 키-값 페어로 연결하여 메타데이터를 사용자에게 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 단원을 참조하십시오.IAM 엔터티 태그 지정의IAM 사용 설명서.
15. [다음: 권한(Next: 검토새 사용자에 추가될 그룹 멤버십의 목록을 확인합니다. 계속 진행할 준비가 되었으면 Create user를 선택합니다.

이와 동일한 절차에 따라 그룹이나 사용자를 추가 생성하여 사용자에게 AWS 계정 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 사용자 권한을 특정 것으로 제한하는 정책을 사용하는 방법을 알아보려면AWS리소스에 대한 자세한 내용은액세스 관리및정책 예제.

> **Important**
>
> 보호AWS 계정. 조직 외부의 누구와도 자격 증명을 보내거나 공유하지 마십시오. Amazon을 합법적으로 대표하는 사람이라면 누구도 자격 증명을 요구하지 않을 것입니다.

IAM 사용자를 생성한 후 자격 증명을 사용하여AWS Management Console. 자세한 내용은 단원을 참조하십시오.IAM 사용자가 로그인하는 방법AWS 계정의IAM 사용 설명서.

IAM 사용자에 대한 액세스 키 생성

액세스 키는 액세스 키 ID 및 보안 액세스 키로 이루어져 있는데, 이를 사용하여 AWS에 보내는 프로그래밍 방식의 요청에 서명할 수 있습니다. 액세스 키가 없는 경우에는 AWS Management Console에서 액세스 키를 생성할 수 있습니다. 가장 좋은 방법은 를 사용하지 않는 것입니다.AWS계정 루트 사용자 액세스 키가 필요하지 않은 작업입니다. 그 대신새 관리자 IAM 사용자를 생성하려면에 대한 액세스 키가 있습니다.

보안 액세스 키는 액세스 키를 생성하는 시점에만 보고 다운로드할 수 있습니다. 나중에 복구할 수 없습니다. 하지만 언제든지 새 액세스 키를 생성할 수 있습니다. 필요한 IAM 작업을 수행할 수 있는 권한도 있어야 합니다. 자세한 내용은 단원을 참조하십시오.IAM 리소스에 액세스하는 데 필요한 권한의IAM 사용 설명서.

IAM 사용자에 대한 액세스 키를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 액세스 키를 생성할 사용자의 이름을 선택한 다음 Security credentials(보안 자격 증명) 탭을 선택합니다.
4. 액세스 키 섹션에서 Create access key(액세스 키 생성)를 선택합니다.
5. 새 액세스 키 페어를 보려면 표시를 선택합니다. 이 대화 상자를 닫은 후에는 보안 액세스 키에 다시 액세스할 수 없습니다. 자격 증명은 다음()과 동일합니다.

   - 액세스 키 ID: 아키아이오스포드엔7 예시
   - 보안 액세스 키: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEY
6. 키 페어 파일을 다운로드하려면 [Download .csv file]을 선택합니다. 안전한 위치에 키를 저장합니다. 이 대화 상자를 닫은 후에는 보안 액세스 키에 다시 액세스할 수 없습니다.

   AWS 계정을 보호하기 위해 키를 기밀로 유지하고, 이메일로 전송하지 마십시오. AWS 또는 Amazon.com의 이름으로 문의가 온다 할지라도 조직 외부로 키를 공유하지 마십시오. Amazon을 합법적으로 대표하는 사람이라면 결코 보안 키를 요구하지 않을 것입니다.
7. `.csv` 파일을 다운로드한 후 닫기를 선택합니다. 액세스 키를 생성하면 키 페어가 기본적으로 활성화되므로 해당 페어를 즉시 사용할 수 있습니다.

관련 주제

- IAM이란 무엇입니까?의IAM 사용 설명서
- AWS보안 자격 증명inAWS일반 참조

다음 단계

선행 조건 단계를 완료했습니다. 앱 러너에 첫 번째 응용 프로그램을 배포하려면시작하기.

앱 러너 시작하기

AWS App Runner는AWS서비스에서 기존 컨테이너 이미지 또는 소스 코드를 실행 중인 웹 서비스로 직접 전환하는 빠르고 간단하며 비용 효율적인 방법을 제공합니다.AWS 클라우드.

이 튜토리얼은 사용하는 방법에 대해 설명합니다AWS App Runner를 사용하여 응용 프로그램을 앱 러너 서비스에 배포합니다. 소스 코드 및 배포, 서비스 빌드 및 서비스 런타임 구성을 안내합니다. 또한 코드 버전을 배포하고 구성을 변경하고 로그를 보는 방법도 보여 줍니다. 마지막으로 자습서에서는 자습서의 절차를 따르는 동안 만든 리소스를 정리하는 방법을 보여 줍니다.

주제

Prerequisites

자습서를 시작하기 전에 다음 작업을 수행해야 합니다.

1. 의 설정 단계를 완료합니다.설정.
2. 만들기GitHub계정이 아직 없다면 지금 만드십시오. GitHub 를 처음 사용한다면GitHub 시작하기의GitHub 문서.
3. GitHub 계정에 리포지토리를 만듭니다. 이 자습서에서는 리포지토리 이름을 사용합니다.`python-hello`. 저장소의 루트 디렉토리에 다음 예제에 지정된 이름과 컨텐트를 사용하여 파일을 만듭니다.

에 대한 파일**`python-hello`**예제 리포지토리

Example requirements.txt

```
Click==7.0
Flask==1.0.2
```

```
itsdangerous==1.1.0
Jinja2==2.10
MarkupSafe==1.1.1
Werkzeug==0.15.5
```

Example server.py

```
from flask import Flask
import os

PORT = 8080
name = os.environ['NAME']
if name == None or len(name) == 0:
  name = "world"
MESSAGE = "Hello, " + name + "!"
print("Message: '" + MESSAGE + "'")

app = Flask(__name__)


@app.route("/")
def root():
  print("Handling web request. Returning message.")
  result = MESSAGE.encode("utf-8")
  return result


if __name__ == "__main__":
  app.run(debug=True, host="0.0.0.0", port=PORT)
```

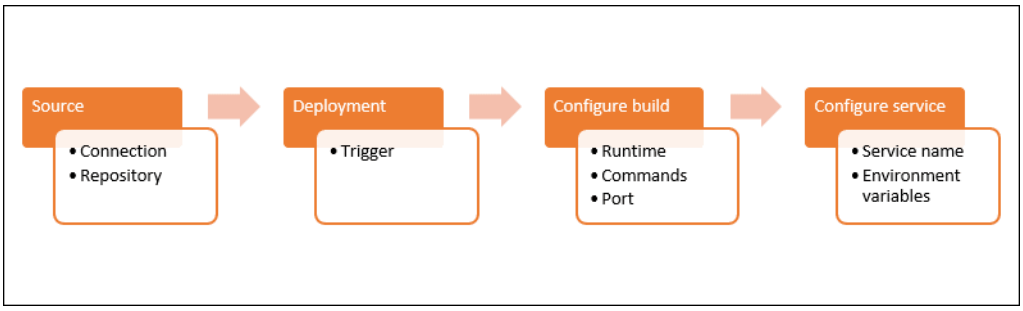
1단계: 앱 러너 서비스 생성

이 단계에서는 GitHub 에서 만든 예제 소스 코드 리포지토리를 기반으로 App Runner 서비스를 만듭니다.the section called “Prerequisites”. 이 예제는 간단한 파이썬 웹 사이트를 포함합니다. 다음은 서비스를 만들기 위해 수행하는 주요 단계입니다.

1. 소스 코드를 구성합니다.
2. 소스 배포를 구성합니다.
3. 애플리케이션 빌드를 구성합니다.
4. 서비스를 구성합니다.

5. 검토 및 확인.

다음 다이어그램에서는 App Runner 서비스를 만드는 단계를 간략하게 설명합니다.

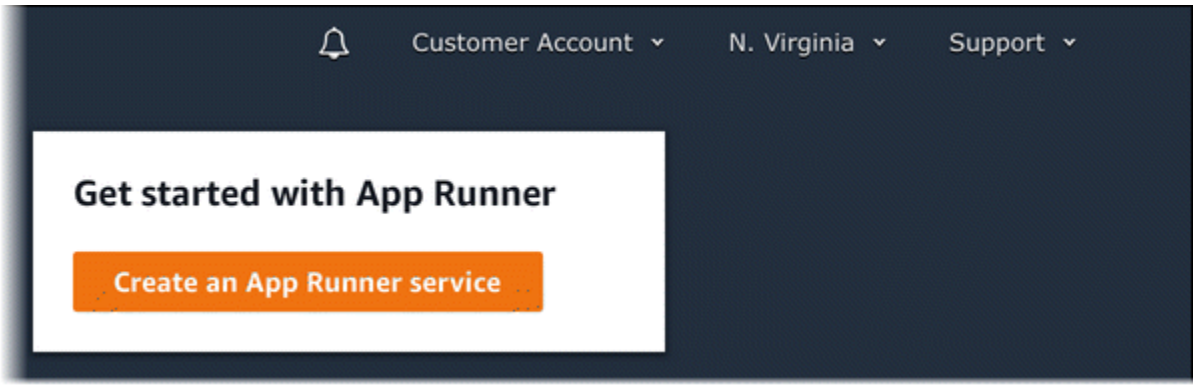


소스 코드 리포지토리를 기반으로 App Runner 서비스를 만들려면

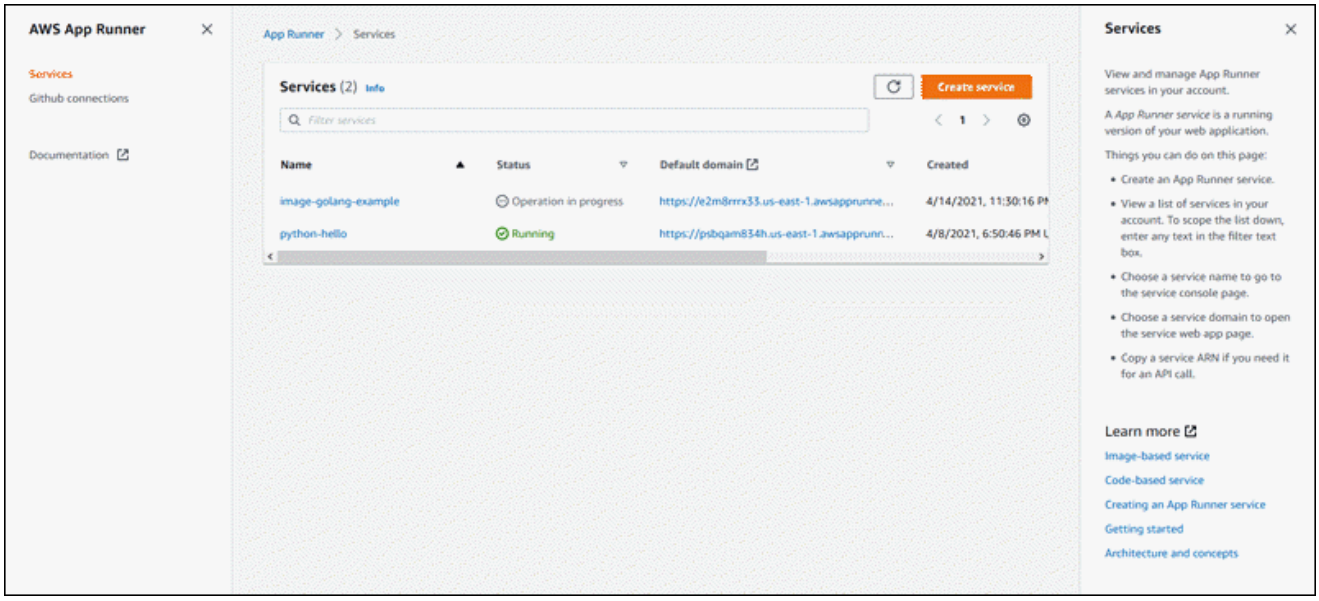
1. 소스 코드를 구성합니다.

   a. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택한 후AWS 리전.

   b. 경우AWS 계정에 아직 App Runner 서비스가 없으면 콘솔 홈 페이지가 표시됩니다. 선택앱 러너 서비스 생성.



   경우AWS 계정에 기존 서비스가 있는 경우서비스페이지가 표시되고 서비스 목록이 표시됩니다. [Create service]를 선택합니다.



c. 에서소스 및 배포페이지의소스섹션에서리포지토리 유형를 선택하고소스 코드 리포지토리.

d. 언더GitHub 에 Connect선택할새로 추가를 클릭한 다음 메시지가 표시되는 경우 GitHub 자격 증명을 제공합니다.

> **Note**
>
> 다음 단계를 설치하려면AWSGitHub 계정에 대한 GitHub 커넥터는 일회성 단계입니다. 연결을 재사용하여 이 계정의 리포지토리를 기반으로 여러 App Runner 서비스를 만들 수 있습니다. 기존 연결이 있는 경우 연결을 선택하고 저장소 선택으로 건너뜁니다.

e. 에서설치AWS용 GitHub 커넥터대화 상자에서 메시지가 표시되면 GitHub 계정 이름을 선택합니다.

f. 권한을 부여하라는 메시지가 표시되면AWSGitHub 용 커넥터에서AWS 연결 권한 부여.

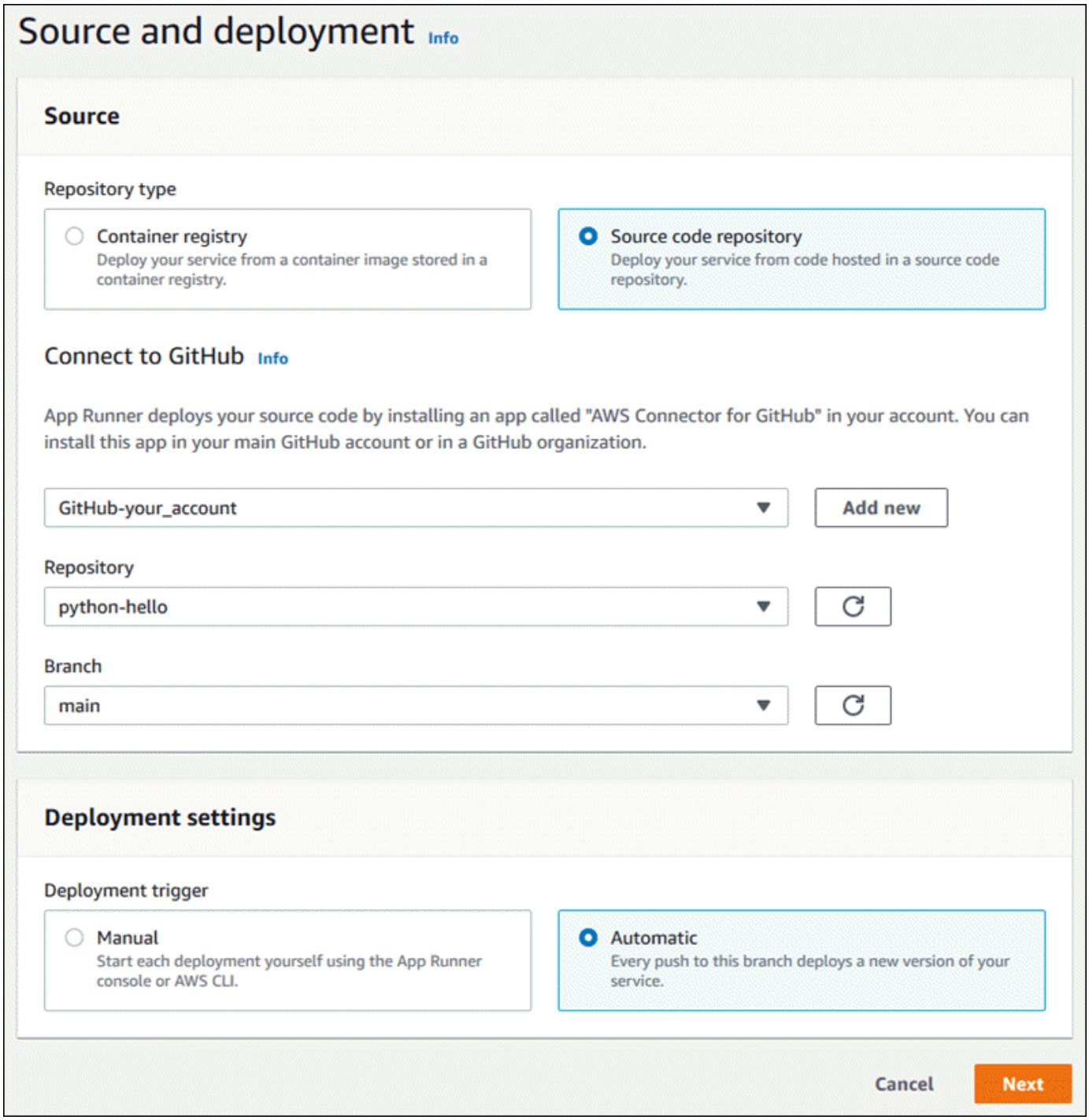
g. [Install]을 선택합니다.

계정 이름이 선택한GitHub 계정/조직. 이제 계정에서 리포지토리를 선택할 수 있습니다.

h. 용리포지토리에서 생성한 예제 리포지토리를 선택하고`python-hello`. 용브랜치에서 리포지토리의 기본 브랜치 이름을 선택합니다 (예:Main).

2. 배포를 구성합니다. 에서배포 설정섹션에서자동를 선택한 후다음.

> **Note**
>
> 자동 배포를 사용하면 저장소에 대한 새로운 커밋이 자동으로 새 버전의 서비스를 배포합니다.



3. 애플리케이션 빌드를 구성합니다.

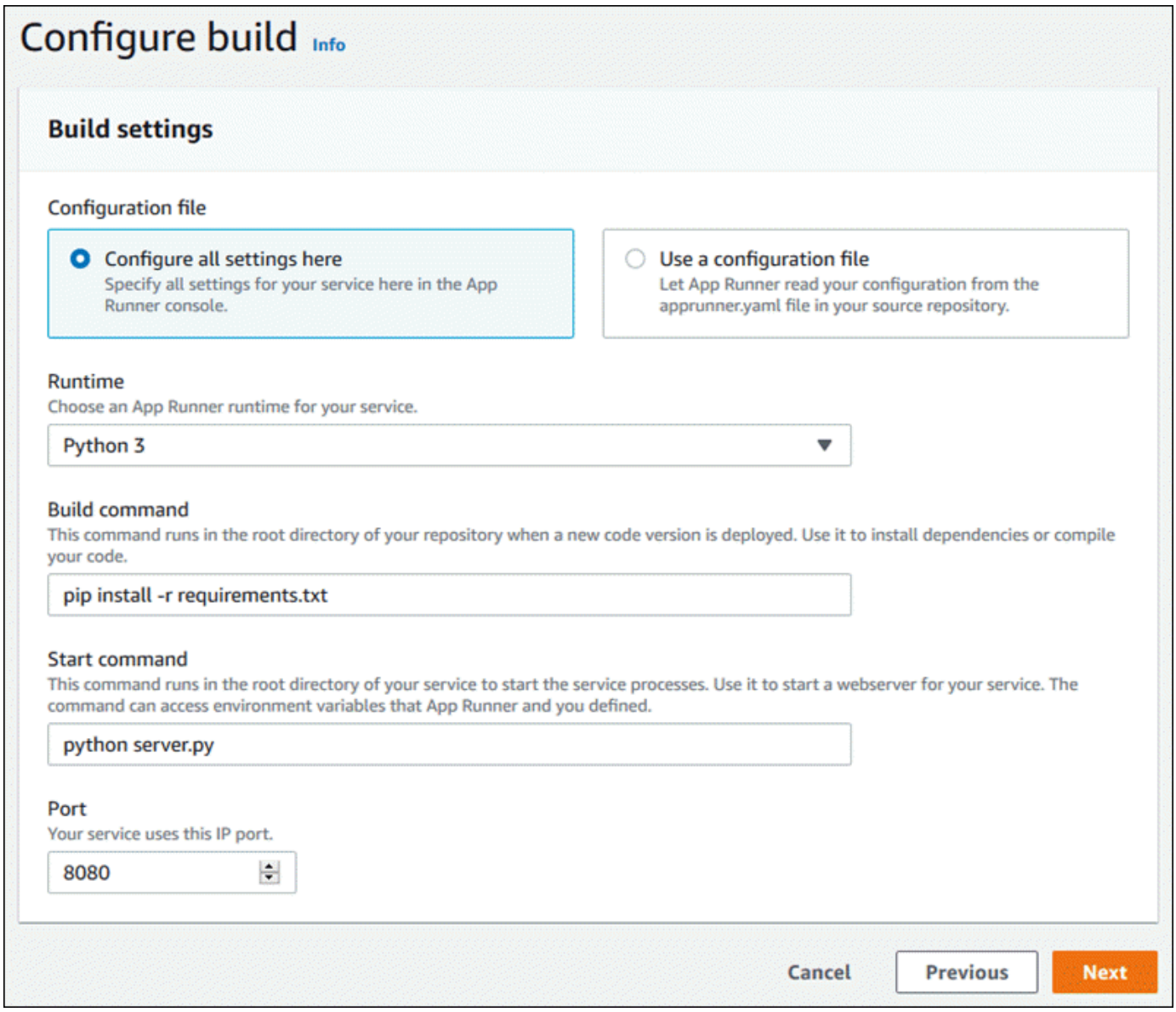
   a. 에서빌드 구성페이지에서구성 파일를 선택하고여기에서 모든 설정 구성.

   b. 다음 빌드 설정을 제공합니다.

      - 런타임— 선택을 선택합니다.Python 3.
      - 빌드 명령— 를 입력합니다.**`pip install -r requirements.txt`**.
      - 시작 명령— 를 입력합니다.**`python server.py`**.
      - 포트— 를 입력합니다.**`8080`**.

   c. [Next]를 선택합니다.

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



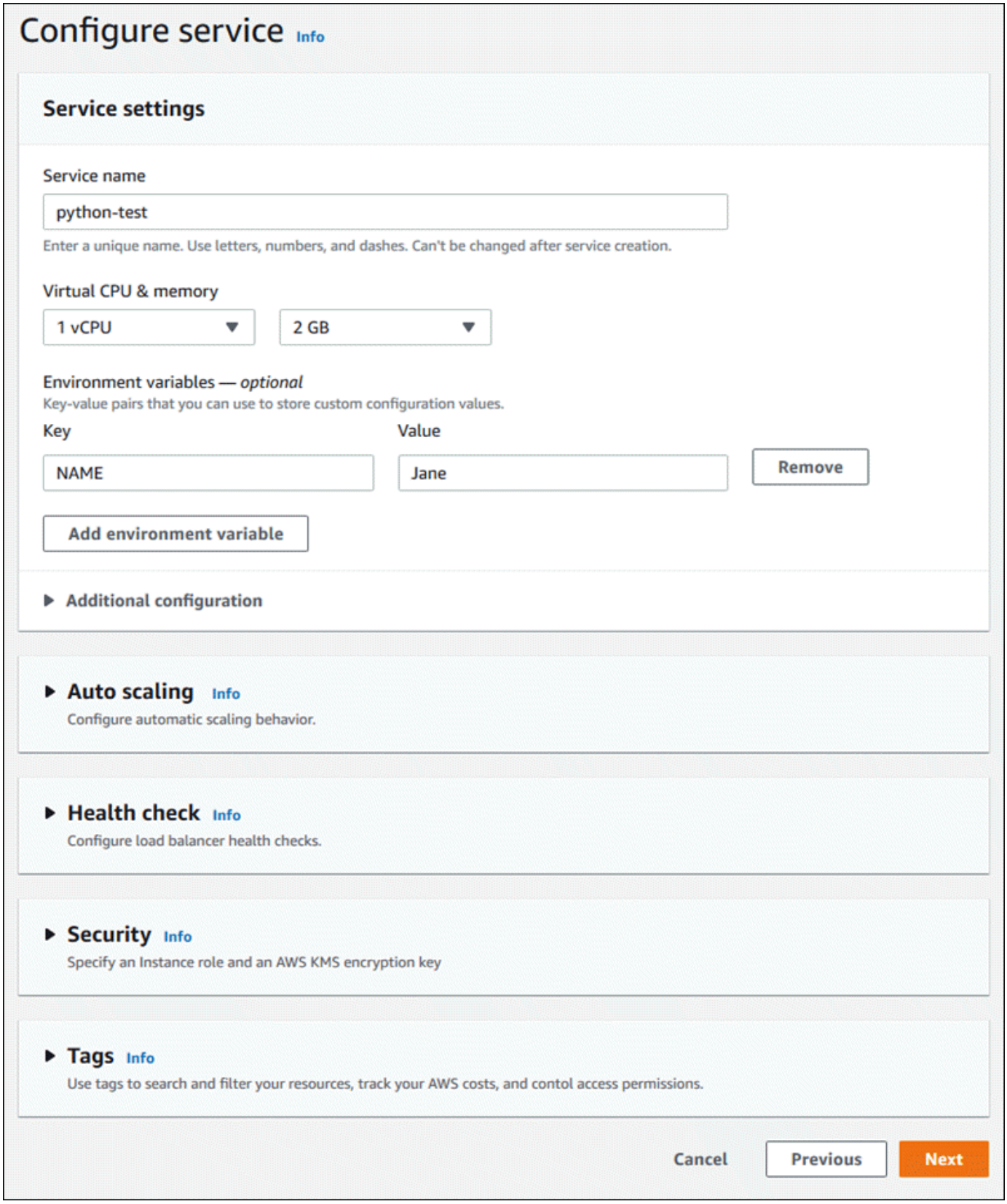
4. 서비스를 구성합니다.

   a. 에서서비스 구성페이지의서비스 설정섹션에서 서비스 이름을 입력합니다.

   b. 언더환경 변수에 단일 환경 변수를 추가하십시오. 용Key를 입력하고**NAME**, 그리고값에서 이름 (예: 이름) 을 입력합니다.

> **Note**
>
> 예제 응용 프로그램은 이 환경 변수에 설정한 이름을 읽고 해당 웹 페이지에 이름을 표시합니다.

c. [Next]를 선택합니다.


Configure service Info
Service settings
Service name
python-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.
Key
Value
NAME
Jane
Remove
Add environment variable
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

5. 에서검토 및 생성페이지에서 입력한 모든 세부 정보를 확인한 다음생성 및 배포.

   서비스가 성공적으로 생성되면 콘솔에 서비스 대시보드가 표시되고서비스 개요새 서비스의

6. 서비스가 실행 중인지 확인합니다.

   a. 서비스 대시보드 페이지에서 서비스가 될 때까지 기다립니다.상태확장하는 데[Running].

   b. 선택을 선택합니다.기본 도메인값 (서비스 웹 사이트에 대한 URL입니다).

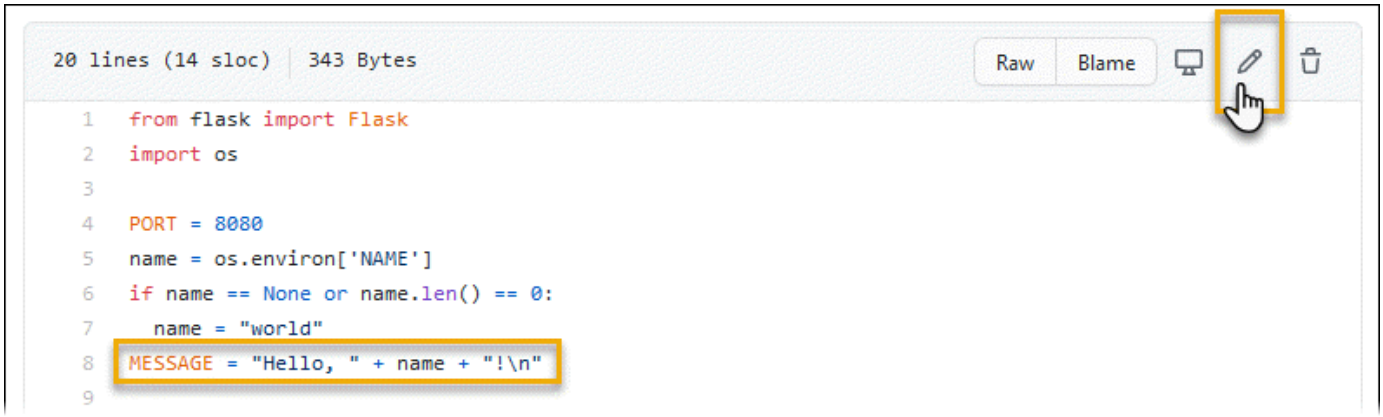
      웹 페이지에 다음 항목이 표시됩니다. Hello,**##**!

2단계: 서비스 코드 변경

이 단계에서는 소스 코드 리포지토리를 변경합니다. App Runner CI/CD 기능은 변경 사항을 자동으로 빌드하고 서비스에 배포합니다.

서비스 코드를 변경하려면

1. 예제 GitHub 리포지토리로 이동합니다.
2. 파일 이름을 선택합니다.`server.py`를 클릭하여 해당 파일로 이동합니다.
3. 선택파일 편집(연필 아이콘) 을 클릭합니다.
4. 변수에 할당 된 표현식에서`MESSAGE`에서 텍스트 변경`Hello`to`Good morning`.



5. 변경 사항 커밋을 선택합니다.

   새 커밋이 배포되기 시작합니다. 서비스 대시보드 페이지에서 서비스상태변경 사항작업 진행 중.

6. 배포가 끝날 때까지 기다립니다. 서비스 대시보드 페이지에서 서비스상태로 다시 변경해야합니다.[Running].

7. 배포가 성공적인지 확인합니다. 서비스의 웹 페이지가 표시되는 브라우저 탭을 새로 고칩니다.

   이제 페이지에 수정된 메시지가 표시됩니다. 좋은 아침입니다.**##**!

3단계: 구성 변경

이 단계에서는 변경 내용을 해당**NAME**환경 변수 값을 사용하여 서비스 구성 변경을 보여 줍니다.

서비스에 대한 로그를 보려면

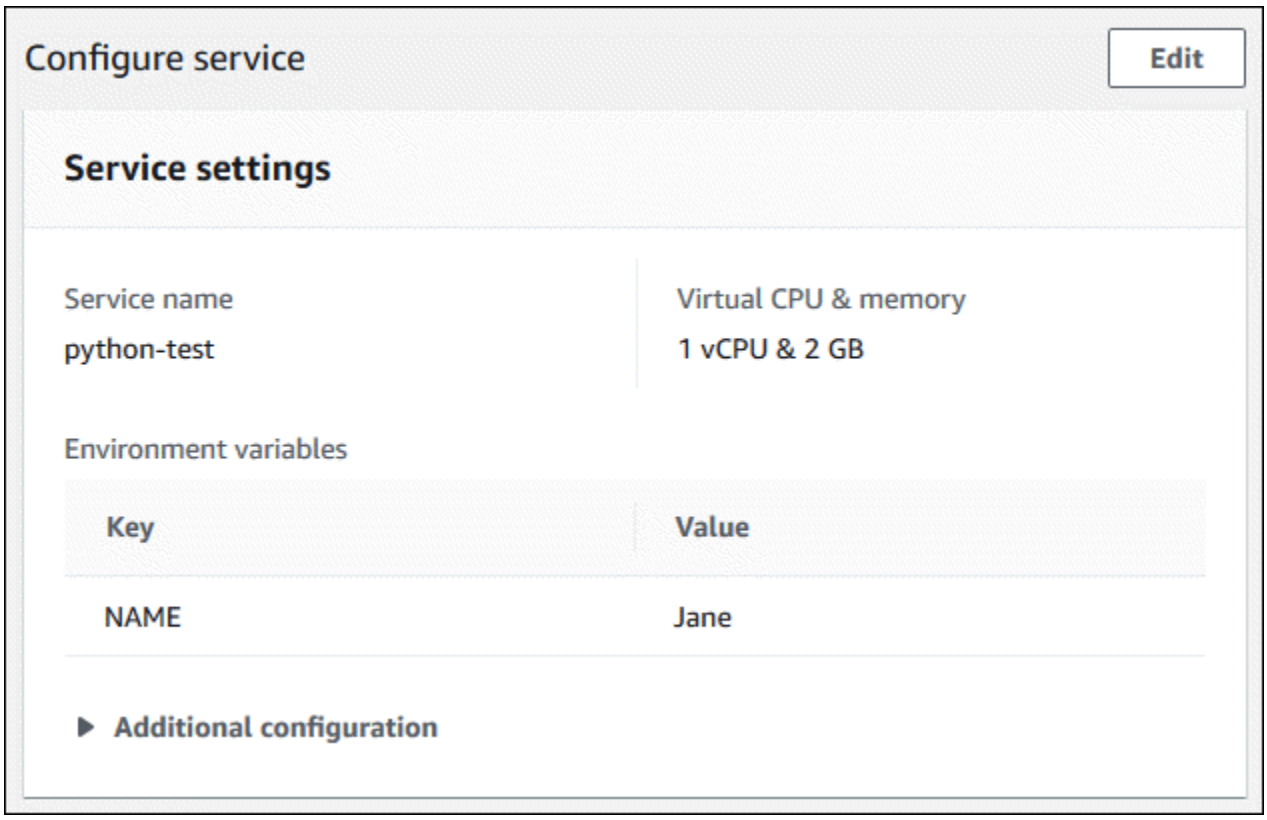
1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택한 후AWS 리전.
2. 탐색 창에서 [][]] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.

3. 서비스 대시보드 페이지에서구성탭을 선택합니다.

   콘솔에는 서비스 구성 설정이 여러 섹션에 표시됩니다.

4. 에서서비스 구성섹션에서Edit.



5. 키가있는 환경 변수의 경우**NAME**에서 값을 다른 이름으로 변경합니다.
6. [Apply changes]를 선택합니다.

   앱 러너가 업데이트 프로세스를 시작합니다. 서비스 대시보드 페이지에서 서비스상태변경 사항작업 진행 중.
7. 업데이트가 끝날 때까지 기다리십시오. 서비스 대시보드 페이지에서 서비스상태로 다시 변경해야 합니다.[Running].
8. 업데이트가 성공했는지 확인합니다. 서비스의 웹 페이지가 표시되는 브라우저 탭을 새로 고칩니다.

   이제 페이지에 수정된 이름이 표시됩니다. 좋은 아침입니다.**# ##**!

4단계: 서비스에 대한 로그 보기

이 단계에서는 앱 러너 콘솔을 사용하여 앱 러너 서비스에 대한 로그를 봅니다. 앱 러너는 로그를 Amazon CloudWatch Logs (CloudWatch 로그) 로 스트리밍하고 서비스 대시보드에 표시합니다. 앱 러너 로그에 대한 자세한 내용은the section called “로그 (CloudWatch Logs)”.

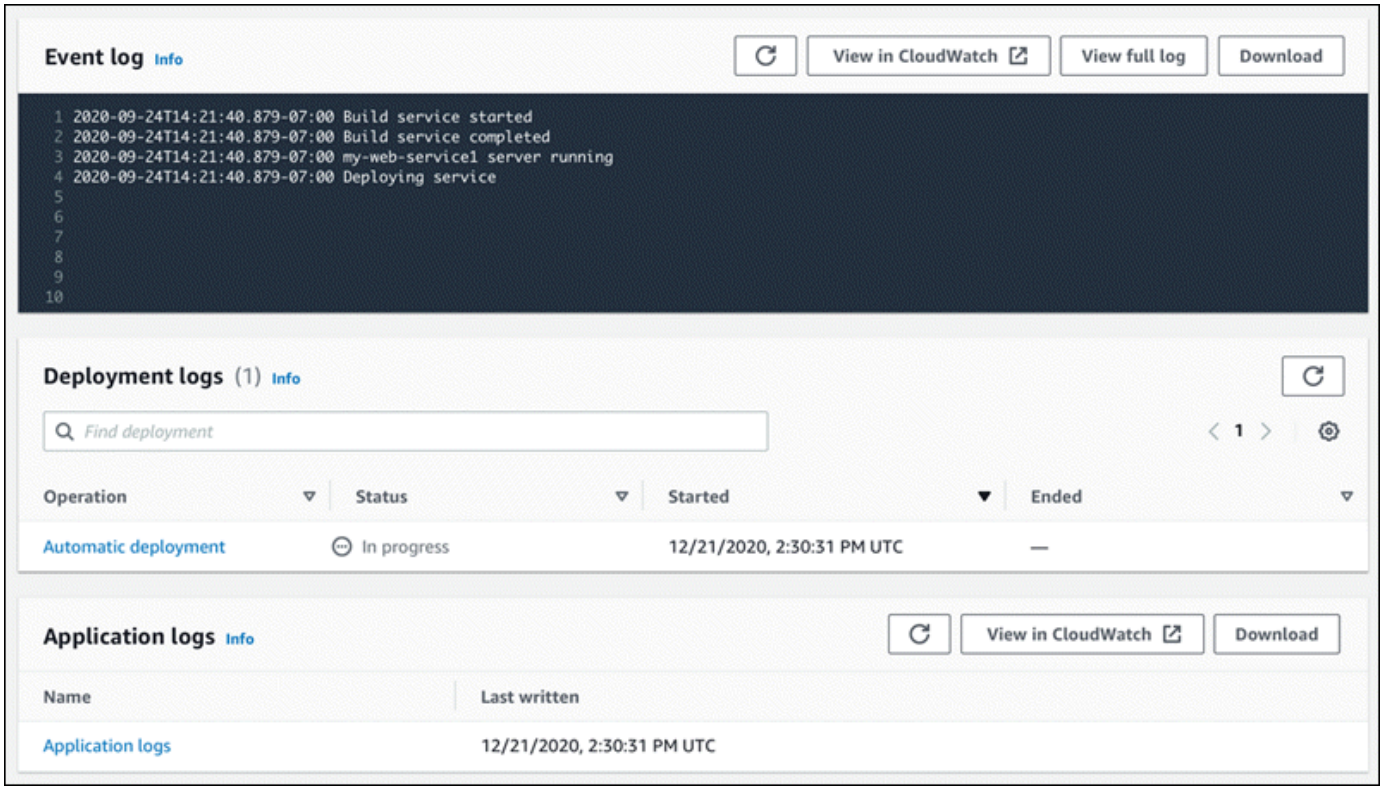
서비스에 대한 로그를 보려면

1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택한 후AWS 리전.
2. 탐색 창에서 [][]] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 서비스 대시보드 페이지에서로그탭을 선택합니다.

   콘솔에는 몇 가지 유형의 로그가 여러 섹션에 표시됩니다.

   - 이벤트 로그— 앱 러너 서비스 수명 주기의 활동입니다. 콘솔이 최신 이벤트를 표시합니다.
   - 배포 로그— 앱 러너 서비스에 대한 소스 리포지토리 배포입니다. 콘솔에는 각 배포에 대해 별도의 로그 스트림이 표시됩니다.
   - 애플리케이션 로그— App Runner 서비스에 배포된 웹 응용 프로그램의 출력입니다. 콘솔은 실행 중인 모든 인스턴스의 출력을 단일 로그 스트림으로 결합합니다.



4. 특정 배포를 찾으려면 검색어를 입력하여 배포 로그 목록의 범위를 좁힙니다. 테이블에 나타나는 값을 검색할 수 있습니다.

5. 로그의 내용을 보려면전체 로그 보기(이벤트 로그) 또는 로그 스트림 이름 (배포 및 응용 프로그램 로그) 을 입력합니다.
6. 선택다운로드를 클릭하여 로그를 다운로드합니다. 배치 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.
7. 선택CloudWatch 보기을 클릭하여 CloudWatch 콘솔을 열고 전체 기능을 사용하여 앱 러너 서비스 로그를 탐색할 수 있습니다. 배치 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.

> **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 변경을 수행하는 방법을 배웠습니다. 이 단계에서는 더 이상 필요 없는 리소스를 제거하기 위해 서비스를 삭제합니다.

서비스를 삭제하려면

1. 서비스 대시보드 페이지에서작업를 선택한 후서비스 삭제.
2. 확인 대화 상자에 요청된 텍스트를 입력한 다음삭제.

   Result: 콘솔이 탐색하는서비스페이지로 이동합니다. 방금 삭제한 서비스의 상태가DELETING. 잠시 후 목록에서 사라집니다.

이 자습서에서는 생성한 GitHub 연결도 삭제합니다. 자세한 내용은 the section called “연결” 단원을 참조하세요.

다음 단계

첫 번째 App Runner 서비스를 배포했으므로 다음 항목에서 자세히 알아보십시오.

- 아키텍처 및 개념— 아키텍처, 주요 개념 및AWS앱 러너와 관련된 리소스.
- 이미지 기반 서비스및코드 기반 서비스— App Runner가 배포할 수 있는 두 가지 유형의 응용 프로그램 원본입니다.

- 앱 러너를 위한 개발— App Runner로 배포할 응용 프로그램 코드를 개발하거나 마이그레이션할 때 알아야 할 사항
- 앱 Runner— 앱 러너 콘솔을 사용하여 서비스를 관리하고 모니터링합니다.
- 서비스 관리— 앱 러너 서비스의 수명 주기를 관리합니다.
- 로깅 및 모니터링— 지표를 보고, 로그를 읽고, 서비스 작업 호출을 추적하여 App Runner 서비스를 모니터링할 수 있습니다.
- 앱 러너 구성 파일— App Runner 서비스의 빌드 및 런타임 동작에 대한 옵션을 지정하는 구성 기반 방법입니다.
- 앱 러너 API— 앱 러너 API (애플리케이션 프로그래밍 인터페이스) 를 사용하여 앱 러너 리소스를 생성, 읽기, 업데이트 및 삭제합니다.
- 보안— 다른 방법으로AWS앱 러너 및 기타 서비스를 사용하는 동안 클라우드 보안을 보장할 수 있습니다.

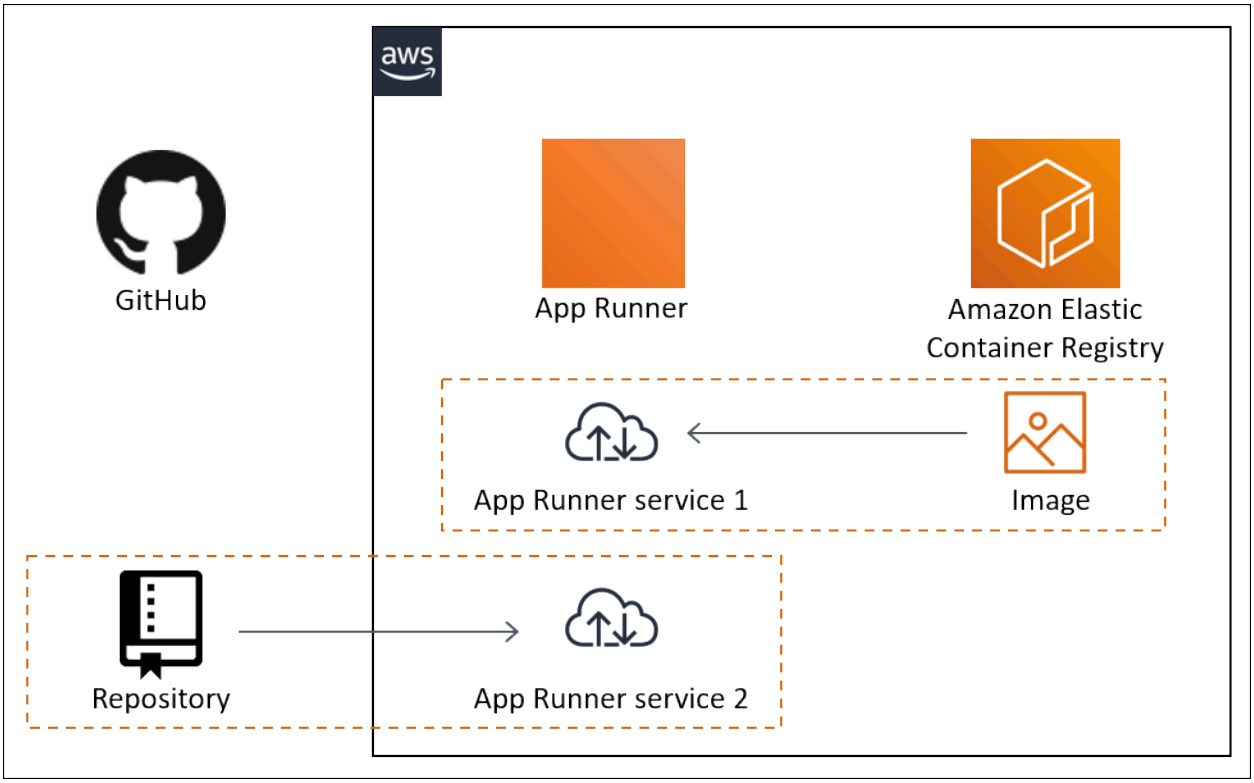
앱 러너 아키텍처 및 개념

AWS App Runner는 저장소에서 소스 코드 또는 소스 이미지를 가져 와서 실행중인 웹 서비스를AWS 클라우드. 일반적으로 하나의 App Runner 액션만 호출해야합니다.CreateService를 사용해 서비스를 생성합니다.

소스 이미지 리포지토리를 사용하면 App Runner가 웹 서비스를 실행하기 위해 배포 할 수있는 즉시 사용할 수있는 컨테이너 이미지를 제공합니다. 소스 코드 리포지토리를 사용하면 App Runner에서 관리하는 여러 런타임 환경 중 하나를 위해 설계된 웹 서비스를 빌드하고 실행하기 위한 코드 및 지침을 제공할 수 있습니다.

현재 App Runner는 소스 코드를GitHub리포지토리에서 소스 이미지를 검색하거나Amazon Elastic Container Registry (Amazon ECR)귀하의AWS 계정.

다음 다이어그램에는 App Runner 서비스 아키텍처가 개략적으로 나와 있습니다. 다이어그램에는 두 가지 예제 서비스가 있습니다. 하나는 GitHub 에서 소스 코드를 배포하고 다른 하나는 Amazon ECR에서 소스 이미지를 배포합니다.



App Runner

다음은 App Runner에서 실행되는 웹 서비스와 관련된 주요 개념입니다.

- App Runner 서비스—AWS리소스를 사용하여 소스 코드 리포지토리 또는 컨테이너 이미지를 기반으로 응용 프로그램을 배포하고 관리하는 데 사용합니다. 앱 러너 서비스는 응용 프로그램의 실행 버전입니다. 서비스 생성에 대한 자세한 내용은 단원을 참조하십시오.the section called “생성”.
- 소스 유형— App Runner 서비스를 배포하기 위해 제공하는 소스 리포지토리 유형입니다.소스 코드또는소스 이미지.
- 리포지토리 공급자— 애플리케이션 소스를 포함하는 리포지토리 서비스 (예:GitHub또는Amazon ECR).
- App Runner—AWS리소스를 사용하여 앱 러너가 저장소 공급자 계정 (예: GitHub 계정 또는 조직) 에 액세스할 수 있습니다. 연결에 대한 자세한 내용은 the section called “연결” 단원을 참조하십시오.
- 런타임— 소스 코드 리포지토리를 배포하기 위한 기본 이미지입니다. 앱 Runner의 다양한 제공관리되는 런타임다른 프로그래밍 환경. 자세한 내용은 코드 기반 서비스 단원을 참조하세요.
- 배포— 소스 리포지토리 버전 (코드 또는 이미지) 을 App Runner 서비스에 적용하는 작업입니다. 서비스에 대한 첫 번째 배포는 서비스 생성의 일부로 수행됩니다. 이후 배포는 다음 두 가지 방법 중 하나로 발생할 수 있습니다.
  - 자동 배포— CI/CD 기능 응용 프로그램의 각 버전을 리포지토리에 나타나는 대로 자동으로 빌드하고 배포하도록 App Runner 서비스를 구성할 수 있습니다. 소스 코드 저장소의 새 커밋이거나 소스 이미지 저장소의 새 이미지 버전이 될 수 있습니다.
  - 수동 배포— 명시적으로 시작하는 App Runner 서비스에 대한 배포입니다.
- 사용자 지정 도메인— 앱 러너 서비스와 연결하는 도메인입니다. 웹 응용 프로그램의 사용자는 기본 App Runner 하위 도메인 대신이 도메인을 사용하여 웹 서비스에 액세스할 수 있습니다. 자세한 내용은 the section called “사용자 지정 도메인 이름” 단원을 참조하세요.
- 유지 관리- App Runner가 앱 러너 서비스를 실행하는 인프라에서 가끔 수행하는 활동입니다. 유지보수가 진행 중이면 서비스 상태가 일시적으로`OPERATION_IN_PROGRESS`(작업 진행 중콘솔에서) 몇 분 동안 사용할 수 있습니다. 이 기간 동안 서비스에 대한 작업 (예: 배포, 구성 업데이트, 일시 중지/다시 시작 또는 삭제) 이 차단됩니다. 몇 분 후에 서비스 상태가`RUNNING`.

> **Note**
>
> 작업이 실패한다고해서 App Runner 서비스가 다운되었음을 의미하지는 않습니다. 응용 프로그램이 활성 상태이며 요청을 계속 처리합니다. 서비스가 다운타임을 경험하는 것은 거의 없습니다.

특히 App Runner는 서비스를 호스팅하는 기본 하드웨어에서 문제를 감지하면 서비스를 마이그레이션합니다. 서비스 중단 시간을 방지하기 위해 App Runner는 서비스를 새로운 인스턴스 집합에 배포

하고 트래픽을 해당 인스턴스로 이동합니다 (파란색 녹색 배포). 때때로 요금이 일시적으로 증가할 수 있습니다.

App Runner 리소스

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

다음 표에 이러한 리소스의 개략이 나와 있습니다.

리소스 이름	설명
Service	실행 중인 애플리케이션 버전을 나타냅니다. 이 가이드의 나머지 부분에서는 서비스 유형, 관리, 구성 및 모니터링에 대해 설명합니다. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:service/` *`service-name`* `[/`*`service-id`* `]`
Connection	타사 공급자와 함께 저장된 개인 리포지토리에 대한 액세스 권한을 App Runner 서비스에 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`* `]`
AutoScalingConfiguration	응용 프로그램의 자동 크기 조정을 제어하는 설정을 App Runner 서비스에 제공합니다. 여러 서비스에서 공유하기 위한 별도의 리소스로 존재합니다. 자동 크기 조정에 대한 자세한 내용은 단원을 참조하십시오.the section called “Auto Scaling”. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:autoscalingconfiguration/` *`config-name`* `[/`*`config-revision`* `[/`*`config-id`* `]]`

App Runner 리소스 할당량

AWS는 계정에 일부 할당량 (제한이라고도 함) 을 부과합니다.AWS각 리소스 사용량AWS 리전. 다음 표에는 App Runner 리소스와 관련된 할당량이 나열되어 있습니다. 할당량은AWS App Runner엔드포인트 및 할당량의AWS일반 참조.

리소스 할당량	설명	기본값	조정 가능?
Services	각 계정에서 생성할 수 있는 최대 서비스 수입니다.AWS 리전.	10	✓ 예
Connections	각 계정에서 생성할 수 있는 최대 연결 수입니다.AWS 리전. 여러 서비스에서 단일 연결을 공유할 수 있습니다.	10	✓ 예
Auto scaling configurations- 이름	계정에서 생성하는 Auto Scaling 구성에서 각 이름에 대해 가질 수 있는 고유 이름의 최대 수AWS 리전. 여러 서비스에서 Auto Scaling 구성을 사용할 수 있습니다.	10	✓ 예
Auto scaling configurations- 각 이름의 개정	각 계정에서 생성할 수 있는 최대 Auto Scaling 구성 수정 버전 수입니다.AWS 리전각 고유한 이름을 입력합니다. 여러 서비스에서 Auto Scaling 구성 버전을 사용할 수 있습니다.	10	× 아니요

대부분의 할당량을 조정할 수 있으며 할당량 증가를 요청할 수 있습니다. 자세한 내용은 단원을 참조하십시오.할당량 증가 요청Service Quotas 사용 설명서의 서비스 할당량 사용 설명서를 참조하십시오.

소스 이미지를 기반으로 한 앱 러너 서비스

다음을 수행할 수 있습니다.AWS App Runner를 사용하여 근본적으로 다른 두 가지 유형의 서비스 소스를 기반으로 서비스를 만들고 관리할 수 있습니다.소스 코드및소스 이미지. 소스 유형에 관계없이 App Runner는 서비스의 시작, 실행, 확장 및 부하 분산을 처리합니다. App Runner의 CI/CD 기능을 사용하여 소스 이미지 또는 코드의 변경 사항을 추적할 수 있습니다. App Runner가 변경 사항을 발견하면 소스 코드의 경우 자동으로 빌드되고 새 버전이 App Runner 서비스에 배포됩니다.

이 장에서는 소스 이미지를 기반으로 하는 서비스에 대해 설명합니다. 소스 코드 기반 서비스에 대한 자세한 내용은코드 기반 서비스.

A소스 이미지는 이미지 저장소에 저장된 공용 또는 개인 컨테이너 이미지입니다. App Runner를 이미지로 가리키고이 이미지를 기반으로 컨테이너를 실행하는 서비스를 시작합니다. 빌드 단계가 필요하지 않습니다. 대신 바로 배포할 수 있는 이미지를 제공합니다.

이미지 저장소 제공자

App Runner는 다음과 같은 이미지 저장소 공급자를 지원합니다.

- Amazon Elastic Container Registry (Amazon ECR)— 개인 이미지를AWS 계정.
- Amazon Elastic Container Registry Public에서 사용되는 컨테이너— 공개적으로 읽을 수 있는 이미지를 저장합니다.

Amazon ECR 에서 배포

Amazon ECR는 이미지를 리포지토리에 저장합니다. 개인 및 공용 저장소가 있습니다. 프라이빗 리포지토리에서 앱 러너 서비스로 이미지를 배포하려면 App Runner가 Amazon ECR에서 이미지를 읽을 수 있는 권한이 필요합니다. 앱 러너에게 해당 권한을 부여하려면 앱 러너에액세스 역할. 이것은AWS Identity and Access Management가 필요한 Amazon ECR 작업 권한이 있는 (IAM) 역할. App Runner 콘솔을 사용하여 서비스를 만들 때 계정에서 기존 역할을 선택할 수 있습니다. 또는 IAM 콘솔을 사용하여 새 사용자 지정 역할을 생성하거나, App Runner 콘솔에서 관리형 정책을 기반으로 역할을 생성하도록 선택할 수 있습니다.

앱 러너 API 또는AWS CLI를 선택하면 2단계 프로세스가 완료됩니다. 먼저 IAM 콘솔을 사용하여 액세스 역할을 생성합니다. App Runner가 제공하는 관리형 정책을 사용하거나 사용자 지정 권한을 입력할 수 있습니다. 그런 다음 사용 하 여 서비스를 만드는 동안 액세스 역할을 제공CreateServiceAPI 작업.

앱 러너 서비스 만들기에 대한 자세한 내용은the section called “생성”.

Amazon ECR 공개에서 배포

Amazon ECR는 공개적으로 읽을 수있는 이미지를 저장합니다. 다음은 앱 러너 서비스의 맥락에서 알아야 할 Amazon ECR과 Amazon ECR 퍼블릭 간의 주요 차이점입니다.

- Amazon ECR 공개 이미지는 공개적으로 읽을 수 있습니다. Amazon ECR 공개 이미지를 기반으로 서비스를 생성할 때 액세스 역할을 제공할 필요가 없습니다.
- 앱 러너는 Amazon ECR 공개 이미지에 대한 자동 배포를 지원하지 않습니다.

소스 코드를 기반으로 한 앱 러너 서비스

다음을 수행할 수 있습니다.AWS App Runner를 사용하여 근본적으로 다른 두 가지 유형의 서비스 소스를 기반으로 서비스를 만들고 관리할 수 있습니다.소스 코드및소스 이미지. 소스 유형에 관계없이 App Runner는 서비스의 시작, 실행, 확장 및 부하 분산을 처리합니다. App Runner의 CI/CD 기능을 사용하여 소스 이미지 또는 코드의 변경 사항을 추적할 수 있습니다. App Runner가 변경 사항을 발견하면 소스 코드의 경우 자동으로 빌드되고 새 버전이 App Runner 서비스에 배포됩니다.

이 장에서는 소스 코드를 기반으로 서비스에 대해 설명합니다. 원본 이미지를 기반으로 하는 서비스에 대한 자세한 내용은이미지 기반 서비스.

소스 코드는 App Runner가 빌드하고 배포하는 응용 프로그램 코드입니다. App Runner를 소스 코드 리포지토리로 가리키고 적절한실행 시간. App Runner는 런타임의 기본 이미지와 응용 프로그램 코드를 기반으로 하는 이미지를 빌드합니다. 그런 다음이 이미지를 기반으로 컨테이너를 실행하는 서비스를 시작합니다.

앱 러너는 편리한 언어별 제공관리되는 런타임. 이러한 각 런타임은 소스 코드에서 컨테이너 이미지를 작성하고 언어 런타임 종속성을 이미지에 추가합니다. 컨테이너 구성을 제공하고 Dockerfile과 같은 지침을 빌드 할 필요가 없습니다.

이 장의 하위 주제는 다양한런타임(일반 Dockerfile 런타임 및관리되는 런타임다른 프로그래밍 환경.

주제

- 소스 코드 저장소 공급자
- 앱 러너 관리 런타임
- 파이썬 관리 런타임 사용
- Node.js 관리 런타임 사용

소스 코드 저장소 공급자

App Runner는 소스 코드 저장소에서 소스 코드를 읽어 소스 코드를 배포합니다. App Runner는 하나의 소스 코드 저장소 공급자를 지원합니다. GitHub.

GitHub 에서 배포

응용 프로그램 실행기 서비스에 소스 코드를 배포하려면GitHub리포지토리에서 앱 러너는 GitHub 에 대한 연결을 설정합니다. 저장소가 비공개 인 경우 (즉, GitHub 에서 공개적으로 액세스 할 수없는 경

우) App Runner에 연결 세부 정보를 제공해야합니다. 앱 러너 콘솔을 사용하여서비스 생성서비스 생성 절차의 일부로 연결 세부 정보를 제공합니다.

앱 러너 API 또는AWS CLI인 경우 연결은 별도의 리소스입니다. 먼저 를 사용하여 연결을 만듭니다.CreateConnectionAPI 작업 그런 다음 서비스를 생성하는 동안 연결의 ARNCreateServiceAPI 작업

앱 실행기 서비스 생성에 대한 자세한 내용은 단원을 참조하십시오.the section called “생성”. 앱 실행기 연결에 대한 자세한 내용은 단원을 참조하십시오.the section called “연결”.

앱 러너 관리 런타임

앱 러너는 다양한 프로그래밍 환경에 대한 관리 런타임을 제공합니다. 각 관리되는 런타임을 사용하면 특정 프로그래밍 언어 또는 런타임 환경을 기반으로 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. 관리되는 런타임을 사용하는 경우 App Runner는 관리되는 런타임 이미지로 시작합니다. 이 이미지는Amazon Linux 도커 이미지이며 언어 런타임 패키지와 일부 도구 및 널리 사용되는 종속성 패키지가 포함되어 있습니다. App Runner는이 관리되는 런타임 이미지를 기본 이미지로 사용하고 응용 프로그램 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

앱 러너 서비스에 대한 런타임을 지정할 때서비스 생성앱 러너 콘솔 또는CreateServiceAPI. 소스 코드의 일부로 런타임을 지정할 수도 있습니다. 사용`runtime`키워드입니다.앱 러너 구성 파일코드 저장소에 포함 할 수 있습니다. 관리되는 런타임의 명명 규칙은*<language-name> <major-version>*.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 응용 프로그램에 관리되는 런타임의 특정 버전이 필요한 경우`runtime-version`키워드입니다.앱 러너 구성 파일. 부 버전을 다음과 같이 지정하십시오.*<major>*.*<minor>*를 사용하여 주 버전과 부 버전을 잠급니다 (App Runner는 패치 버전 만 업데이트합니다). 특정 패치 수준을*<major>*.*<minor>*.*<patch>*를 사용하여 특정 런타임 버전에서 서비스를 잠글 수 있습니다 (App Runner는 런타임을 업데이트하지 않습니다).

파이썬 관리 런타임 사용

AWS App Runner는 Python 관리 런타임을 제공합니다. Python 런타임에서는 Python 기반 웹 응용 프로그램으로 컨테이너를 간단하게 빌드하고 실행할 수 있습니다. 파이썬 런타임을 사용하면 앱 러너는 관리되는 파이썬 런타임 이미지로 시작합니다. 이 이미지는Amazon Linux 도커 이미지이며 Python 런타임 패키지와 일부 도구 및 인기있는 종속성 패키지가 포함되어 있습니다. App Runner는이 관리되는 런타임 이미지를 기본 이미지로 사용하고 응용 프로그램 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

앱 러너 서비스에 대한 런타임을 지정할 때서비스 생성앱 러너 콘솔 또는CreateServiceAPI. 소스 코드의 일부로 런타임을 지정할 수도 있습니다. 사용`runtime`키워드로앱 러너 구성 파일코드 저장소에 포함 할 수 있습니다. 관리되는 런타임의 명명 규칙은*`<language-name> <major-version>`*.

유효한 Python 런타임 이름에 대해서는the section called “릴리스 정보”.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 응용 프로그램에 관리되는 런타임의 특정 버전이 필요한 경우`runtime-version`키워드입니다.앱 러너 구성 파일. 부 버전을 다음과 같이 지정하십시오.*`<major>`*.*`<minor>`*를 사용하여 주 버전과 부 버전을 잠급니다 (App Runner는 패치 버전 만 업데이트합니다). 특정 패치 수준을*`<major>`*.*`<minor>`*.*`<patch>`*를 사용하여 특정 런타임 버전에서 서비스를 잠글 수 있습니다 (App Runner는 런타임을 업데이트하지 않습니다).

주제

- Python 런타임 구성
- Python 런타임 예제
- Python 런타임 릴리스 정보

Python 런타임 구성

관리되는 런타임을 선택하는 경우 최소한 빌드 및 실행 명령도 구성해야 합니다. 당신은 동안 그들을 구성creating또는업데이트앱 러너 서비스. 이를 수행하는 데는 몇 가지 방법이 있습니다.

- 앱 러너 콘솔 사용— 속성 매니저에서 명령을 지정합니다.빌드 구성섹션은 생성 프로세스 또는 구성 탭에서 확인할 수 있습니다.
- 앱 러너 API 사용를 호출합니다.CreateService또는UpdateService. 사용 하 여 명령을 지정`BuildCommand`및`StartCommand`의 멤버코드구성값데이터 형식입니다.
- 사용구성 파일— 최대 세 개의 빌드 단계에서 하나 이상의 빌드 명령과 응용 프로그램을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일을 제공하는 것은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때 App Runner가 만드는 동안 직접 구성 설정을 가져오는지 구성 파일에서 가져오는지 지정합니다.

Python 런타임 예제

다음 예제에서는 Python 서비스를 빌드하고 실행하기 위한 App Runner 구성 파일을 보여 줍니다. 마지막 예는 Python 런타임 서비스에 배포할 수 있는 완전한 Python 응용 프로그램의 소스 코드입니다.

Python 구성 파일

이 예제에서는 Python 관리 런타임에서 사용할 수 있는 최소 구성 파일을 보여 줍니다. App Runner가 최소한의 구성 파일로 만드는 가정에 대해서는the section called “구성 파일 예시”.

Example apprunner.yaml

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

Python 구성 파일

이 예제는 Python 관리 런타임에서 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"
```

Python 응용 프로그램 소스를 끝내십시오.

이 예에서는 Python 런타임 서비스에 배포할 수 있는 전체 Python 응용 프로그램의 소스 코드를 보여 줍니다.

Example requirements.txt

```
Click==7.0
Flask==1.0.2
itsdangerous==1.1.0
Jinja2==2.10
MarkupSafe==1.1.1
Werkzeug==0.15.5
```

Example server.py

```
from flask import Flask
import os

PORT = 8080
name = os.environ['NAME']
if name == None or len(name) == 0:
  name = "world"
MESSAGE = "Hello, " + name + "!"
print("Message: '" + MESSAGE + "'")

app = Flask(__name__)


@app.route("/")
def root():
  print("Handling web request. Returning message.")
  result = MESSAGE.encode("utf-8")
  return result
```

```
if __name__ == "__main__":
  app.run(debug=True, host="0.0.0.0", port=PORT)
```

Example apprunner.yaml

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

Python 런타임 릴리스 정보

이 항목에서는 App Runner가 지원하는 Python 런타임 버전에 대한 전체 세부 정보를 나열합니다.

Python 3

세부 정보	설명
실행 시간 이름	Python3
마이너 버전	3.7, 3.8
포함된 패키지	파이썬, 핍, 셋업 도구, 휠, 버추얼

Node.js 관리 런타임 사용

AWS App Runner는 Node.js 관리 런타임을 제공합니다. 런타임에서는 Node.js 기반 웹 응용 프로그램을 사용하여 컨테이너를 쉽게 빌드하고 실행할 수 있습니다. Node.js 런타임을 사용할 때 앱 러너는 관리되는 Node.js 런타임 이미지로 시작합니다. 이 이미지는Amazon Linux 도커 이미지Node.js 런타임 패키지와 일부 도구가 포함되어 있습니다. App Runner는이 관리되는 런타임 이미지를 기본 이미지로 사용하고 응용 프로그램 코드를 추가하여 Docker 이미지를 빌드합니다. 그런 다음이 이미지를 배포하여 컨테이너에서 웹 서비스를 실행합니다.

앱 러너 서비스에 대한 런타임을 지정할 때서비스를 생성하려면앱 러너 콘솔 또는CreateServiceAPI를 참조하십시오. 소스 코드의 일부로 런타임을 지정할 수도 있습니다. 사용`runtime`키워드로앱 러

너 구성 파일코드 저장소에 포함 할 수 있습니다. 관리되는 런타임의 명명 규칙은*<language-name> <major-version>*.

유효한 Node.js 런타임 이름을 보려면the section called “릴리스 정보”.

App Runner는 모든 배포 또는 서비스 업데이트에서 서비스의 런타임을 최신 버전으로 업데이트합니다. 응용 프로그램에 관리되는 런타임의 특정 버전이 필요한 경우`runtime-version`키워드의앱 러너 구성 파일. 부 버전을 다음과 같이 지정하십시오.*<major>*.*<minor>*를 사용하여 주 버전과 부 버전을 잠급니다 (App Runner는 패치 버전 만 업데이트합니다). 특정 패치 수준을*<major>*.*<minor>*.*<patch>*를 사용하여 특정 런타임 버전에서 서비스를 잠글 수 있습니다 (App Runner는 런타임을 업데이트하지 않습니다).

주제

- Node.js 런타임 구성
- Node.js 런타임 예제
- Node.js 런타임 릴리스 정보

Node.js 런타임 구성

관리되는 런타임을 선택하는 경우 최소한 빌드 및 실행 명령도 구성해야 합니다. 당신은 동안 그들을 구성creating또는업데이트앱 러너 서비스. 다음과 같은 몇 가지 방법이 있습니다.

- 앱 러너 콘솔 사용— 속성 매니저에서 명령을 지정합니다.빌드 구성섹션은 생성 프로세스 또는 구성 탭에서 확인할 수 있습니다.
- 앱 러너 API 사용— 전화CreateService또는UpdateService. 사용 하 여 명령을 지정`BuildCommand`및`StartCommand`의 멤버코드구성값데이터 형식입니다.
- 사용구성 파일— 최대 3개의 빌드 단계에서 하나 이상의 빌드 명령과 애플리케이션을 시작하는 데 사용되는 단일 실행 명령을 지정합니다. 추가 선택적 구성 설정이 있습니다.

구성 파일을 제공하는 것은 선택 사항입니다. 콘솔 또는 API를 사용하여 App Runner 서비스를 만들 때 App Runner가 만드는 동안 직접 구성 설정을 가져오는지 구성 파일에서 가져오는지 지정합니다.

특히 Node.js 런타임을 사용하면 JSON 파일을 사용하여 빌드 및 런타임을 구성 할 수도 있습니다.`package.json`를 해당 소스 리포지토리의 루트에 추가합니다. 이 파일을 사용하여 Node.js 엔진 버전, 종속성 패키지 및 다양한 명령 (명령줄 응용 프로그램) 을 구성할 수 있습니다. npm 또는 yarn과 같은 패키지 관리자는이 파일을 명령에 대한 입력으로 해석합니다.

예:

- npm install에 의해 정의 된 패키지를 설치dependencies및devDependencies노드package.json.
- npm start또는npm run start에 의해 정의 된 명령을 실행scripts/start노드package.json.

다음은 예제 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": "12.18.4"
  },
  "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 내용package.json에 대한 자세한 내용은package.json 가이드Node.js 웹 사이트에서

Tips

- 만약 당신의package.json파일은start명령으로 사용할 수 있습니다.run명령을 다음 예제와 같이 App Runner 구성 파일에 저장합니다.

Example

package.json

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

apprunner.yaml

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

- 실행할 때npm install개발 환경에서 npm은 파일을 만듭니다.`package-lock.json`. 이 파일에는 방금 설치 한 패키지 버전의 스냅 샷이 들어 있습니다. 그런 다음 npm이 종속성을 설치하면 이러한 정확한 버전을 사용합니다. 마찬가지로 원사는`yarn.json`. 응용 프로그램이 개발 및 테스트한 종속성 버전과 함께 설치되도록 소스 코드 저장소에 이러한 파일을 커밋합니다.
- 앱 러너 구성 파일을 사용하여 Node.js 버전 및 시작 명령을 구성할 수도 있습니다. 이렇게 하면 이러한 정의는`package.json`. 사이의 충돌`node`버전`package.json`및`runtime-version`값을 응용 프로그램 러너 구성 파일에 실패 하려면 응용 프로그램 러너 빌드 단계를 발생 합니다.

Node.js 런타임 예제

다음 예에서는 Node.js 서비스를 빌드하고 실행하기 위한 앱 러너 구성 파일을 보여 줍니다.

최소 Node.js 구성 파일

이 예에서는 Node.js 관리 런타임에 사용할 수 있는 최소 구성 파일을 보여 줍니다. App Runner가 최소한의 구성 파일로 만드는 가정에 대해서는the section called “구성 파일 예시”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs12
```

```
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

Node.js 확장 구성 파일

이 예에서는 Node.js 관리 런타임에서 모든 구성 키를 사용하는 방법을 보여 줍니다.

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: 12.18.4
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Node.js 응용 프로그램

이 예제는 Grunt로 개발 된 Node.js 응용 프로그램을 구성하는 방법을 보여줍니다. 그런트는 명령 줄 JavaScript 작업 러너입니다. 반복적인 작업을 실행하고 프로세스 자동화를 관리하여 인적 오류를 줄입니다. 그런트와 그런트 플러그인은 NPM을 사용하여 설치 및 관리됩니다. 당신은 포함하여 그런트를 구성`Gruntfile.js`파일을 해당 소스 리포지토리의 루트에 저장합니다.

Example package.json

```
{
  "scripts": {
    "build": "grunt uglify",
    "start": "node app.js"
  },
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  },
  "dependencies": {
    "express": "^4.15.2"
  },
}
```

Example Gruntfile.js

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

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

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

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

};
```

Example apprunner.yaml

```
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: 12.18.4
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Node.js 런타임 릴리스 정보

이 항목에서는 App Runner가 지원하는 Node.js 런타임 버전에 대한 전체 세부 정보를 나열합니다.

Node.js 12

세부 정보	설명
실행 시간 이름	노드 12
마이너 버전	최신
포함된 패키지	노드 (npm 포함), 원사

앱 러너 용 응용 프로그램 코드 개발

이 장에서는 배포용 응용 프로그램 코드를 개발하거나 마이그레이션할 때 고려해야 할 런타임 정보 및 개발 지침에 대해 설명합니다.AWS App Runner.

런타임 정보

컨테이너 이미지를 제공하든 App Runner가 자동으로 빌드하든 관계없이 App Runner는 컨테이너 인스턴스에서 응용 프로그램 코드를 실행합니다. 다음은 컨테이너 인스턴스 런타임 환경의 몇 가지 주요 측면입니다.

- 프레임워크 지원— App Runner는 웹 애플리케이션을 구현하는 모든 이미지를 지원합니다. 선택하는 프로그래밍 언어와 사용하는 웹 응용 프로그램 서버 또는 프레임 워크 (사용하는 경우) 에는 무관합니다. 사용자의 편의를 위해 응용 프로그램 빌드 프로세스 및 추상적 이미지 생성을 간소화하기 위해 언어별 관리 런타임을 제공합니다.
- 웹 요청— 컨테이너 인스턴스는 기본적으로 포트 8080에서 HTTP 요청을 수신해야 합니다. 서비스 구성에 대한 자세한 내용은 단원을 참조하십시오.the section called “Configuration”. HTTPS 보안 트래픽 처리를 구현하지 않아도 됩니다. 앱 러너는 들어오는 HTTPS 트래픽이 필요하며 요청을 컨테이너 인스턴스로 전달하기 전에 HTTPS를 종료합니다.
- 상태 비저장 앱— App Runner는 단일 수신 웹 요청을 처리하는 기간 이후의 상태 지속성을 보장하지 않습니다.
- 스토리지- App Runner는 컨테이너 인스턴스의 파일 시스템을휘발성 저장. 파일이 일시적입니다. 예를 들어 앱 러너 서비스를 일시 중지했다가 다시 시작해도 그 외 상태가 유지되지 않습니다. 보다 일반적으로 파일은 응용 프로그램의 상태 비 저장 특성의 일부로 단일 요청 처리 이상으로 유지되지 않을 수 있습니다. 그러나 저장된 파일은 수명 기간 동안 App Runner 서비스의 저장소 할당의 일부를 차지합니다.

> **Note**
>
> 임시 저장소 파일은 요청 간에 유지되지 않을 수도 있지만가끔지속될 수 있습니다. 이 기능은 특정 상황에서 유용 할 수 있습니다. 예를 들어 요청을 처리할 때 향후 요청에 필요할 경우 응용 프로그램에서 다운로드하는 파일을 캐시할 수 있습니다. 이렇게하면 향후 요청 처리 속도가 빨라질 수 있지만 속도 향상을 보장 할 수는 없습니다. 코드는 이전 요청에서 다운로드한 파일이 여전히 존재한다고 가정해서는 안됩니다.

높은 처리량, 낮은 대기 시간 인 메모리 데이터 저장소를 사용하여 보장된 캐싱의 경우Amazon ElastiCache.

- 환경 변수— 기본적으로 앱 러너는PORT환경 변수를 사용할 수 있습니다. 포트 정보를 사용하여 변수 값을 구성하고 사용자 지정 환경 변수 및 값을 추가할 수 있습니다. 서비스 구성에 대한 자세한 내용은 단원을 참조하십시오.the section called “Configuration”.
- 인스턴스 역할- 응용 프로그램 코드에서AWS서비스 API 또는AWSSDK를 사용하려면 인스턴스 역할을 생성하려면AWS Identity and Access Management(IAM). 그런 다음 앱을 만들 때 앱 러너 서비스에 연결하십시오. 모두 포함AWS인스턴스 역할에서 코드에 필요한 서비스 작업 권한입니다. 자세한 내용은 the section called “인스턴스 역할” 단원을 참조하세요.

코드 개발 지침

App Runner 웹 응용 프로그램용 코드를 개발할 때 다음 지침을 고려하십시오.

- 상태 비저장 코드 설계— App Runner 서비스에 배포하는 웹 애플리케이션을 상태 비저장으로 디자인합니다. 코드는 들어오는 단일 웹 요청을 처리하는 기간 이후에도 상태가 지속되지 않는다고 가정해야합니다.
- 임시 파일 삭제— 파일을 만들면 파일 시스템에 저장되며 서비스의 스토리지 할당의 일부를 차지합니다. 저장소 부족 오류를 방지하려면 임시 파일을 장기간 보관하지 마십시오. 파일 캐싱 결정을 내릴 때 스토리지 크기와 요청 처리 속도를 조정합니다.

앱 러너 콘솔 사용

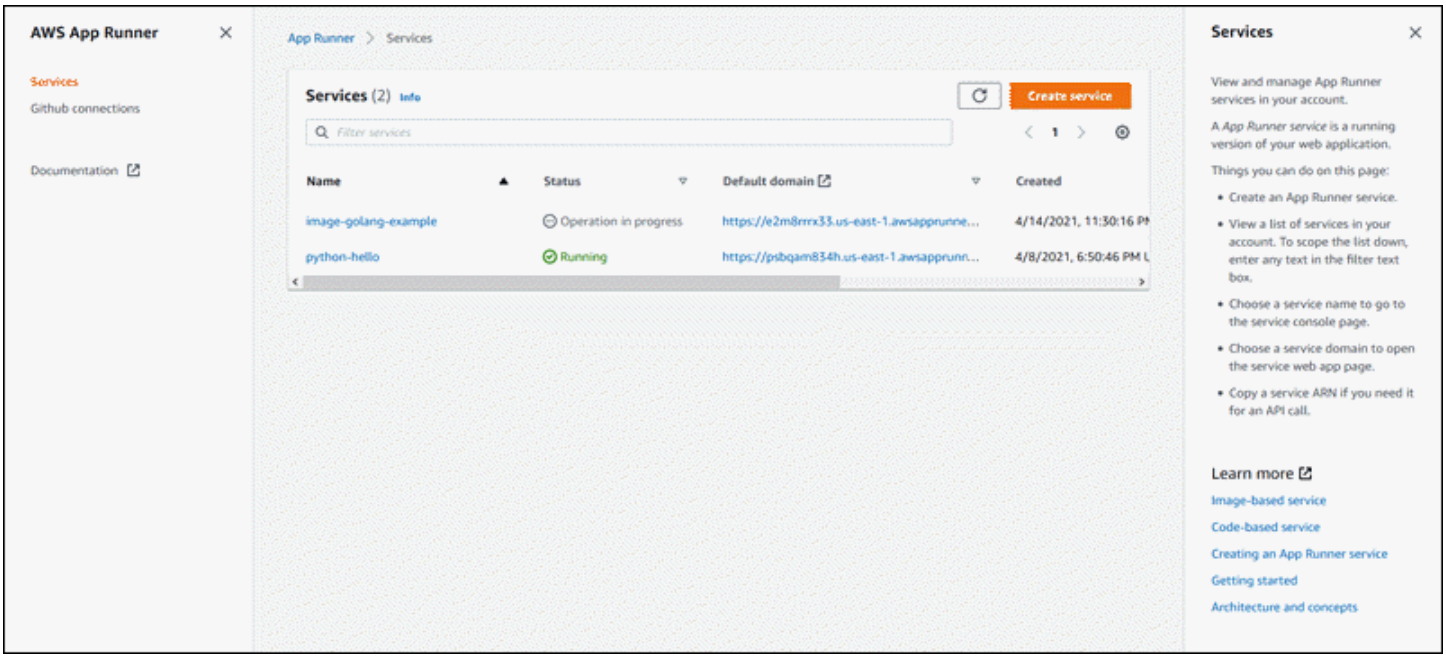
사용AWS App Runner콘솔을 사용하여 App Runner 서비스 및 연결 등의 관련 리소스를 생성, 관리 및 모니터링할 수 있습니다. 기존 서비스를 보고, 새 서비스를 만들고, 서비스를 구성할 수 있습니다. App Runner 서비스의 상태를 보고 로그를 보고, 활동을 모니터링하고, 메트릭을 추적할 수 있습니다. 서비스의 웹 사이트 또는 소스 저장소로 이동할 수도 있습니다.

다음 섹션에서는 콘솔의 레이아웃과 기능에 대해 설명하고 관련 정보를 안내합니다.

전체 콘솔 레이아웃

앱 러너 콘솔에는 세 가지 영역이 있습니다. 왼쪽에서 오른쪽으로:

- 탐색 창— 축소하거나 확장할 수 있는 측면 창입니다. 이 도구를 사용하여 사용하려는 최상위 콘솔 페이지를 선택합니다.
- 컨텐츠 창— 콘솔 페이지의 기본 부분입니다. 이 도구를 사용하여 정보를 보고 작업을 수행할 수 있습니다.
- 도움말 창— 자세한 내용을 볼 수 있는 측면 창입니다. 해당 페이지를 확장하면 현재 사용 중인 페이지에 대한 도움말을 볼 수 있습니다. 또는 원하는Info링크를 클릭하여 상황에 맞는 도움말을 볼 수 있습니다.



서비스 페이지

이서비스페이지에는 계정의 앱 러너 서비스가 나열됩니다. 필터 텍스트 상자를 사용하여 목록의 범위를 좁힐 수 있습니다.

에 도착하려면Services페이지를 방문하십시오

1. 열기앱 Runner에 있는Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스.

여기에서 할 수있는 일:

- 앱 Runner 서비스를 만듭니다. 자세한 내용은 the section called “생성” 단원을 참조하세요.
- 서비스 대시보드 콘솔 페이지로 이동하려면 서비스 이름을 선택합니다.
- 서비스 도메인을 선택하여 서비스 웹 앱 페이지를 엽니다.

서비스 대시보드 페이지

앱 러너 서비스에 대한 정보를 보고 서비스 dashbaord 페이지에서 관리할 수 있습니다. 페이지 상단에서 서비스 이름을 볼 수 있습니다.

서비스 대시보드로 이동하려면서비스페이지 (이전 섹션 참조) 를 클릭한 다음 App Runner 서비스를 선택합니다.

이서비스 개요섹션은 App Runner 서비스 및 응용 프로그램에 대한 기본 세부 정보를 제공합니다. 여기에서 할 수있는 일:

- 상태, 상태 및 ARN 과 같은 서비스 세부 정보를 봅니다.
- 로 이동합니다.기본 도메인- 서비스에서 실행되는 웹 응용 프로그램에 대해 App Runner가 제공하는 도메인입니다. 이 하위 도메인은`awsapprunner.com`응용 프로그램 러너가 소유 한 도메인.
- 서비스에 배포된 소스 리포지토리로 이동합니다.
- 서비스에 대한 소스 저장소 배포를 시작합니다.
- 서비스를 일시 중지, 다시 시작 및 삭제합니다.

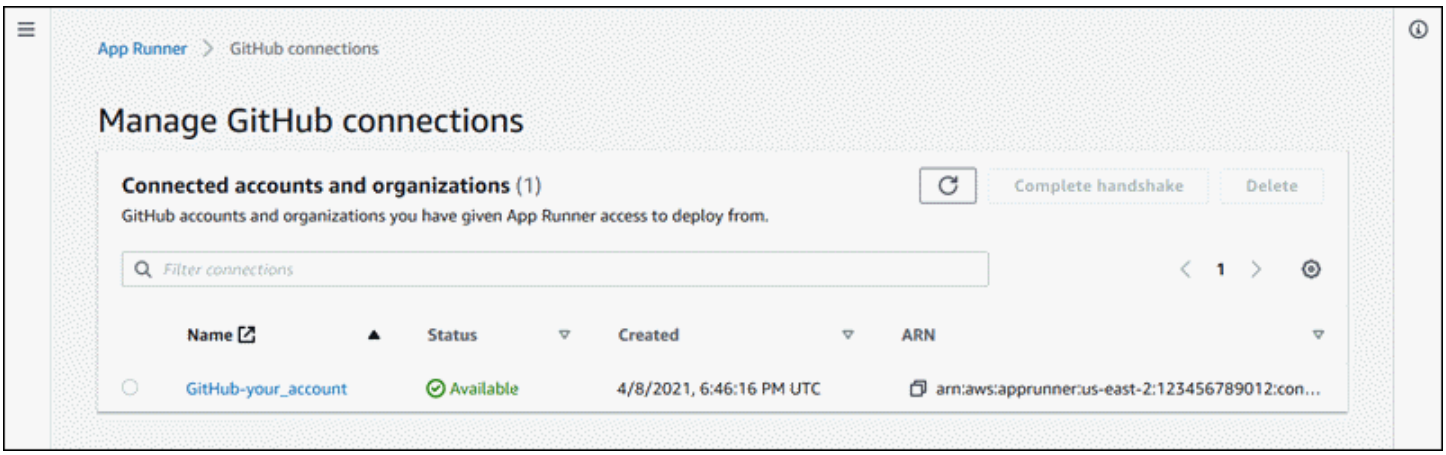
서비스 개요 아래의 탭은 서비스용입니다.management및모니터링.

GitHub 연결 페이지

이GitHub 연결페이지에는 계정의 GitHub 에 대한 앱 러너 연결이 나열됩니다. 필터 텍스트 상자를 사용하여 목록의 범위를 좁힐 수 있습니다. 연결에 대한 자세한 내용은 the section called “연결” 단원을 참조하십시오.

에 도착하려면GitHub 연결페이지를 방문하십시오

1. 열기앱 Runner에 있는Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.GitHub 연결.



여기에서 할 수있는 일:

- 계정의 GitHub 연결 목록을 봅니다. 목록의 범위를 좁히려면 필터 텍스트 상자에 텍스트를 입력합니다.
- 연결 이름을 선택하여 관련 GitHub 계정 또는 조직으로 이동합니다.
- 서비스를 만드는 과정에서 방금 설정한 연결에 대한 핸드셰이크를 완료하거나 연결을 삭제하려면 연결을 선택합니다.

앱 러너 서비스 수명 주기 관리

이 장에서는 수명 주기를 관리하는 방법에 대해 설명합니다.AWS App Runner서비스를 제공합니다. 이 장에서는 서비스를 생성, 구성 및 삭제하는 방법, 서비스에 새 응용 프로그램 버전을 배포하는 방법 및 연결을 관리하는 방법에 대해 알아봅니다. 서비스를 일시 중지했다가 다시 시작하여 웹 서비스의 가용성을 제어하는 방법도 알아봅니다.

주제

- 앱 러너 서비스 만들기
- 새 애플리케이션 버전 배포
- 앱 실행기 서비스 구성
- 앱 러너 연결 관리
- 앱 러너 자동 크기 조정 관리
- App Runner 서비스의 사용자 지정 도메인 이름 관리
- 앱 러너 서비스 일시 중지 및 다시 시작
- 앱 러너 서비스 삭제

앱 러너 서비스 만들기

AWS App Runner는 컨테이너 이미지 또는 소스 코드 리포지토리에서 자동으로 확장되는 실행 중인 웹 서비스로 이동하는 프로세스를 자동화합니다. App Runner를 소스 이미지 또는 코드로 가리키고 필요한 설정 수가 적은 경우에만 지정합니다. App Runner는 필요한 경우 응용 프로그램을 빌드하고, 계산 리소스를 프로비저닝하고, 응용 프로그램에서 실행할 응용 프로그램을 배포합니다.

서비스를 생성할 때 앱 러너는service리소스로 이동합니다. 경우에 따라, 를 제공해야 할 수 있습니다.ODBC리소스로 이동합니다. App Runner 콘솔을 사용하는 경우 콘솔은 암시적으로 연결 리소스를 만듭니다. 앱 러너 리소스 유형에 대한 자세한 내용은the section called “App Runner 리소스”. 이러한 리소스 유형에는 각AWS 리전. 자세한 내용은 the section called “App Runner 리소스 할당량” 단원을 참조하세요.

소스 유형 및 공급자에 따라 서비스를 만드는 절차에는 미묘한 차이가 있습니다. 이 항목에서는 상황에 가장 적합한 소스 유형을 따를 수 있도록 서로 다른 소스 유형을 만드는 절차를 완전히 분리하여 보여 줍니다. 코드에 대한 예제가 포함된 기본 시작 절차는 단원을 참조하십시오.시작하기.

Prerequisites

App Runner 서비스를 생성하기 전에 다음 작업을 완료해야 합니다.

- 의 설치 단계를 수행합니다.설정.
- 애플리케이션 소스를 준비하십시오. 코드 리포지토리 중 하나를 사용할 수 있습니다.GitHub또는 컨테이너 이미지Elastic Container Registry (Amazon ECR)를 사용하여 앱 러너 서비스를 만들 수 있습니다.

서비스 생성

이 섹션에서는 두 가지 App Runner 서비스 유형 (소스 코드 기반 및 컨테이너 이미지 기반) 에 대한 생성 프로세스를 안내합니다.

GitHub 코드 저장소에서 서비스 만들기

다음 단원에서는 소스가 코드 리포지토리인 경우 App Runner 서비스를 생성하는 방법을 보여줍니다.GitHub. GitHub 를 사용할 때 앱 러너는 GitHub 조직 또는 계정에 연결해야합니다. 따라서 이 연결을 설정하는 데 도움이 필요합니다. 앱 러너 연결에 대한 자세한 내용은 단원을 참조하십시오.the section called “연결”.

서비스를 만들면 App Runner가 애플리케이션 코드와 종속성이 포함된 Docker 이미지를 빌드합니다. 그런 다음이 이미지의 컨테이너 인스턴스를 실행하는 서비스를 시작합니다.

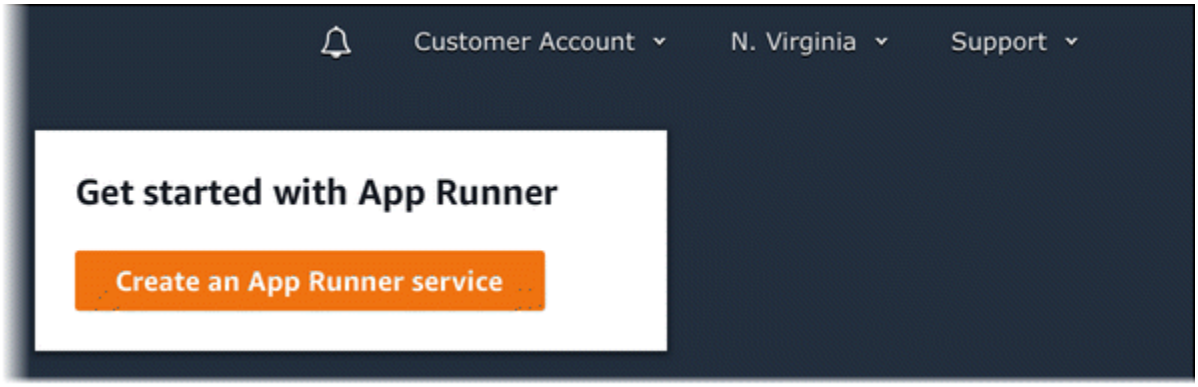
주제

- 앱 러너 콘솔을 사용하여 코드에서 서비스 만들기
- 앱 러너 API를 사용하여 코드에서 서비스 만들기 또는AWS CLI

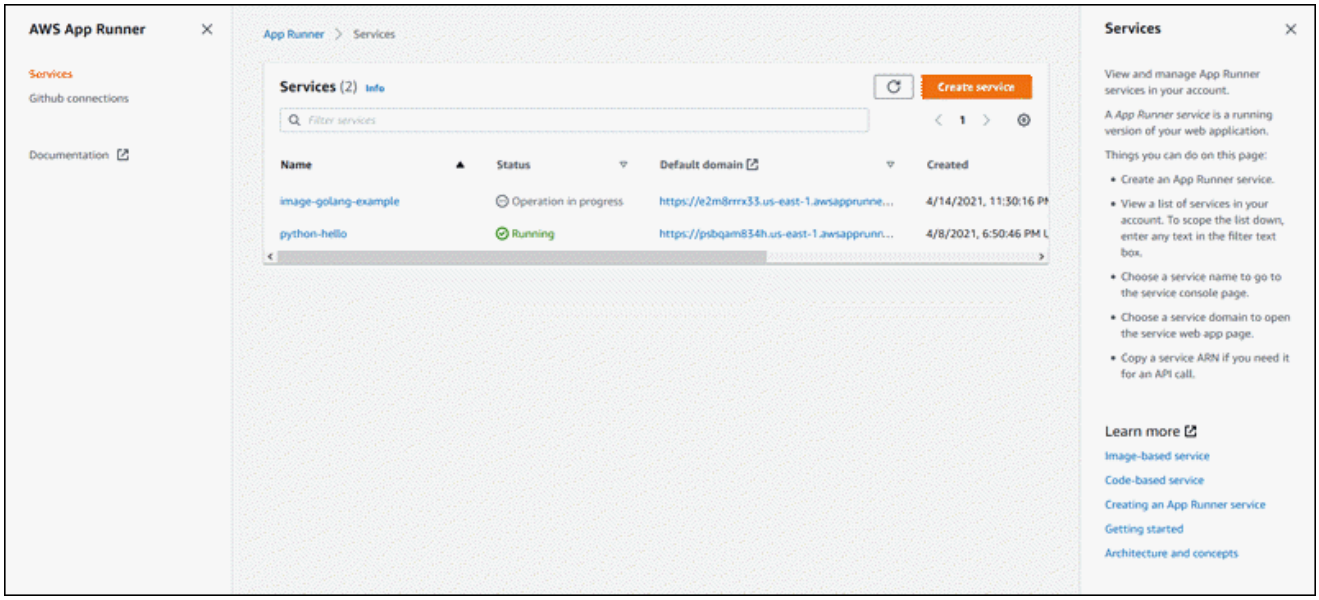
앱 러너 콘솔을 사용하여 코드에서 서비스 만들기

콘솔을 사용하여 앱 러너 서비스를 생성하려면

1. 소스 코드를 구성합니다.
    a. 열기콘솔 앱 러너에서, 그리고Regions목록에서AWS 리전.
    b. 경우AWS 계정에 아직 App Runner 서비스가 없으면 콘솔 홈 페이지가 표시됩니다. 선택앱 러너 서비스 생성.



경우AWS 계정에 기존 서비스가 있는 경우서비스페이지가 표시되고 서비스 목록이 표시됩니다. [Create service]를 선택합니다.



c. 에소스 및 배포[] 페이지에서소스섹션에서리포지토리 유형를 선택하고소스 코드 리포지토리.

d. 용GitHub 에 Connect.에서 이전에 사용한 GitHub 계정 또는 조직을 선택하거나새로 추가. 그런 다음 GitHub 자격 증명을 제공하고 연결할 GitHub 계정 또는 조직을 선택하는 과정을 진행합니다.

e. 용리포지토리에서 애플리케이션 코드가 포함된 리포지토리를 선택합니다.

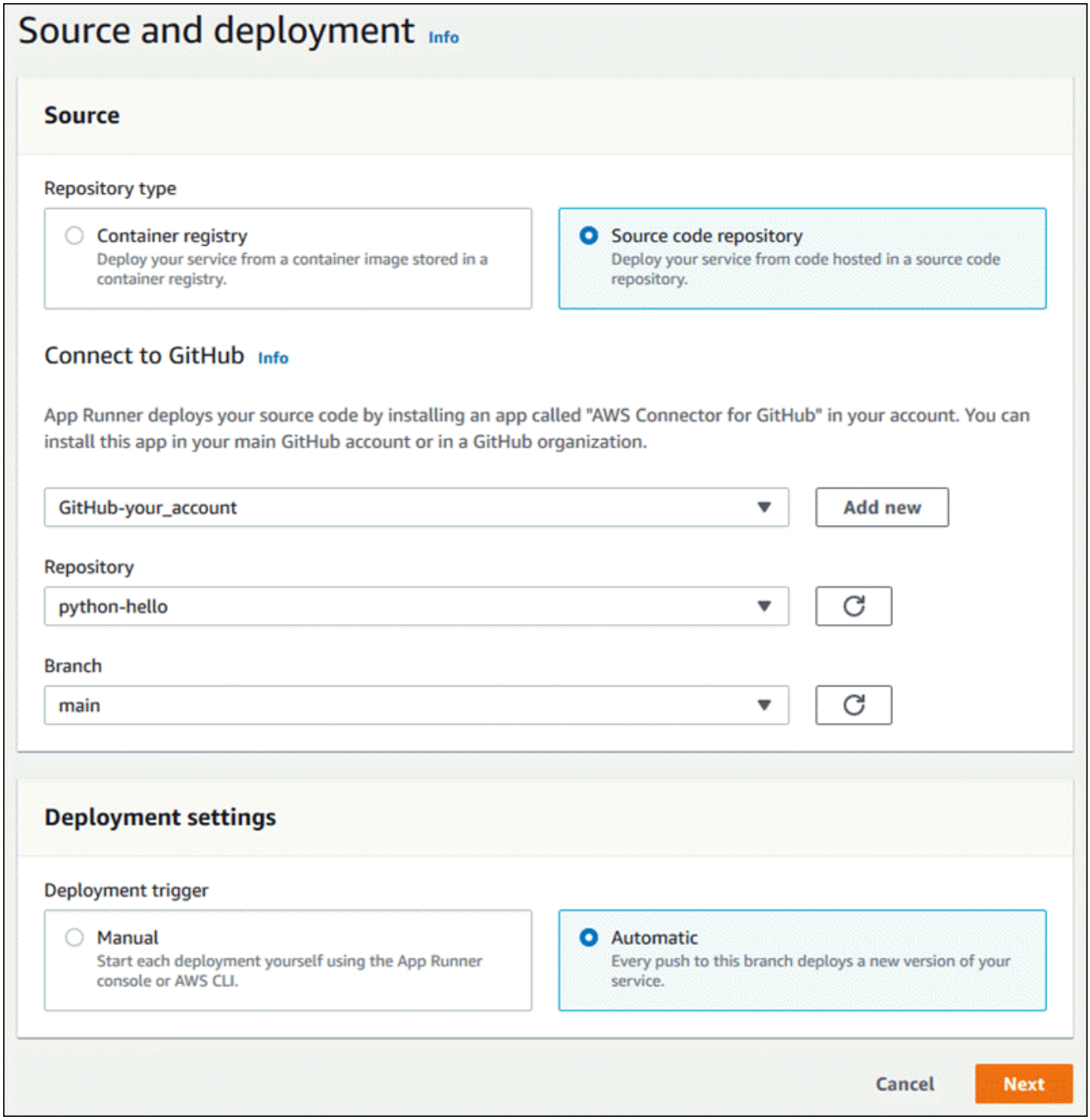
f. 용브랜치에서 배포할 브랜치를 선택합니다.

2. 배치를 구성합니다.

a. 에서배포 설정섹션에서수동또는자동.

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

b. [Next]를 선택합니다.



3. 응용 프로그램 빌드를 구성합니다.

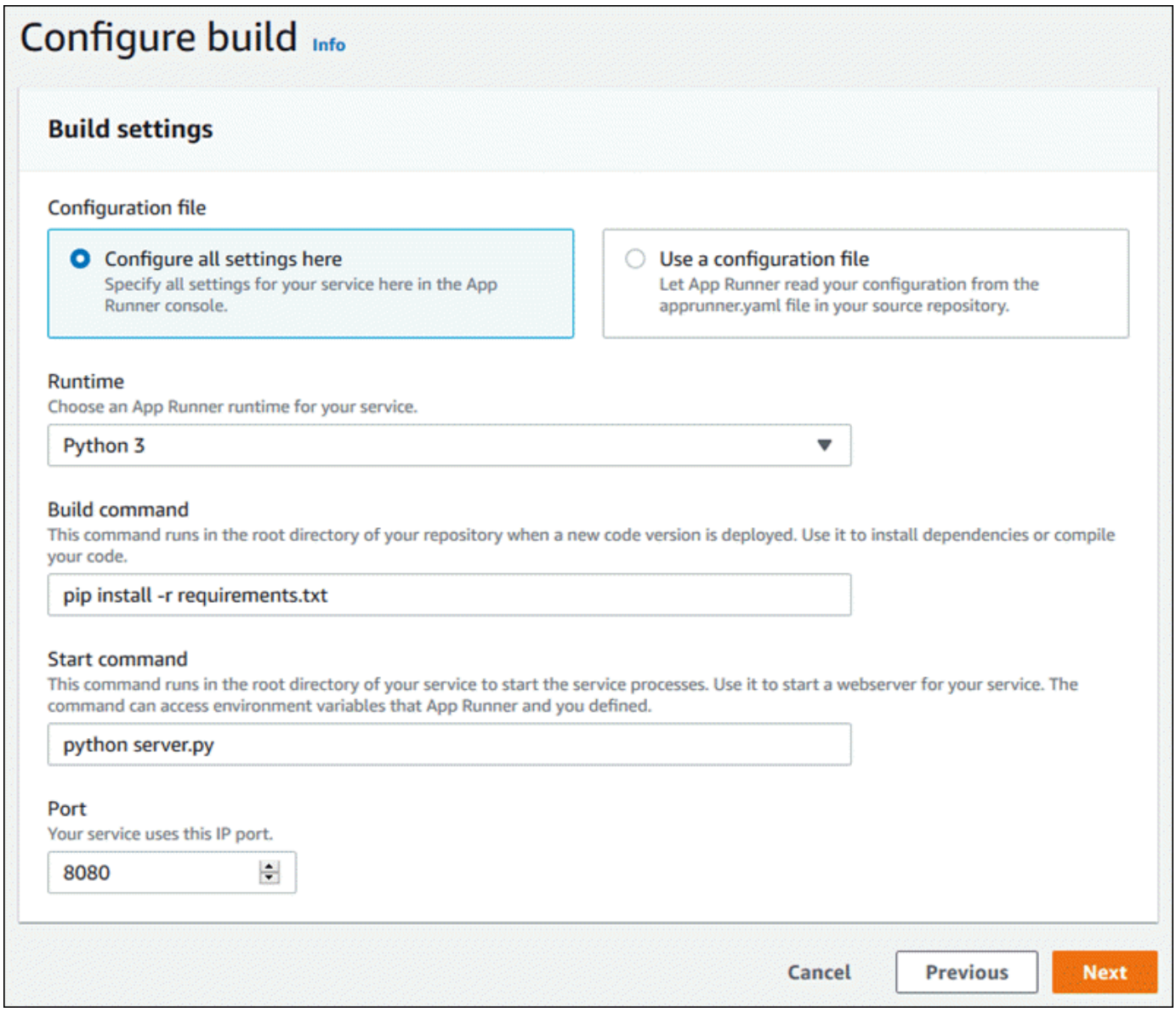
   a. 에빌드 구성[] 페이지에서구성 파일를 선택하고여기에서 모든 설정 구성저장소에 App Runner 구성 파일이 포함되어 있지 않은 경우구성 파일 사용가 작동하는지 확인합니다.

> **Note**
>
> App Runner 구성 파일은 응용 프로그램 소스의 일부로 빌드 구성을 유지 관리하는 방법입니다. 하나를 제공하면 App Runner가 파일에서 일부 값을 읽고 콘솔에서 설정할 수 없습니다.

b. 다음 빌드 설정을 제공합니다.

- 런타임— 애플리케이션에 대한 특정 관리 런타임을 선택합니다.
- 빌드 명령— 소스 코드를 통한 애플리케이션 빌드를 위한 명령을 입력합니다. 이것은 언어별 도구 또는 코드와 함께 제공되는 스크립트 일 수 있습니다.
- 시작 명령— 웹 서비스를 시작하는 명령을 입력합니다.
- 포트— 웹 서비스가 수신하는 IP 포트를 입력합니다.

c. [Next]를 선택합니다.



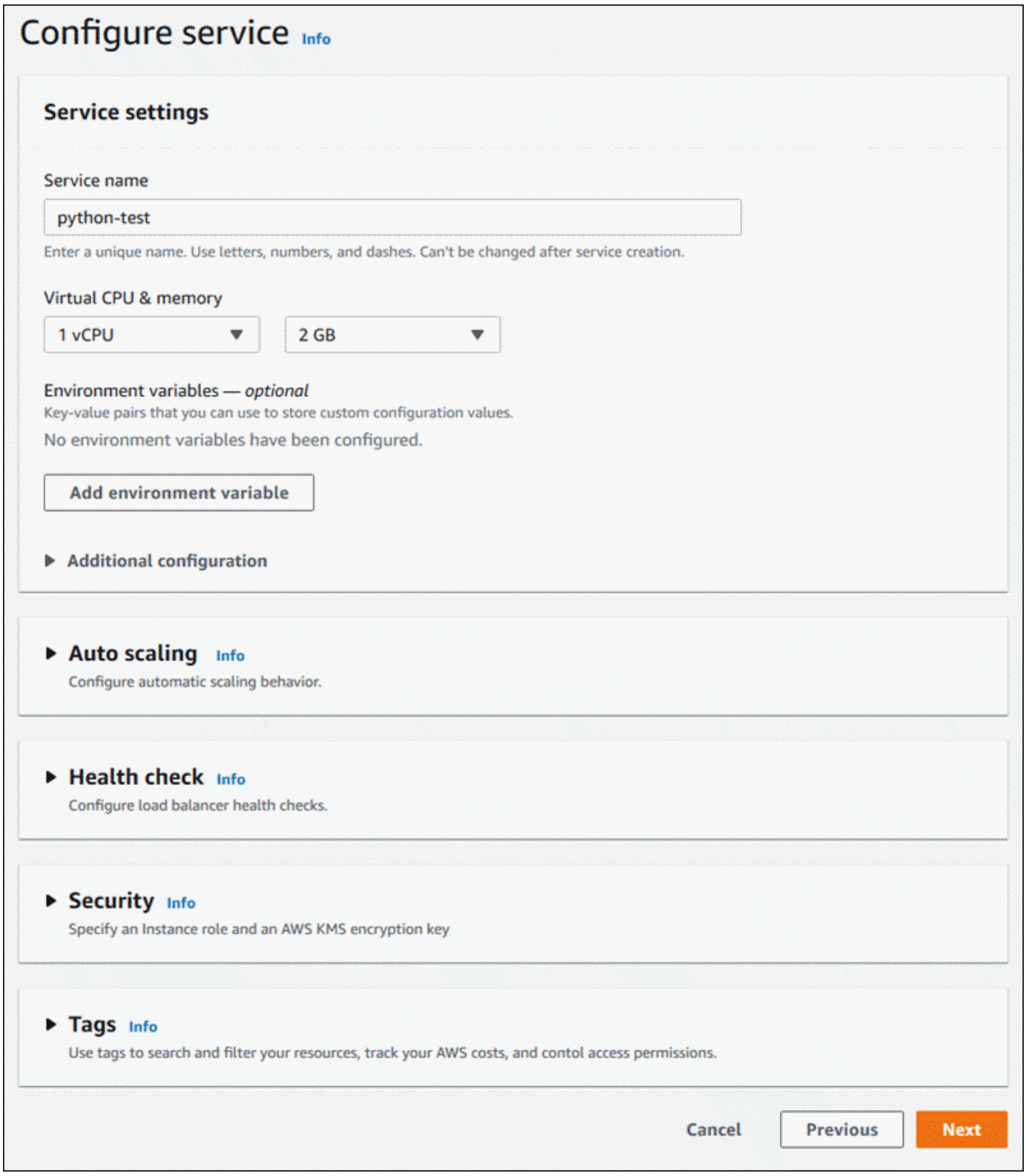
4. 서비스를 구성합니다.

   a. 에서비스 구성[] 페이지에서서비스 설정섹션에서 서비스 이름을 입력합니다.

   > **Note**
   >
   > 다른 모든 서비스 설정은 선택 사항이거나 본체에서 제공하는 기본값이 있습니다.

   b. 필요에 따라 애플리케이션 요구 사항에 맞게 다른 설정을 변경하거나 추가할 수 있습니다.

   c. [Next]를 선택합니다.



5. 에검토 및 생성페이지에서 입력한 모든 세부 정보를 확인한 다음생성 및 배포.

Result: 서비스 생성이 성공하면 콘솔에 서비스 대시보드가 표시되고서비스 개요새 서비스의.

6. 귀사의 서비스가 실행 중인지 확인합니다.

   a. 서비스 대시보드 페이지에서 서비스가 될 때까지 기다립니다.상태확장하는 데[Running].
   b. 선택을 선택합니다.기본 도메인값 (서비스 웹 사이트 URL입니다).
   c. 웹 사이트를 사용하고 올바르게 실행되고 있는지 확인하십시오.

앱 러너 API를 사용하여 코드에서 서비스 만들기 또는AWS CLI

앱 러너 API를 사용하여 서비스를 만들거나AWS CLI를 호출합니다.`CreateService`API 작업. 자세한 내용과 예제는 단원을 참조하십시오.CreateService. 특정 GitHub 조직 또는 계정을 사용하여 서비스를 처음 만드는 경우CreateConnection. 이것은 앱 러너와 GitHub 조직 또는 계정 간의 연결을 설정합니다. 앱 러너 연결에 대한 자세한 내용은 단원을 참조하십시오.the section called “연결”.

호출이 성공적인 응답을 반환하면 서비스 생성이 시작됩니다서비스객체 표시`"Status": "CREATING"`.

호출 예제는 단원을 참조하십시오.소스 코드 리포지토리 서비스 생성의AWS App RunnerAPI 참조

Amazon ECR 이미지를 통한 서비스 생성

다음 단원에서는 소스가 컨테이너 이미지일 때 App Runner 서비스를 생성하는 방법을 보여줍니다.Amazon ECR. Amazon ECRAWS서비스에 연결됩니다. 따라서 Amazon ECR 이미지를 기반으로 서비스를 생성하려면 필요한 Amazon ECR 작업 권한이 포함된 액세스 역할을 앱 러너에 제공해야 합니다.

> **Note**
>
> 이미지가 공개적으로 사용 가능한 Amazon ECR Public에 저장되어 있는 경우에는 액세스 역할이 필요하지 않습니다.

서비스를 만드는 동안 App Runner는 제공하는 이미지의 컨테이너 인스턴스를 실행하는 서비스를 시작합니다. 이 경우에는 빌드 단계가 없습니다.

주제

- 앱 러너 콘솔을 사용하여 이미지에서 서비스 만들기
- 앱 러너 API를 사용하여 이미지에서 서비스 만들기 또는AWS CLI

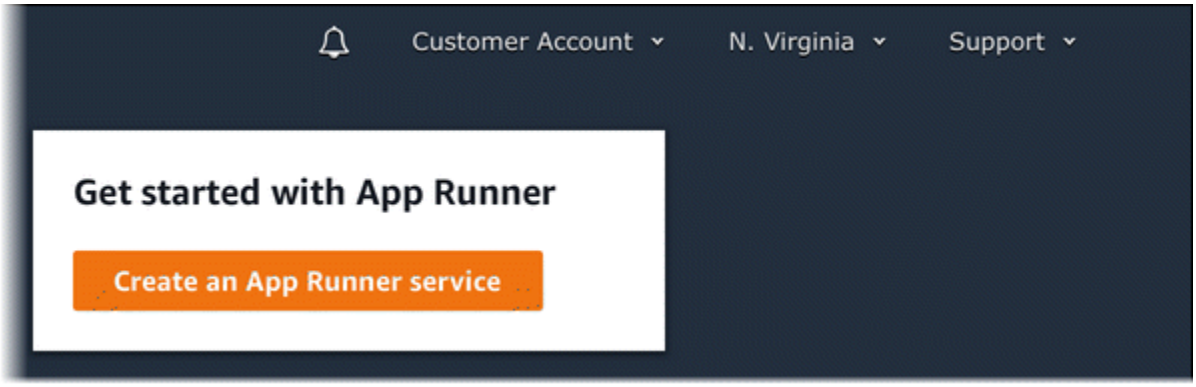
앱 러너 콘솔을 사용하여 이미지에서 서비스 만들기

콘솔을 사용하여 앱 러너 서비스를 생성하려면

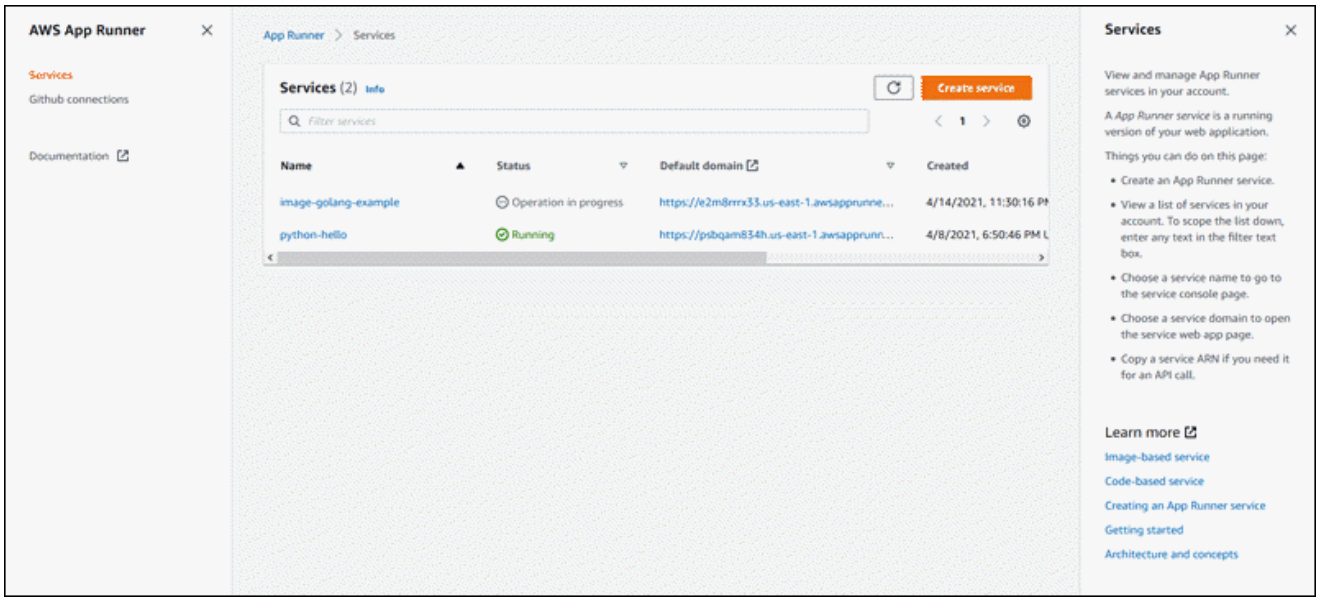
1. 소스 코드를 구성합니다.

   a. 열기콘솔 앱 러너에서, 그리고Regions목록에서AWS 리전.

   b. 경우AWS 계정에 아직 App Runner 서비스가 없으면 콘솔 홈 페이지가 표시됩니다. 선택앱 러너 서비스 생성.



   경우AWS 계정에 기존 서비스가 있는 경우서비스페이지가 표시되고 서비스 목록이 표시됩니다. [Create service]를 선택합니다.



   c. 에소스 및 배포[] 페이지에서소스섹션에서리포지토리 유형를 선택하고컨테이너 레지스트리.

   d. 용Provider에서 이미지가 저장된 공급자를 선택합니다.

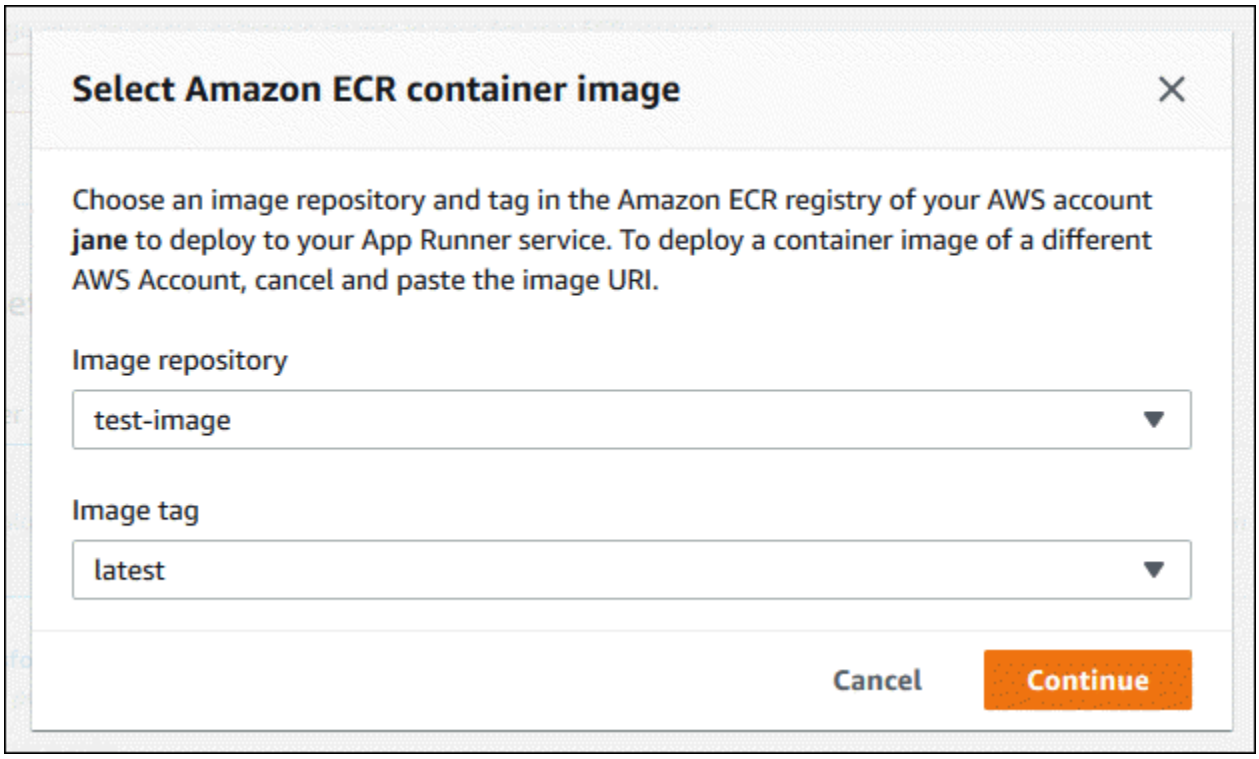
   - Amazon ECR— Amazon ECR에 저장된 프라이빗 이미지AWS 계정.

- Amazon ECR— Amazon ECR 공용에 저장된 공개적으로 읽을 수 있는 이미지입니다.

e. 용컨테이너 이미지 URI를 선택하고찾아보기.

f. 에서아마존 ECR 컨테이너 이미지 선택대화 상자에서이미지 리포지토리에서 이미지가 들어 있는 저장소를 선택합니다.

g. 용이미지 태그에서 배포하려는 특정 이미지 태그를 선택합니다 (예:최신를 선택한 다음계속.



2. 배치를 구성합니다.

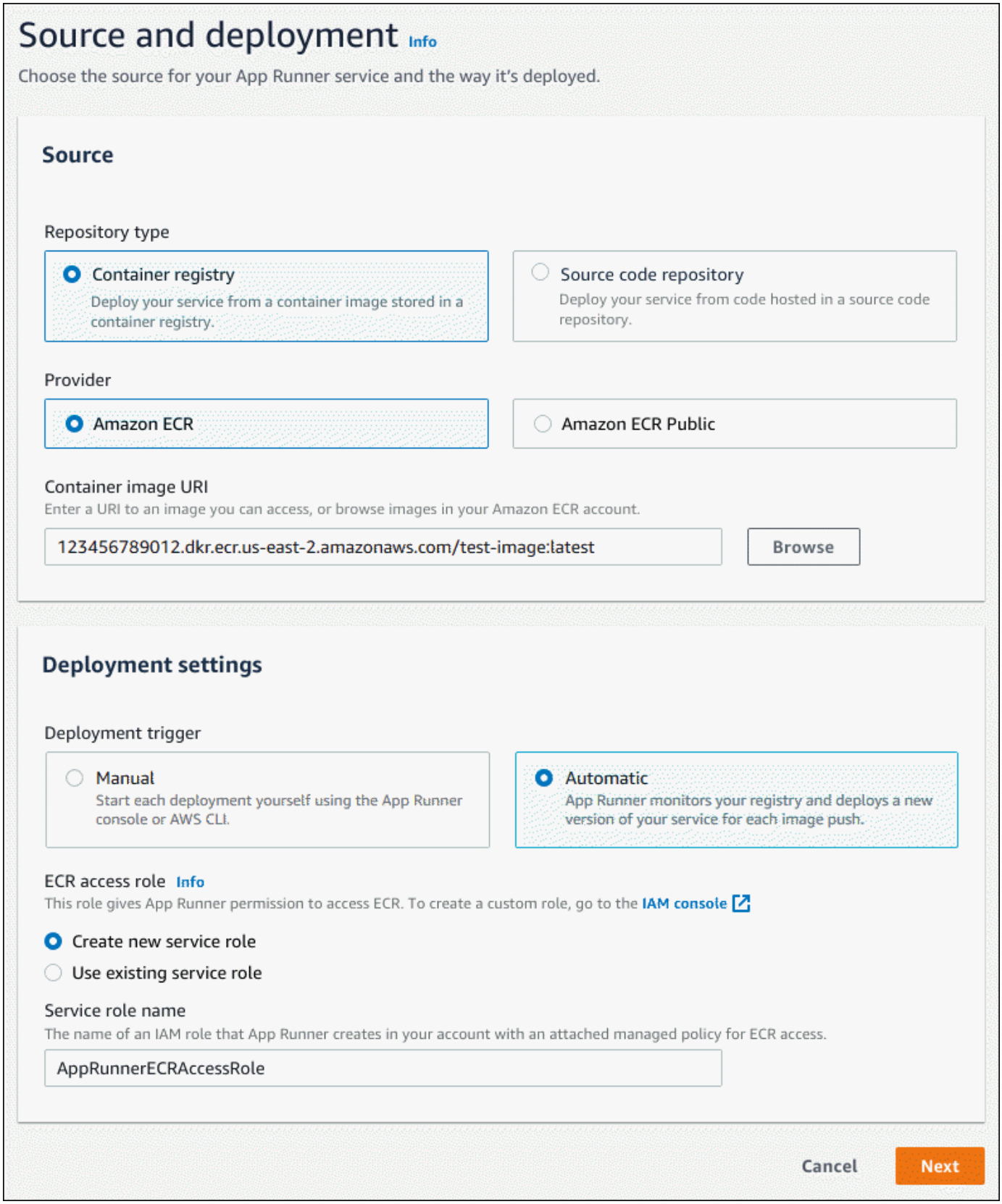
a. 에서배포 설정섹션에서수동또는자동.

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

> **Note**
> 앱 러너는 Amazon ECR 공개 이미지에 대한 자동 배포를 지원하지 않습니다.

b. [Amazon ECR공급자]ECR 액세스에서 기존 서비스 역할을 선택하거나 새 역할을 선택합니다. 수동 배포를 사용하는 경우 배포 시 IAM 사용자 역할을 사용하도록 선택할 수도 있습니다.

c. [Next]를 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

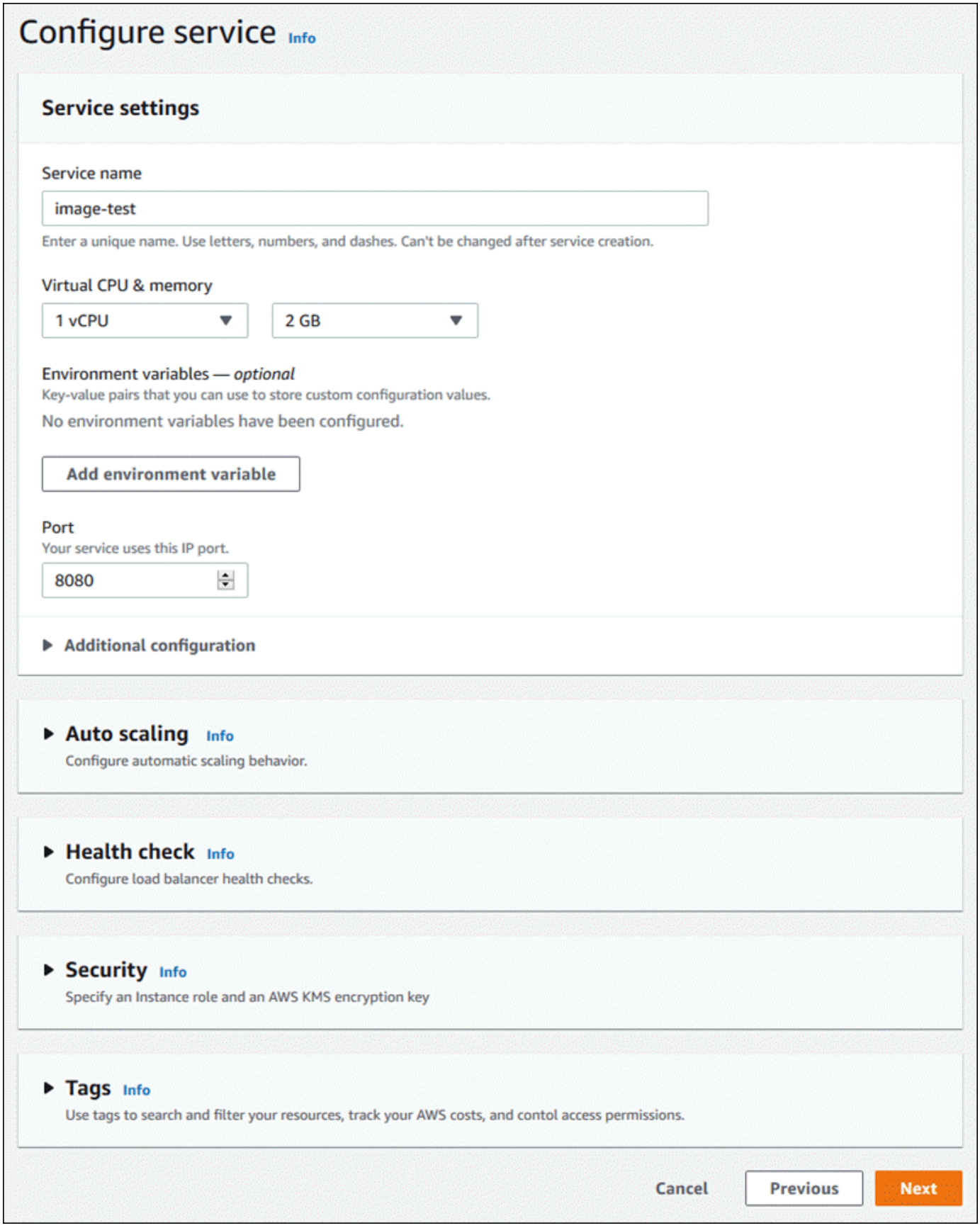
3. 서비스를 구성합니다.

   a. 에서비스 구성[] 페이지에서서비스 설정섹션에서 서비스 웹 사이트에서 수신하는 서비스 이름과 IP 포트를 입력합니다.

   > **Note**
   >
   > 다른 모든 서비스 설정은 선택 사항이거나 본체에서 제공하는 기본값이 있습니다.

   b. (선택 사항) 응용 프로그램의 필요에 맞게 다른 설정을 변경하거나 추가합니다.

   c. [Next]를 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. 에검토 및 생성페이지에서 입력한 세부 정보를 모두 확인한 후생성 및 배포.

   Result: 서비스 생성이 성공하면 콘솔에 서비스 대시보드가 표시되고서비스 개요새 서비스의.

5. 귀사의 서비스가 실행 중인지 확인합니다.

   a. 서비스 대시보드 페이지에서 서비스가 될 때까지 기다립니다.상태확장하는 데[Running].

   b. 선택을 선택합니다.기본 도메인값 (서비스 웹 사이트 URL입니다).

   c. 웹 사이트를 사용하고 올바르게 실행되고 있는지 확인하십시오.

앱 러너 API를 사용하여 이미지에서 서비스 만들기 또는AWS CLI

앱 러너 API를 사용하여 서비스를 만들거나AWS CLI를 호출합니다.CreateServiceAPI 작업.

호출이 성공적인 응답을 반환하면 서비스 생성이 시작됩니다서비스객체 표시`"Status": "CREATING"`.

호출 예제는 단원을 참조하십시오.소스 이미지 저장소 서비스 만들기의AWS App RunnerAPI 참조

서비스 생성에 실패할 경우

App Runner 서비스를 만들려는 시도가 실패하면 서비스에`CREATE_FAILED`(생성 실패콘솔).

응용 프로그램 코드, 빌드 프로세스 또는 구성 문제, 리소스 할당량에 도달했거나 기본AWS서비스를 사용할 수 있습니다. 오류 문제를 해결하려면 다음 작업을 수행하는 것이 좋습니다. 먼저 서비스 이벤트와 로그를 읽고 오류의 원인을 찾으십시오. 그런 다음 코드 또는 구성을 필요에 따라 변경합니다. 마지막으로 서비스 할당량에 도달하면 하나 이상의 서비스를 삭제하십시오. 그런 다음이 모든 단계를 완료 한 후 서비스를 다시 생성하십시오.

> **Important**
>
> 실패한 서비스를 사용할 수 없습니다. 초기 생성 시도 이후에도 추가 요금이 발생하지 않습니다. 그러나 App Runner는 실패한 서비스를 자동으로 삭제하지 않으며 여전히 서비스 할당량에 포함됩니다. 실패 분석이 완료되면 실패한 서비스를 삭제해야 합니다.

새 애플리케이션 버전 배포

때서비스를 생성하려면inAWS App Runner에서 애플리케이션 소스 (컨테이너 이미지 또는 소스 저장소) 를 구성합니다. App Runner는 서비스를 실행하기 위해 리소스를 프로비저닝하고 응용 프로그램을 배포합니다.

이 항목에서는 새 버전을 사용할 수 있을 때 응용 프로그램 원본을 App Runner 서비스에 다시 배포하는 방법에 대해 설명합니다. 이미지 저장소의 새 이미지 버전이거나 코드 저장소의 새 커밋이 될 수 있습니다. App Runner는 서비스에 배포하는 두 가지 방법을 제공합니다.자동및수동.

배포 방법

App Runner는 응용 프로그램 배포가 시작되는 방법을 제어할 수 있는 다음과 같은 방법을 제공합니다.

자동 배포

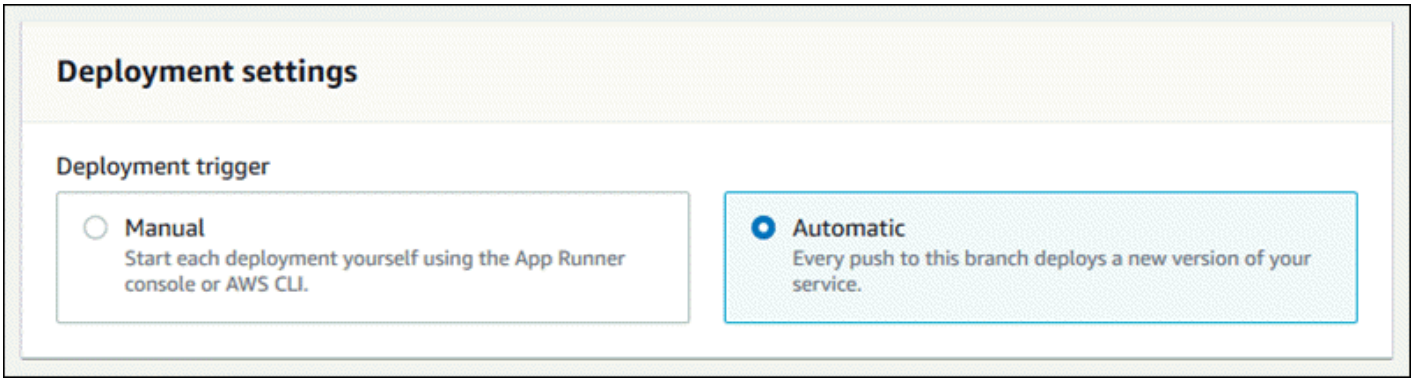
서비스에 대한 CICD (연속 통합 및 배포) 동작을 원할 경우 자동 배포를 사용합니다. 앱 러너는 이미지 또는 코드 저장소를 모니터링합니다. 새 이미지 버전을 이미지 리포지토리에 푸시하거나 코드 리포지토리에 새 커밋을 할 때마다 App Runner는 추가 작업없이 서비스에 자동으로 배포합니다.

수동 배포

서비스에 대한 각 배포를 명시적으로 시작하려는 경우 수동 배포를 사용합니다. 서비스에 대해 구성한 리포지토리에 배포하려는 새 버전이 있는 경우 배포를 시작합니다. 자세한 내용은 the section called “수동 배포” 단원을 참조하세요.

다음 방법으로 서비스에 대한 배포 방법을 구성할 수 있습니다.

- 콘솔- 생성 중인 새 서비스 또는 기존 서비스의 경우배포 설정의 단원소스 및 배포구성 페이지 (configuration) 에서수동또는자동.



- API 또는AWS CLI— 중 하나에 대한 호출에서CreateService또는UpdateService작업을 수행하려면`AutoDeploymentsEnabled`멤버 (configuration)SourceConfiguration매개 변수를`False`를 사용하여 수동 배포 또는`True`를 사용하여 자동 배포합니다.

수동 배포

수동 배포를 사용하면 서비스에 대한 각 배포를 명시적으로 시작해야 합니다. 새 버전의 응용 프로그램 이미지 또는 코드를 배포할 준비가 된 경우 다음 섹션을 참조하여 콘솔과 API를 사용하여 배포를 수행하는 방법을 배울 수 있습니다.

앱 러너 콘솔을 사용하여 응용 프로그램 버전 배포

앱 러너 콘솔을 사용하여 배포하려면

1. 열기콘솔 앱에 있는Regions목록에서AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를서비스 개요.
3. [Deploy]를 선택합니다.

   Result: 새 버전의 배포가 시작됩니다. 서비스 대시보드 페이지에서 서비스상태를 로 변경작업 진행 중.
4. 배포가 끝날 때까지 기다립니다. 서비스 대시보드 페이지에서 서비스상태로 다시 변경해야합니다.[Running].
5. 배포가 성공적인지 확인하려면 서비스 대시보드 페이지에서기본 도메인값 (서비스 웹 사이트 URL입니다). 웹 응용 프로그램을 검사하거나 상호 작용하고 버전 변경을 확인합니다.

앱 러너 API를 사용하여 응용 프로그램 버전을 배포하거나AWS CLI

앱 러너 API를 사용하여 배포하거나AWS CLI를 호출하려면StartDeploymentAPI 작업 전달할 유일한 매개 변수는 서비스 ARN 입니다. 서비스를 만들 때 응용 프로그램 소스 위치를 이미 구성했으며 App Runner는 새 버전을 찾을 수 있습니다. 호출이 성공적인 응답을 반환하면 배포가 시작됩니다.

앱 실행기 서비스 구성

언제생성AWS App Runnerservice에서 다양한 구성 값을 설정합니다. 서비스를 생성한 후 이러한 구성 설정 중 일부를 변경할 수 있습니다. 다른 설정은 서비스를 생성하는 동안에만 적용할 수 있으며 그 이

후에는 변경할 수 없습니다. 이 항목에서는 앱 러너 API, 앱 러너 콘솔 및 앱 러너 구성 파일을 사용한 서비스 구성에 대해 설명합니다.

앱 러너 API를 사용하여 서비스를 구성하거나AWS CLI

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

- UpdateServiceaction - 일부 구성 설정을 업데이트하기 위해 생성 후 호출 할 수 있습니다.
  - 업데이트될 수 있음— 설정을 업데이트할 수 있습니다.`SourceConfiguration`,`InstanceConfiguration`, 및`HealthCheckConfiguration`파라미터. 그러나,`SourceConfiguration`를 사용하는 경우 소스 유형을 코드에서 이미지로 또는 다른 방법으로 전환 할 수 없습니다. 서비스를 만들 때 제공한 것과 동일한 리포지토리를 제공해야 합니다. 그것은 어느 쪽이든`CodeRepository`또는`ImageRepository`.

    업데이트할 수도 있습니다.`AutoScalingConfigurationArn`, 서비스와 연결된 Auto Scaling 구성 리소스의 ARN 입니다.
  - 업데이트할 수 없음을 변경할 수 없습니다.`ServiceName`및`EncryptionConfiguration`매개 변수에서 사용할 수 있는CreateServiceaction. 생성된 후에는 변경할 수 없습니다. 이UpdateService작업에는 이러한 매개 변수가 포함되어 있지 않습니다.
  - API 및— 을 설정할 수 있습니다.`ConfigurationSource`매개 변수의코드구성유형 (소스 코드 저장소에`SourceConfiguration`) 을`Repository`. 이 경우 앱 러너는 구성 설정을 무시합니다.`CodeConfigurationValues`에서 이러한 설정을 읽고구성 파일리포지토리로 이동합니다. 당신이 설정 한 경우`ConfigurationSource`toAPI로 설정하면 App Runner는 API 호출에서 모든 구성 설정을 가져오고 구성 파일이 있는 경우에도 해당 구성 파일을 무시합니다.
- TagResourceaction - 서비스를 만든 후에 호출하여 서비스에 태그를 추가하거나 기존 태그의 값을 업데이트할 수 있습니다.
- UntagResourceaction - 서비스에서 태그를 제거하기 위해 서비스를 만든 후에 호출 할 수 있습니다.

앱 러너 콘솔을 사용하여 서비스 구성

콘솔은 응용 프로그램 러너 API를 사용하여 구성 업데이트를 적용합니다. 이전 섹션에서 정의한 대로 API가 부과하는 업데이트 규칙에 따라 콘솔을 사용하여 구성할 수 있는 항목이 결정됩니다. 서비스 생성 중에 사용할 수 있었던 일부 설정은 나중에 수정할 수 없습니다. 또한, 당신이 사용하기로 결정한 경우구성 파일로 설정하면 추가 설정이 콘솔에 숨겨지고 App Runner가 파일에서 해당 설정을 읽습니다.

서비스를 구성하려면

1. 열기앱 런너 콘솔에서, 그리고Regions목록에서 [] 및 []AWS 리전.
2. 탐색 창에서 [] [] []서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를서비스 개요.
3. 서비스 대시보드 페이지에서구성탭을 선택합니다.

   Result: 콘솔에는 서비스의 현재 구성 설정이 여러 섹션에 표시됩니다. 소스 및 배포,빌드 구성, 및서비스 구성.
4. 모든 범주의 설정을 업데이트하려면Edit.
5. 구성 편집 페이지에서 원하는 대로 변경한 후변경 사항 저장.

앱 러너 구성 파일을 사용하여 서비스 구성

App Runner 서비스를 만들거나 업데이트할 때 소스 저장소의 일부로 제공하는 구성 파일에서 일부 구성 설정을 읽도록 App Runner에 지시할 수 있습니다. 이렇게하면 코드 자체와 함께 소스 제어에서 소스 코드와 관련된 설정을 관리 할 수 있습니다. 구성 파일은 콘솔 또는 API를 사용하여 설정할 수 없는 특정 고급 설정도 제공합니다. 자세한 내용은 앱 러너 구성 파일 단원을 참조하세요.

앱 러너 연결 관리

때서비스 생성inAWS App Runner에서 응용 프로그램 소스 (컨테이너 이미지 또는 공급자와 함께 저장된 소스 저장소) 를 구성합니다. 타사 공급자와 함께 저장된 리포지토리가 비공개 (공개적으로 읽을 수 없음) 인 경우 App Runner는 공급자와 인증되고 인증된 연결을 설정해야 합니다. 그런 다음 App Runner는 저장소를 읽고 서비스에 배포 할 수 있습니다. 앱 러너에 저장된 코드에 액세스하는 서비스를 만들 때 연결 설정이 필요하지 않습니다.AWS 계정또는 공개 코드 위치에 있습니다.

App Runner는 연결 정보를ODBC. App Runner는 타사 연결 정보가 필요한 서비스를 만들 때 연결 리소스가 필요합니다. 다음은 연결에 대한 몇 가지 중요한 정보입니다.

- Provi— 앱 러너는 현재GitHub.
- 공유됨- 연결 리소스를 사용하여 동일한 저장소 공급자 계정을 사용하는 여러 App Runner 서비스를 만들 수 있습니다.
- 리소스 관리— 앱 러너에서 연결을 만들고 삭제할 수 있습니다. 그러나 기존 연결은 수정할 수 없습니다.

- 리소스 할당량— 연결 리소스에 할당된 할당량이 설정되어 있습니다.AWS 계정For EachAWS 리전. 이 할당량에 도달하면 새 공급자 계정에 연결하려면 먼저 연결을 삭제해야 할 수 있습니다. 앱 러너를 사용하여 연결을 삭제할 수 있습니다.콘솔또는API. 자세한 내용은 the section called “App Runner 리소스 할당량” 단원을 참조하세요.

앱 러너 콘솔을 사용하여 연결 관리

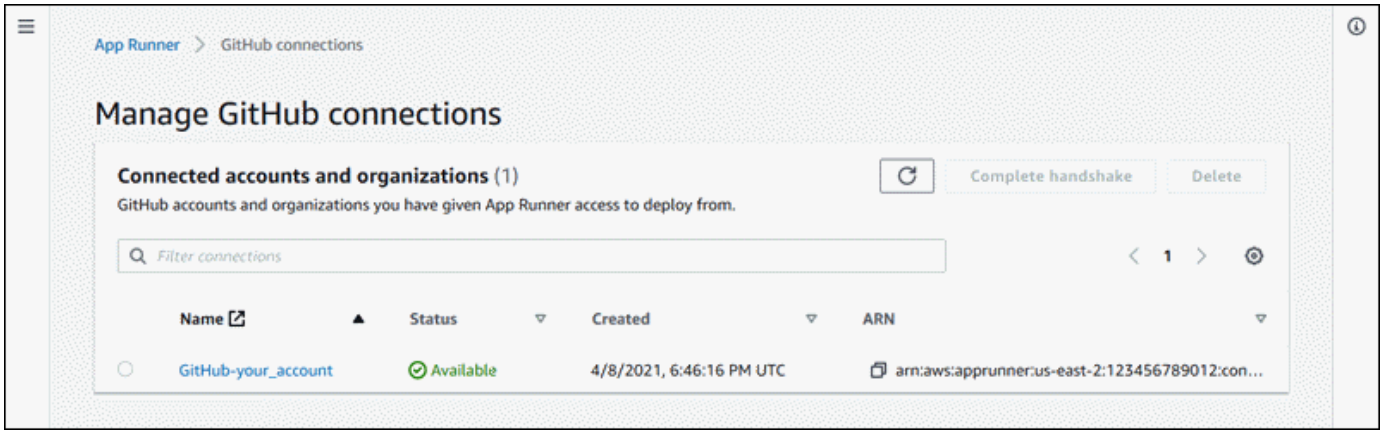
앱 러너 콘솔을 사용하여서비스 생성에서 연결 세부 정보입니다. 연결 리소스를 명시적으로 생성할 필요는 없습니다. 콘솔에서 이전에 연결한 GitHub 계정에 연결하거나 새 계정에 연결하도록 선택할 수 있습니다. 필요한 경우 App Runner는 연결 리소스를 만듭니다. 새 연결의 경우 일부 공급자 (예: GitHub)는 연결을 사용하기 전에 인증 핸드셰이크를 완료해야 합니다. 콘솔은 이 프로세스를 안내합니다.

콘솔에는 기존 연결을 관리하기 위한 페이지도 있습니다. 서비스를 만들 때 연결하지 않은 경우 연결에 대한 인증 핸드셰이크를 완료할 수 있습니다. 더 이상 사용하지 않는 연결을 삭제할 수도 있습니다. 다음 절차는 GitHub 연결을 관리하는 방법을 보여줍니다.

계정에서 GitHub 연결을 관리하려면

1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.GitHub 연결.

   그러면 콘솔에 계정의 GitHub 연결 목록이 표시됩니다.



3. 이제 목록의 연결을 사용하여 다음 작업 중 하나를 수행할 수 있습니다.

   - GitHub 계정 또는 조직 열기— 연결 이름을 선택합니다.
   - 인증 핸드셰이크 완료- 연결을 선택한 후핸드셰이크. 콘솔은 인증 핸드셰이크 프로세스를 안내합니다.
   - 연결 삭제- 연결을 선택한 후삭제. 삭제 프롬프트의 지침을 따릅니다.

앱 러너 API를 사용하여 연결 관리 또는AWS CLI

다음과 같은 App Runner API 작업을 사용하여 연결을 관리할 수 있습니다.

- CreateConnection— 저장소 공급자 계정에 대한 연결을 생성합니다. 연결이 만들어지면 App Runner 콘솔을 사용하여 인증 핸드셰이크를 수동으로 완료해야 합니다. 이 프로세스는 이전 단원에서 설명합니다.
- 목록 연결— 와 연결된 앱 러너 연결 목록을 반환합니다.AWS 계정.
- DeleteConnection— 연결을 삭제합니다. 에 대한 연결 할당량에 도달하면 불필요한 연결을 삭제해야 할 수 있습니다.AWS 계정.

앱 러너 자동 크기 조정 관리

AWS App Runner는 App Runner 애플리케이션에 대해 계산 리소스 (인스턴스) 를 자동으로 확장하거나 축소합니다. 자동 크기 조정은 들어오는 트래픽이 많을 때 적절한 요청 처리를 제공하며 트래픽 속도가 느려질 경우 비용을 절감합니다. 서비스에 대한 Auto Scaling 동작을 조정하기 위해 몇 가지 매개변수를 구성할 수 있습니다.

App Runner는 호출 된 리소스에서 자동 크기 조정 설정을 유지자동 크기 조정구성. 서비스를 만들거나 업데이트할 때 Auto Scaling 구성 리소스를 제공할 수 있습니다. 앱 러너 콘솔은 새 앱 러너 서비스를 만들 때 당신을 위해 하나를 만듭니다. Auto Scaling 구성을 제공하는 것은 선택 사항입니다. 이 옵션을 제공하지 않으면 App Runner가 권장 값과 함께 기본 Auto Scaling 구성을 제공합니다.

자동 크기 조정 구성에는name과 숫자개정. 구성의 여러 개정이 동일한 이름과 다른 개정 번호를 갖습니다. 고가용성 또는 저렴한 비용과 같은 다양한 Auto Scaling 시나리오에 대해 서로 다른 구성 이름을 사용할 수 있습니다. 각 이름에 대해 여러 개의 수정기호를 추가하여 특정 시나리오에 대한 설정을 미세 조정할 수 있습니다.

다음은 Auto Scaling 구성에 대한 몇 가지 중요한 정보입니다.

- 설정구성할 수 있는 사항은 다음과 같습니다.
  - 최대 동시성— 인스턴스가 처리하는 최대 동시 요청 수입니다. 동시 요청 수가 이 할당량을 초과하면 App Runner가 서비스를 확장합니다.
  - 최대 크기— 서비스가 확장되는 최대 인스턴스 수입니다. 대부분의 이 인스턴스 수는 서비스에 대한 트래픽을 적극적으로 제공하고 있습니다.
  - 최솟값 크기— App Runner가 서비스에 대해 프로비저닝하는 최소 인스턴스 수입니다. 서비스에는 항상 최소한 이 수의 프로비저닝된 인스턴스가 있습니다. 그들 중 일부는 적극적으로 트래픽을

제공합니다. 나머지 인스턴스 (프로비저닝된 인스턴스 및 비활성 인스턴스) 는 비용 효율적인 컴퓨팅 용량 예약으로 되어 신속하게 활성화할 수 있습니다. 프로비저닝된 모든 인스턴스의 메모리 사용량에 대한 비용을 지불합니다. 활성 하위 집합의 CPU 사용량에 대해서만 비용을 지불합니다.

App Runner는 배포 중에 프로비저닝된 인스턴스 수를 일시적으로 두 배로 늘려 이전 코드와 새 코드에 대해 동일한 용량을 유지합니다.

- 개정— 이름으로 만든 첫 번째 구성은 개정 번호 1을 가져옵니다. 같은 이름을 가진 후속 구성은 연속적인 개정 번호 (2로 시작) 를 얻습니다. App Runner 서비스를 특정 자동 크기 조정 구성 개정판 또는 최신 구성 버전과 연결할 수 있습니다.
- 공유됨— 여러 App Runner 서비스에서 단일 Auto Scaling 구성 리소스를 공유할 수 있습니다. 이 기능은 비슷한 크기 조정 요구 사항이있는 경우에 유용합니다. 특히 구성 이름을 지정하되 개정을 지정하지 않고 최신 버전의 구성을 사용하도록 여러 서비스를 구성할 수 있습니다. 이렇게 하면 서비스를 업데이트할 때 이 방법으로 구성한 모든 서비스가 Auto Scaling 구성 업데이트를 받습니다. 구성 변경에 대한 자세한 내용은 단원을 참조하십시오.the section called “Configuration”.
- 리소스 관리— App Runner를 사용하여 자동 크기 조정 구성을 만들고 삭제할 수 있습니다. 구성을 직접 업데이트할 수는 없습니다. 대신 기존 구성 이름에 새 개정을 작성하여 구성을 효과적으로 업데이트할 수 있습니다.

> **Note**
>
> 현재 App Runner 콘솔에서 단일 버전으로 구성만 만들 수 있습니다. 추가 수정 버전을 만들고 구성을 삭제하려면 App RunnerAPI.

- 리소스 할당량— Auto Scaling 구성 리소스에 대해 가질 수 있는 고유한 구성 이름 및 수정 개수에 대해 설정된 할당량이 있습니다.AWS 리전. 이러한 할당량에 도달하면 구성 이름을 삭제하거나 해당 수정 버전 중 일부를 삭제해야 더 많이 만들 수 있습니다. 앱 러너 사용API을 클릭하여 삭제합니다. 자세한 내용은 the section called “App Runner 리소스 할당량” 단원을 참조하세요.

앱 러너 콘솔을 사용하여 자동 크기 조정 관리

시작할 때서비스 생성에서 기본 자동 크기 조정 구성 또는 사용자 지정 구성을 사용할 수 있습니다. 사용자 지정 구성을 사용하려면 기존 구성을 선택하거나 새 이름 및 설정을 제공합니다. 새 구성 인 경우 App Runner는 새 Auto Scaling 구성 리소스를 만든 다음 새 서비스와 연결합니다.

앱 러너 API를 사용하여 자동 크기 조정 관리 또는AWS CLI

다음과 같은 App Runner API 작업을 통해 Auto Scaling 구성을 관리할 수 있습니다.

- 자동 확장구성 만들기자동 크기 조정— 새 Auto Scaling 구성 또는 기존 구성에 대한 리비전을 생성합니다.
- 목록 자동 크기 조정 구성— 와 연결된 Auto Scaling 구성 목록을 반환합니다.AWS 계정요약 정보와 함께.
- 설명자동 크기 조정구성— Auto Scaling 구성에 대한 전체 설명을 반환합니다.
- 삭제자동 크기 조정구성— Auto Scaling 구성을 삭제합니다. 특정 리비전이나 최신 활성 리비전을 삭제할 수 있습니다. Auto Scaling 구성 할당량에 도달하면 불필요한 Auto Scaling 구성을 삭제해야 할 수 있습니다.AWS 계정.

App Runner 서비스의 사용자 지정 도메인 이름 관리

를 생성할 때AWS App Runner서비스를 사용하면 App Runner가 도메인 이름을 할당합니다. 이 하위 도메인은awsapprunner.com응용 프로그램 러너가 소유 한 도메인입니다. 서비스에서 실행중인 웹 애플리케이션에 액세스하는 데 사용할 수 있습니다.

도메인 이름이 있으면 이를 App Runner 서비스에 연결할 수 있습니다. App Runner가 새 도메인의 유효성을 검사한 후 앱 러너 도메인 외에 응용 프로그램에 액세스하는 데 사용할 수 있습니다. 최대 5개의 사용자 지정 도메인을 연결할 수 있습니다.

> **Note**
>
> 선택적으로 포함할 수 있습니다.www도메인의 하위 도메인입니다. 그러나 현재 API에서만 지원합니다. 앱 러너 콘솔은 지원하지 않습니다.

사용자 지정 도메인을 서비스와 연결하면 App Runner는 일련의 인증서 유효성 검사 레코드를 제공합니다. App Runner가 도메인을 소유하거나 제어하는지 확인할 수 있도록 DNS (도메인 이름 시스템) 에 추가합니다. 또한 DNS에 CNAME 또는 ALIAS 레코드를 추가하여 앱 러너 도메인을 대상으로 지정합니다. 사용자 지정 도메인에 대해 하나의 레코드를 추가하고www하위 도메인 (이 옵션을 선택한 경우) 그런 다음 사용자 지정 도메인 상태가 될 때까지 기다립니다.활성 상태앱 러너 콘솔에서. 이 작업은 일반적으로 몇 분 정도 걸리지만 24-48시간이 걸릴 수 있습니다. 이 시점에서 사용자 지정 도메인의 유효성을 검사하고 App Runner는 이 도메인에서 웹 응용 프로그램으로 트래픽을 라우팅하기 시작합니다.

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

- 루트 도메인— 예:example.com. 선택적으로 연결할 수 있습니다.www.example.com도 동일한 작업의 일부로 사용할 수 있습니다.

- 하위 도메인— 예:`login.example.com`또는`admin.login.example.com`. 당신은 선택적으로 연결할 수 있습니다`www`하위 도메인도 동일한 작업의 일부로 사용할 수 있습니다.
- 와일드카드— 예:`*.example.com`. 을 사용할 수 없습니다.`www`이 예제의 경우 옵션을 선택합니다. 와일드카드는 루트 도메인의 바로 아래 하위 도메인으로만 지정할 수 있으며 그 자체로만 지정할 수 있습니다 (유효한 사양은 아닙니다.`login*.example.com`,`*.login.example.com`). 이 와일드카드 사양은 모든 즉시 하위 도메인을 연결하며 루트 도메인 자체를 연결하지 않습니다 (루트 도메인은 별도의 작업에서 연결되어야 함).

보다 구체적인 도메인 연결은 덜 구체적인 도메인 연결을 재정의합니다. 예,`login.example.com`재지정`*.example.com`. 보다 구체적인 연결의 인증서 및 CNAME이 사용됩니다.

다음 예제에서는 여러 사용자 지정 도메인 연결을 사용하는 방법을 보여줍니다.

1. 연결성`example.com`를 서비스 홈 페이지로 이동하십시오. 활성화`www`연관시킬 수도 있습니다.`www.example.com`.
2. 연결성`login.example.com`를 서비스의 로그인 페이지로 이동하십시오.
3. 연결성`*.example.com`를 사용자 정의 “찾을 수 없음”페이지로 바꿉니다.

App Runner 서비스에서 사용자 지정 도메인을 연결 해제 (연결 해제) 할 수 있습니다. 도메인 연결을 해제하면 App Runner는 이 도메인에서 웹 애플리케이션으로의 트래픽 라우팅을 중지합니다. DNS에서 이 도메인에 대한 레코드를 삭제해야 합니다.

App Runner는 내부적으로 도메인 유효성을 추적하는 인증서를 만듭니다. 이들 객체는 에 저장되어 있습니다.AWS Certificate Manager(ACM). App Runner는 도메인이 서비스에서 연결 해제된 후 또는 서비스가 삭제된 후 7일 동안 이러한 인증서를 삭제하지 않습니다.

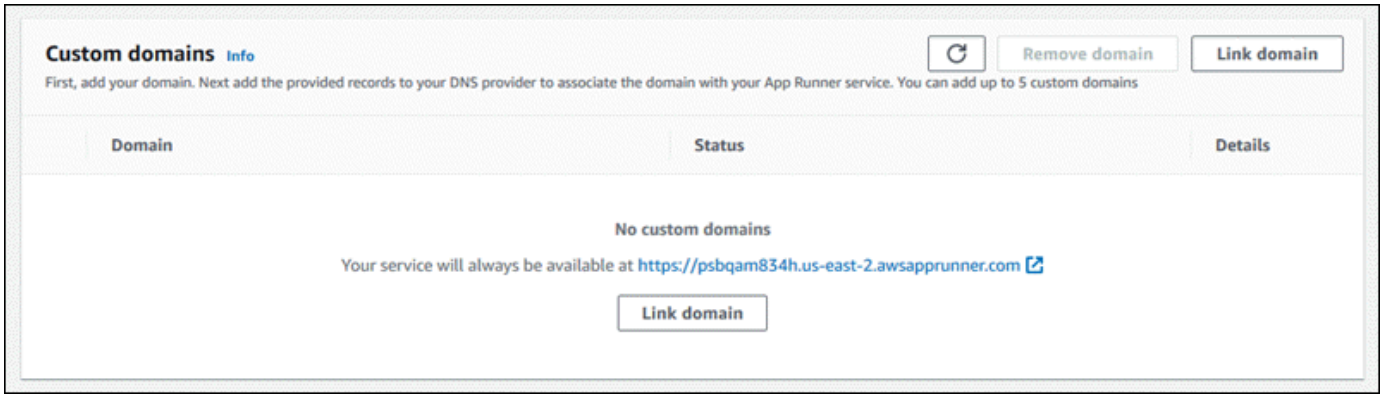
앱 러너 콘솔을 사용하여 사용자 지정 도메인 관리

App Runner 콘솔을 사용하여 사용자 지정 도메인을 연결 (연결) 하려면

1. 열기콘솔 앱 러너에 있는Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 서비스 대시보드 페이지에서사용자 지정 도메인탭에서 을 선택합니다.

   콘솔에 서비스와 연결된 사용자 지정 도메인이 표시됩니다.사용자 지정 도메인 없음.



4. 에사용자 지정 도메인탭에서 을 선택합니다.도메인 링크.
5. 에서사용자 지정 도메인 링크대화 상자에서 도메인 이름을 입력한 다음DNS 구성 보기.
6. 의 지시 사항을 따릅니다.DNS 구성페이지를 클릭하여 도메인 유효성 검사 프로세스를 시작합니다.
7. 도메인 상태가 로 변경되면활성 상태에서 도메인을 탐색하여 트래픽을 라우팅하는 데 사용할 수 있는지 확인합니다.

App Runner 콘솔을 사용하여 사용자 지정 도메인의 연결을 끊으려면 (연결 해제)

1. 에사용자 지정 도메인탭에서 연결을 해제할 도메인의 타일을 선택한 다음도메인 링크 해제 해제.
2. 에서도메인 링크 해제 해제대화 상자에서도메인 링크 해제 해제.

앱 러너 API를 사용하여 사용자 지정 도메인을 관리하거나AWS CLI

앱 러너 API를 사용하여 사용자 지정 도메인을 서비스와 연결하려면AWS CLI을 호출할 때연관사용자 정의 도메인API 작업. 호출이 성공하면CustomDomain서비스와 연결 되는 사용자 지정 도메인을 설명하는 개체입니다. 개체의 상태가 표시되어야 합니다.CREATING의 목록이 포함되어 있습니다.인증서검증레코드객체입니다. DNS에 추가할 수 있는 레코드입니다.

앱 러너 API를 사용하여 서비스에서 사용자 지정 도메인을 연결 해제하거나AWS CLI을 호출할 때연결 해제사용자 정의 도메인API 작업. 호출이 성공하면CustomDomain서비스에서 연결 해제되는 사용자 지정 도메인을 설명하는 개체입니다. 개체의 상태가 표시되어야 합니다.DELETING.

앱 러너 서비스 일시 중지 및 다시 시작

웹 응용 프로그램을 일시적으로 비활성화하고 코드 실행을 중지해야하는 경우AWS App Runner서비스를 제공합니다. 앱 러너는 서비스의 계산 용량을 0으로 줄입니다.

응용 프로그램을 다시 실행할 준비가 되면 App Runner 서비스를 다시 시작할 수 있습니다. App Runner는 새로운 계산 용량을 프로비전하고, 응용 프로그램을 배포하고, 응용 프로그램을 실행합니다. 응용 프로그램 소스가 다시 배포되지 않으므로 빌드가 필요하지 않습니다. 오히려 App Runner는 현재 배포 된 버전으로 다시 시작됩니다. 응용 프로그램은 앱 러너 도메인을 유지합니다.

> **Important**
>
> - 서비스를 일시 중지하면 응용 프로그램의 상태가 손실됩니다. 예를 들어, 코드에서 사용한 임시 저장소는 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 중지하면 서비스를 다시 시작하기 전에 새 버전을 배포할 수 없습니다.
>
> 따라서 서비스를 계속 실행 하 고 마지막 안정적인 응용 프로그램 버전으로 롤백하는 것이 좋습니다.
> - 서비스를 다시 시작하면 App Runner는 서비스를 일시 중지하기 전에 사용한 마지막 응용 프로그램 버전을 배포합니다. 서비스를 일시 중지한 후 새 소스 버전을 추가한 경우 자동 배포를 선택하더라도 App Runner는 자동으로 배포하지 않습니다. 예를 들어 이미지 저장소에 새 이미지 버전이 있거나 코드 저장소에 새 커밋이 있다고 가정합니다. 이러한 버전은 자동으로 배포되지 않습니다.
>
> 새 버전을 배포하려면 App Runner 서비스를 다시 시작한 후 수동 배포를 수행하거나 소스 저장소에 다른 버전을 추가하십시오.

비교된 일시 중지 및 삭제

Pause(일시 중지)앱 러너 서비스를일시적으로를 비활성화합니다. 계산 리소스만 종료되고 저장된 데이터 (예: 응용 프로그램 버전의 컨테이너 이미지) 는 그대로 유지됩니다. 서비스를 빠르게 재개할 수 있습니다. 애플리케이션을 새로운 컴퓨팅 리소스에 배포할 수 있습니다. 앱 러너 도메인은 동일하게 유지됩니다.

삭제앱 러너 서비스를영구적으로를 제거합니다. 저장된 데이터가 삭제됩니다. 서비스를 다시 만들어야하는 경우 App Runner는 소스를 다시 가져 와서 코드 저장소 인 경우 빌드해야합니다. 웹 응용 프로그램은 새로운 App Runner 도메인을 가져옵니다.

서비스가 일시 중지된 경우

서비스를 일시 중지하면Paused상태에서는 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, 작업이 실행 중인 서비스의 동작, 크기 조정 또는 기타 특성을 변경하는 경우 일시 중지된 서비스에서 해당 작업을 수행할 수 없습니다.

다음 목록에서는 일시 중지된 서비스에서 수행할 수 있고 수행할 수 없는 API 작업에 대한 정보를 제공합니다. 이와 동등한 콘솔 작업은 비슷하게 허용되거나 거부됩니다.

행동 당신can일시 중지된 서비스에서 수행

- *List**및*Describe**작업— 정보만 읽는 작업입니다.
- *DeleteService*- 언제든지 서비스를 삭제할 수 있습니다.
- *TagResource*,*UntagResource*— 태그는 서비스와 연결되지만 정의의 일부가 아니며 런타임 동작에 영향을 주지 않습니다.

행동 당신cannot일시 중지된 서비스에서 수행

- *StartDeployment*작업(또는수동 배포콘솔을 사용하여)
- *UpdateService*(또는 태그를 변경하는 경우를 제외하고 콘솔을 사용한 구성 변경)
- *CreateCustomDomainAssociations*, *DeleteCustomDomainAssociations*
- *CreateConnection*, *DeleteConnection*

앱 러너 콘솔을 사용하여 서비스 일시 중지 및 다시 시작

앱 러너 콘솔을 사용하여 서비스를 일시 중지하려면

1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 선택작업를 선택한 다음 를 선택합니다.Pause(일시 중지).

   서비스 대시보드 페이지에서 서비스상태를 로 변경작업 진행 중을 선택한 다음Paused. 이제 서비스가 일시 중지되었습니다.

앱 러너 콘솔을 사용하여 서비스를 다시 시작하려면

1. 선택작업를 선택한 다음 를 선택합니다.Resume.

   서비스 대시보드 페이지에서 서비스상태를 로 변경작업 진행 중.
2. 서비스가 다시 시작될 때까지 기다립니다. 서비스 대시보드 페이지에서 서비스상태변경 사항이 다시[Running].
3. 서비스 다시 시작이 성공적인지 확인하려면 서비스 대시보드 페이지에서앱 실행자 도메인USD 상당. 서비스 웹 사이트의 URL입니다. 웹 애플리케이션이 올바르게 실행되는지 확인합니다.

앱 러너 API를 사용하여 서비스를 일시 중지했다가 다시 시작하거나AWS CLI

앱 러너 API를 사용하여 서비스를 일시 중지하거나AWS CLI를 호출하고일시 중지 서비스API 작업 호출이 성공적인 응답을 반환하는 경우서비스객체 표시`"Status": "OPERATION_IN_PROGRESS"`로 설정하면 앱 러너가 서비스를 일시 중지하기 시작합니다.

앱 러너 API를 사용하여 서비스를 다시 시작하거나AWS CLI를 호출하고이력서 서비스API 작업 호출이 성공적인 응답을 반환하는 경우서비스객체 표시`"Status": "OPERATION_IN_PROGRESS"`를 선택하면 앱 러너가 서비스를 다시 시작합니다.

앱 러너 서비스 삭제

에서 실행중인 웹 응용 프로그램을 종료하려는 경우AWS App Runner서비스를 삭제할 수 있습니다. 서비스를 삭제하면 실행 중인 웹 서비스가 중지되고 기본 리소스가 제거되며 연결된 데이터가 삭제됩니다.

다음 중 하나 이상의 이유로 App Runner 서비스를 삭제할 수 있습니다.

- 더 이상 웹 응용 프로그램이 필요하지 않습니다.- 예를 들어, 은퇴했거나 사용 완료 한 개발 버전입니다.
- App Runner 서비스 할당량에 도달했습니다.— 동일한 서비스를 새로 만들려는 경우AWS 리전계정에 연결된 할당량에 도달했습니다. 자세한 내용은 the section called “App Runner 리소스 할당량” 단원을 참조하세요.
- 보안 또는 개인 정보 고려 사항— App Runner가 서비스에 대해 저장하는 데이터를 삭제하도록 합니다.

삭제 및 일시 중지

Pause(일시 중지)앱 러너 서비스를일시적으로비활성화합니다. 계산 리소스만 종료되고 저장된 데이터(예: 응용 프로그램 버전의 컨테이너 이미지) 는 그대로 유지됩니다. 서비스를 빠르게 재개할 수 있습니다. 애플리케이션을 새로운 컴퓨팅 리소스에 배포할 수 있습니다. 앱 러너 도메인은 동일하게 유지됩니다.

삭제앱 러너 서비스를영구적으로를 제거합니다. 저장된 데이터가 삭제됩니다. 서비스를 다시 만들어야하는 경우 App Runner는 소스를 다시 가져 와서 코드 저장소 인 경우 빌드해야합니다. 웹 응용 프로그램은 새로운 App Runner 도메인을 가져옵니다.

앱 러너는 무엇을 삭제합니까?

서비스를 삭제하면 App Runner가 일부 관련 항목을 삭제하고 다른 항목을 삭제하지 않습니다. 다음 목록은 세부 정보입니다.

앱 러너가 삭제하는 항목:

- 컨테이너 이미지— 배포한 이미지 또는 App Runner가 소스 코드에서 빌드한 이미지의 복사본입니다. Amazon Elastic Container Registry (Amazon ECR) 에 내부AWS 계정앱 러너가 소유하고있는.
- 서비스 구성— 앱 러너 서비스와 연결된 구성 설정입니다. Amazon DynamoDB 에 내부AWS 계정앱 러너가 소유하고있는.

앱 러너가 삭제하지 않는 항목:

- 연결- 서비스와 연결된 연결이 있을 수 있습니다. 앱 러너 연결은 여러 앱 러너 서비스간에 공유될 수 있는 별도의 리소스입니다. 더 이상 연결이 필요하지 않으면 명시 적으로 삭제할 수 있습니다. 자세한 내용은 the section called “연결” 단원을 참조하세요.
- 사용자 지정 도메인 인증서- 사용자 지정 도메인을 App Runner 서비스에 연결하는 경우 App Runner는 내부적으로 도메인 유효성을 추적하는 인증서를 만듭니다. 이들 항목은AWS Certificate Manager(ACM). App Runner는 도메인이 서비스에서 연결 해제된 후 또는 서비스가 삭제된 후 7일 동안 인증서를 삭제하지 않습니다. 자세한 내용은 the section called “사용자 지정 도메인 이름” 단원을 참조하세요.

앱 러너 콘솔을 사용하여 서비스 삭제

앱 러너 콘솔을 사용하여 서비스를 삭제하려면

1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. [Actions]를 선택한 후 [Delete]를 선택합니다.

   콘솔이 이동됩니다.서비스페이지로 이동됩니다. 삭제된 서비스는작업이 진행 중입니다.상태를 누른 다음 서비스가 목록에서 사라집니다. 이제 서비스가 삭제됩니다.

앱 러너 API를 사용하여 서비스를 삭제하거나AWS CLI

앱 러너 API를 사용하여 서비스를 삭제하려면AWS CLI를 호출합니다.DeleteServiceAPI 작업을 수행합니다. 호출이 성공적인 응답을 반환하는 경우서비스객체 표시`"Status": "OPERATION_IN_PROGRESS"`를 선택하면 앱 러너가 서비스 삭제를 시작합니다.

앱 러너 서비스에 대한 로깅 및 모니터링

AWS App Runner몇 가지와 통합AWS서비스를 사용하여 App Runner 서비스에 대한 광범위한 로깅 및 모니터링 도구 제품군을 제공합니다. 이 장의 주제에서는 이러한 기능에 대해 설명합니다.

주제

앱 러너 서비스 활동 추적

AWS App Runner는 작업 목록을 사용하여 App Runner 서비스의 활동을 추적합니다. 작업은 서비스 만들기, 구성 업데이트 및 서비스 배포와 같은 API 작업에 대한 비동기 호출을 나타냅니다. 다음 단원이 앱 Runner 콘솔과 API를 사용하여 활동을 추적하는 방법을 보여줍니다.

콘솔에서 앱 러너 서비스 활동 추적

앱 러너 콘솔은 앱 러너 서비스 활동을 표시하고 작업을 탐색하는 다양한 방법을 제공합니다.

서비스 활동을 보려면

1. 열기앱 실행 콘솔에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 및 [] []서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 서비스 대시보드 페이지에서활동탭을 선택합니다 (아직 선택하지 않은 경우).

   콘솔에 작업 목록이 표시됩니다.
4. 특정 작업을 찾으려면 검색어를 입력하여 목록의 범위를 좁히십시오. 테이블에 나타나는 값을 검색할 수 있습니다.
5. 나열된 작업을 선택하여 관련 로그를 보거나 다운로드합니다.

앱 러너 API를 사용하여 앱 러너 서비스 작업 검색 또는AWS CLI

이ListOperations작업은 App Runner 서비스의 Amazon 리소스 이름 (ARN) 이 지정된 경우 이 서비스에서 발생한 작업 목록을 반환합니다. 각 목록 항목에는 작업 ID와 일부 추적 세부 정보가 포함되어 있습니다.

CloudWatch 로그로 스트리밍된 앱 러너 로그 보기

Amazon CloudWatch Logs s를 사용하여 리소스가 다양한 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다.AWS서비스가 생성됩니다. 자세한 내용은 단원을 참조하십시오.Amazon CloudWatch Logs 사용자 안내서.

AWS App Runner는 애플리케이션 배포 및 활성 서비스의 출력을 수집하여 CloudWatch Logs 로 스트리밍합니다. 다음 섹션에서는 App Runner 로그 스트림을 나열하고 App Runner 콘솔에서 이를 보는 방법을 보여 줍니다.

App Runner 로그 그룹 및 스트림

CloudWatch Logs 는 로그 데이터를 로그 그룹에 추가로 구성하는 로그 스트림에 보관합니다. A로그 스트림특정 소스에서 로그 이벤트 시퀀스입니다. 로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림의 그룹입니다.

앱 러너는 두 개의 CloudWatch Logs 로그 그룹을 정의합니다. 각 로그 그룹은 여러 로그 스트림이 있으며, 각 로그 그룹은AWS 계정.

서비스 로그

서비스 로그 그룹에는 App Runner가 생성한 로깅 출력이 포함되어 있으며 App Runner 서비스를 관리하고 이에 따라 작동합니다.

loggroupname	예
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bc b23da/service`

서비스 로그 그룹 내에서 App Runner는 앱 러너 서비스의 수명 주기에서 활동을 캡처하는 이벤트 로그 스트림을 만듭니다. 예를 들어 응용 프로그램을 시작하거나 일시 중지하는 것일 수 있습니다.

또한 App Runner는 서비스와 관련된 장기 실행 비동기 작업마다 로그 스트림을 만듭니다. 로그 스트림 이름은 작업 유형 및 특정 작업 ID를 반영합니다.

A배포는 작업의 유형입니다. 배포 로그에는 서비스를 만들거나 새 버전의 응용 프로그램을 배포할 때 App Runner가 수행하는 빌드 및 배포 단계의 로깅 출력이 포함됩니다. 배포 로그 스트림 이름은deployment/를 클릭하고 배포를 수행하는 작업의 ID로 끝납니다. 이 작업은CreateService호출에 대한 초기 응용 프로그램 배포 또는StartDeployment각 추가 배포에 대해 호출합니다.

배포 로그 내에서 각 로그 메시지는 접두사로 시작합니다.

- `[AppRunner]`— 배포 중에 App Runner가 생성하는 출력입니다.
- `[Build]`— 자체 빌드 스크립트의 출력.

로그 스트림 이름	예
`events`	해당 사항 없음 (고정 이름)
`operation-type` `/`*`operation-id`*	`deployment/c2c8eeedea164f45 9cf78f12a8953390`

애플리케이션 로그

응용 프로그램 로그 그룹에는 실행 중인 응용 프로그램 코드의 출력이 포함됩니다.

loggroupname	예
`/aws/apprunner/` *`service-n ame`* `/`*`service-id`* `/application`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application`

응용 프로그램 로그 그룹 내에서 App Runner는 응용 프로그램을 실행하는 각 인스턴스 (확장 단위) 에 대한 로그 스트림을 만듭니다.

로그 스트림 이름	예
`instance/ instance-id`	`instance/1a80bc9134a84699b7 b3432ebeebb591`

콘솔에서 앱 Runner 로그 보기

App Runner 콘솔은 서비스에 대한 모든 로그의 요약을 표시하고 이를 보고 탐색하고 다운로드할 수 있도록 합니다.

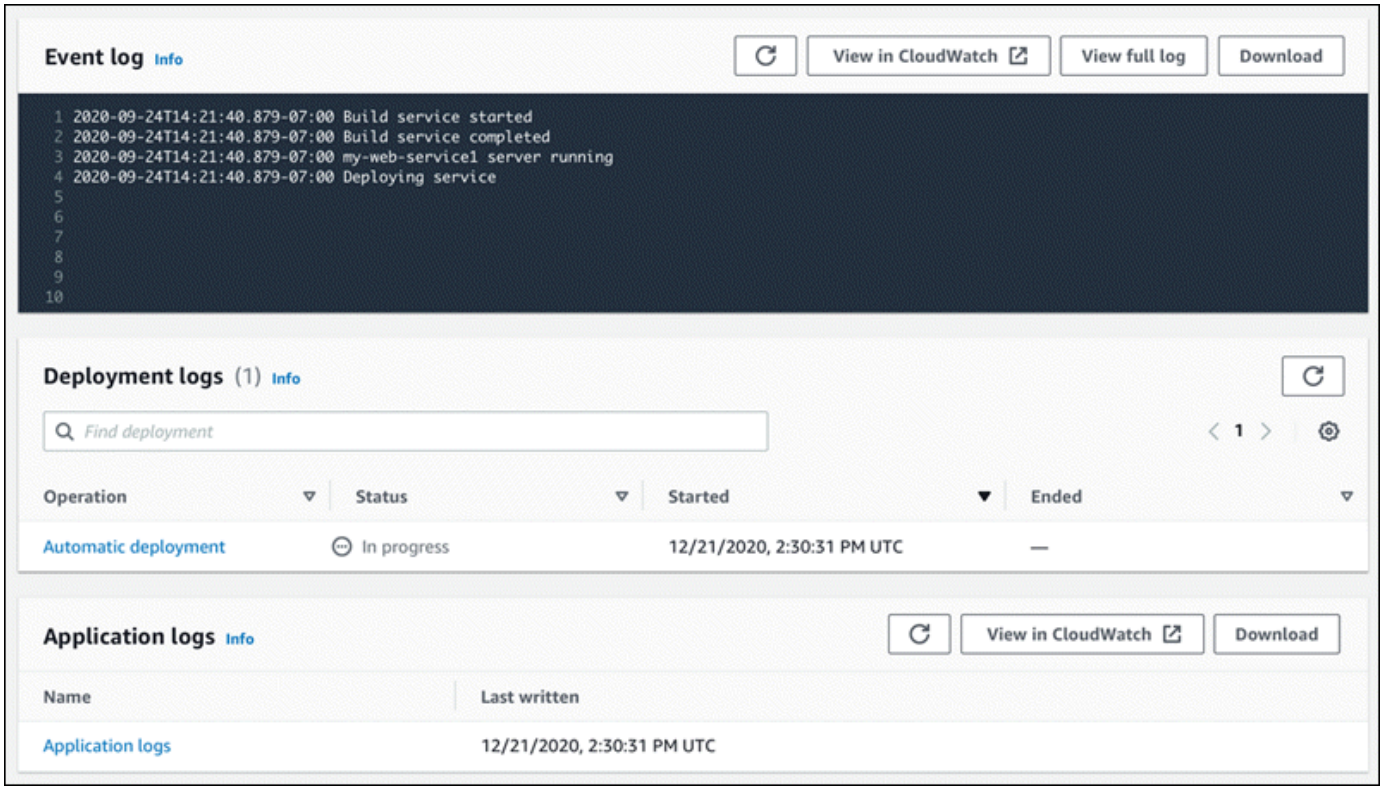
서비스 로그를 보려면

1. 열기콘솔 앱에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 를 선택합니다.서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 서비스 대시보드 페이지에서로그탭을 선택합니다.

   콘솔에는 몇 가지 유형의 로그가 여러 섹션에 표시됩니다.

   - 이벤트 로그— 앱 러너 서비스 수명 주기의 활동입니다. 콘솔이 최신 이벤트를 표시합니다.
   - 배포 로그— 앱 러너 서비스에 대한 소스 리포지토리 배포입니다. 콘솔에는 각 배포에 대해 별도의 로그 스트림이 표시됩니다.
   - 애플리케이션 로그— App Runner 서비스에 배포된 웹 응용 프로그램의 출력입니다. 콘솔은 실행 중인 모든 인스턴스의 출력을 단일 로그 스트림으로 결합합니다.



4. 특정 배포를 찾으려면 검색어를 입력하여 배포 로그 목록의 범위를 좁히십시오. 테이블에 나타나는 모든 값을 검색할 수 있습니다.
5. 로그의 내용을 보려면전체 로그 보기(이벤트 로그) 또는 로그 스트림 이름 (배포 및 응용 프로그램 로그) 을 입력합니다.
6. 선택다운로드를 클릭하여 로그를 다운로드합니다. 배치 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.
7. 선택CloudWatch 에서 보기을 클릭하여 CloudWatch 콘솔을 열고 전체 기능을 사용하여 앱 러너 서비스 로그를 탐색할 수 있습니다. 배치 로그 스트림의 경우 먼저 로그 스트림을 선택합니다.

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

CloudWatch에 보고된 앱 러너 서비스 지표 보기

Amazon CloudWatch 는 Amazon Web Services (AWS) 리소스 및 에서 실행되는 애플리케이션AWS실시간으로. CloudWatch 를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다. 또한 지표를 모니터링하는 경보를 생성할 수 있습니다. 특정 임계값에 도달하면 CloudWatch 가 알림을 보내거나 모니터링되는 리소스를 자동으로 변경합니다. 자세한 내용은 단원을 참조하십시오.Amazon CloudWatch 사용 설명서.

AWS App Runner는 App Runner 서비스의 사용량, 성능 및 가용성에 대한 가시성을 높여주는 다양한 메트릭을 수집합니다. 일부 메트릭은 웹 서비스를 실행하는 개별 인스턴스를 추적하지만 다른 메트릭은 전체 서비스 수준에 있습니다. 다음 섹션에서는 앱 러너 메트릭을 나열하고 앱 러너 콘솔에서 보는 방법을 보여 줍니다.

앱 실행자 지표

앱 러너는 서비스와 관련된 다음 메트릭CloudWatch 수집하여`AWS/AppRunner`네임스페이스

인스턴스 수준 지표는 각 인스턴스 (확장 단위) 에 대해 개별적으로 수집됩니다.

측정된 것은 무엇일까요?	지표	설명
CPU utilization	`CPUUtilization`	1분 동안의 평균 CPU 사용량입니다.
Memory utilization	`MemoryUtilization`	1분 동안의 평균 메모리 사용량입니다.

서비스 수준 지표는 전체 서비스에 대해 수집됩니다.

측정된 것은 무엇일까요?	Metrics	설명
HTTP request count	`Requests`	서비스에서 수신한 HTTP 요청의 수입니다.
HTTP status counts	`2xxStatus Responses`	범주별로 그룹화된 각 응답 상태를 반환한 HTTP 요청의 수입니다 (2XX, 4XX, 5XX).

측정된 것은 무엇일까요?	Metrics	설명

	`4xxStatusResponses`
	
	`5xxStatusResponses`	

HTTP request latency	`RequestLatency`	웹 서비스가 HTTP 요청을 처리하는 데 걸린 시간입니다.
Instance counts	`ActiveInstances`	서비스에 대한 HTTP 요청을 처리하는 인스턴스의 수입니다.

콘솔에서 앱 실행자 지표 보기

App Runner 콘솔은 App Runner가 서비스에 대해 수집하는 메트릭을 그래픽으로 표시하고 이를 탐색하는 다양한 방법을 제공합니다.

> **Note**
>
> 현재 콘솔에는 서비스 메트릭만 표시됩니다. 인스턴스 메트릭을 보려면 CloudWatch 콘솔을 사용합니다.

서비스에 대한 로그를 보려면

1. 열기앱 러너 콘솔에서, 그리고Regions목록에서 [] 를 선택합니다.AWS 리전.
2. 탐색 창에서 [] 및 [] 를 선택합니다서비스을 선택한 다음 앱 러너 서비스를 선택합니다.

   콘솔이 서비스 대시보드를 표시합니다.서비스 개요.
3. 서비스 대시보드 페이지에서지표탭을 클릭합니다.

   콘솔에 메트릭 그래프 집합이 표시됩니다.
4. 기간을 선택합니다 (예:12시간) 를 사용하여 메트릭 그래프의 범위를 해당 기간의 최근 기간으로 지정합니다.

5. 선택대시보드에 추가를 선택하거나 그래프의 메뉴를 사용하여 추가 조사를 위해 CloudWatch 콘솔의 대시보드에 관련 지표를 추가합니다.

이벤트 브리지에서 앱 러너 이벤트 처리

Amazon EventBridge를 사용하여AWS App Runner특정 패턴에 대한 서비스. 규칙의 패턴이 일치하면 EventBridge 는 다음과 같은 대상에서 작업을 시작합니다.AWS Lambda, Amazon ECSAWS Batch및 Amazon SNS 에서 확인할 수 있습니다. 예를 들어 서비스에 배포가 실패할 때마다 Amazon SNS 주제에 신호를 보내 이메일 알림을 보내는 규칙을 설정할 수 있습니다. 또는 서비스 업데이트가 실패할 때마다 Slack에 알리도록 Lambda 함수를 설정할 수 있습니다. EventBridge 에 대한 자세한 내용은 단원을 참조하십시오.Amazon EventBridge 사용 설명서.

앱 러너는 다음 이벤트 유형을 EventBridge 에 보냅니다.

- 서비스 상태— 앱 러너 서비스의 상태 변경입니다. 예를 들어 서비스 상태가`DELETE_FAILED`.
- 서비스 작업 상태— App Runner 서비스에서 긴 비동기 작업의 상태 변경입니다. 예를 들어 서비스 작성을 시작했거나 서비스 업데이트가 성공적으로 완료되었거나 오류가 있는 서비스 배포가 완료되었습니다.

앱 러너 이벤트에 대해 작동하는 EventBridge 규칙 만들기

EventBridgeevent는 일부 표준 EventBridge 필드를 정의하는 객체입니다 (예: 소스AWS서비스 및 세부 정보 (이벤트) 유형 및 이벤트 세부 정보가 포함된 이벤트별 필드 집합이 포함됩니다. EventBridge 규칙을 만들려면 EventBridge 콘솔을 사용하여이벤트 패턴(어떤 이벤트가 추적 베팅한다) 및 지정대상 작업(일치시 수행해야 할 작업). 이벤트 패턴은 일치하는 이벤트와 유사합니다. 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 단원을 참조하십시오.에 대한 규칙 생성 AWSservice의Amazon EventBridge 사용 설명서.

> **Note**
>
> 일부 서비스 지원사전 정의된 패턴EEventBridge 에 이렇게 하면 이벤트 패턴이 생성되는 방식이 간소화됩니다. 양식에서 필드 값을 선택하면 EventBridge 가 자동으로 패턴을 생성합니다. 현재 App Runner는 미리 정의 된 패턴을 지원하지 않습니다. JSON 객체로 패턴을 입력해야합니다. 이 주제의 예제를 출발점으로 사용할 수 있습니다.

앱 러너 이벤트 예

다음은 앱 러너가 EventBridge 로 보내는 이벤트에 대한 몇 가지 예입니다.

- 서비스 상태 변경 이벤트 특히, 에서 변경된 서비스`OPERATION_IN_PROGRESS`을`RUNNING`상태가 표시됩니다.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "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": {
    "PreviousStatus": "OPERATION_IN_PROGRESS",
    "CurrentStatus": "RUNNING",
    "ServiceName": "my-app",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "Message": "Service status is set to RUNNING.",
    "Severity": "INFO"
  }
}
```

- 작업 상태 변경 이벤트입니다. 특히,`UpdateService`작업이 성공적으로 완료되었습니다.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "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": {
    "Status": "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 예제에서는 하나 이상의 App Runner 이벤트를 일치시키기 위해 EventBridge 규칙에서 사용할 수 있는 이벤트 패턴을 보여 줍니다. 이벤트 패턴은 이벤트와 유사합니다. 일치시킬 필드 만 포함하고 각 필드에 스칼라 대신 목록을 제공하십시오.

- 서비스가 더 이상 존재하지 않는 특정 계정의 서비스에 대한 모든 서비스 상태 변경 이벤트 일치RUNNING상태가 표시됩니다.

```
{
  "detail-type": [ "Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "PreviousStatus": [ "RUNNING" ]
  }
}
```

- 작업이 실패한 특정 계정의 서비스에 대한 모든 작업 상태 변경 이벤트를 일치시킵니다.

```
{
  "detail-type": [ "Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "Status": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
```

```
      "ResumeServiceFailed"
    ]
  }
}
```

앱 러너 이벤트 참조

서비스 상태

서비스 상태 변경 이벤트는`detail-type`이 로 설정됩니다.`Service Status Change`. 다음과 같은 세부 필드 및 값이 표시됩니다.

```
"PreviousStatus": "any valid service status",
"CurrentStatus": "any valid service status",
"ServiceName": "your service name",
"ServiceId": "your service ID",
"Message": "Service status is set to CurrentStatus.",
"Severity": "varies"
```

작업 상태

작업 상태 변경 이벤트는`detail-type`이 로 설정됩니다.`Service Operation Status Change`. 다음과 같은 세부 필드 및 값이 표시됩니다.

```
"Status": "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 나와 있습니다.

Status	Message
`CreateServiceStarted`	서비스 생성이 시작되었습니다.
`CreateServiceCompletedSuccessfully`	서비스 생성이 성공적으로 완료되었습니다.

Status	Message
CreateServiceFailed	서비스 생성에 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
DeleteServiceStarted	서비스 삭제가 시작되었습니다.
DeleteServiceCompletedSuccessfully	서비스 삭제가 성공적으로 완료되었습니다.
DeleteServiceFailed	서비스를 삭제하지 못했습니다.
UpdateServiceStarted	
UpdateServiceCompletedSuccessfully	서비스 업데이트가 성공적으로 완료되었습니다. 새 애플리케이션 및 구성이 배포됩니다.
	서비스 업데이트가 성공적으로 완료되었습니다. 새 구성이 배포됩니다.
UpdateServiceFailed	서비스 업데이트에 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
DeploymentStarted	배포가 시작되었습니다.
DeploymentCompletedSuccessfully	배포가 성공적으로 완료되었습니다.
DeploymentFailed	배포가 실패했습니다. 자세한 내용은 서비스 로그를 참조하십시오.
PauseServiceStarted	서비스 일시 중지를 시작했습니다.
PauseServiceCompletedSuccessfully	서비스 일시 중지가 성공적으로 완료되었습니다.
PauseServiceFailed	서비스를 일시 중지하지 못했습니다.
ResumeServiceStarted	서비스 재개가 시작되었습니다.

Status	Message
`ResumeServiceCompletedSuccessfully`	서비스 재개가 성공적으로 완료되었습니다.
`ResumeServiceFailed`	서비스를 재개하지 못했습니다.

을 사용하여 앱 실행기 API 호출 로깅AWS CloudTrail

앱 러너와 통합되어AWS CloudTrail은 사용자, 역할 또는AWS앱 러너에서 서비스. CloudTrail 은 앱 러너에 대한 API 호출을 모두 이벤트로 캡처합니다. 캡처되는 호출에는 App Runner 콘솔에서 수행된 호출과 App Runner API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면, App Runner 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 Event history(이벤트 기록)에서 최신 이벤트를 볼 수 있습니다. CloudTrail 에서 수집한 정보를 사용하여 App Runner에 수행된 요청, 요청이 수행된 곳, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail 에 대한 자세한 내용은AWS CloudTrail사용 설명서.

CloudTrail 의 앱 실행자 정보

CloudTrail 이AWS 계정은 계정 생성 시 앱 러너에서 활동이 발생하면 해당 활동은 다른AWS서비스 이벤트이벤트 기록. 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다.AWS 계정. 자세한 내용은 CloudTrail 이벤트 기록에서 이벤트 보기를 참조하세요.

이벤트를 지속적으로 기록하려면AWS 계정앱 Runner 이벤트를 비롯하여 추적을 생성하십시오. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든AWS 리전. 추적은 에 있는 모든 리전의 이벤트를 로깅합니다.AWS파티션을 생성하고 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 또한 다른 구성 할 수 있습니다.AWS서비스를 사용하여 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리할 수 있습니다. 자세한 정보는 다음을 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 앱 실행자 작업은 CloudTrail 에서 로깅되며AWS App RunnerAPI 참조입니다. 예를 들어 `CreateService`, `DeleteConnection` 및 `StartDeployment` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

앱 실행기 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있도록 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 어떤 소스로부터의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 순서 지정된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예에 `CreateService` 작업을 보여 주는 CloudTrail 로그 항목이 나와 있습니다.

> **Note**
>
> 보안상의 이유로 일부 속성 값은 로그에서 수정되고HIDDEN_DUE_TO_SECURITY_REASONS. 이렇게 하면 의도하지 않은 비밀 정보가 노출되지 않습니다. 그러나 이러한 속성이 요청에 전달되거나 응답으로 반환되었음을 확인할 수 있습니다.

CloudTrail 로그 항목**`CreateService`**앱 실행자 작업

```
{
  "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"
          },
          "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"
}
```

구성 파일을 사용하여 앱 러너 서비스 옵션 설정

> **Note**
>
> 구성 파일은 다음 항목에만 적용할 수 있습니다.소스 코드를 기반으로 하는 서비스. 구성 파일을이미지 기반 서비스.

를 생성할 때AWS App Runner소스 코드 리포지토리를 사용하여AWS App Runner에는 서비스 구축 및 시작에 대한 정보가 필요합니다. 앱 러너 콘솔 또는 API를 사용하여 서비스를 만들 때마다 이 정보를 제공할 수 있습니다. 또는 사용 하 여 서비스 옵션을 설정할 수 있습니다구성 파일. 파일에서 지정하는 옵션은 소스 저장소의 일부가 되며 이러한 옵션에 대한 변경 사항은 소스 코드의 변경 내용을 추적하는 방법과 유사하게 추적됩니다. App Runner 구성 파일을 사용하여 API가 지원하는 것보다 많은 옵션을 지정할 수 있습니다. API가 지원하는 기본 옵션만 필요한 경우 구성 파일을 제공할 필요가 없습니다.

응용 프로그램 러너 구성 파일 이름이 YAML 파일입니다`apprunner.yaml`를 응용 프로그램 리포지토리의 루트 디렉터리에 추가합니다. 그것은 당신의 서비스에 대한 빌드 및 런타임 옵션을 제공합니다. 이 파일의 값은 App Runner에게 서비스를 빌드 및 시작하는 방법을 지시하고 네트워크 설정 및 환경 변수와 같은 런타임 컨텍스트를 제공합니다.

앱 러너 구성 파일에는 CPU 및 메모리와 같은 작동 설정이 포함되어 있지 않습니다.

앱 러너 구성 파일의 예는 단원을 참조하세요the section called “예제”. 전체 참조 가이드는the section called “참조”.

주제

- 앱 러너 구성 파일 예제
- 앱 러너 구성 파일 참조 참조

앱 러너 구성 파일 예제

> **Note**
>
> 구성 파일은 다음 항목에만 적용할 수 있습니다.소스 코드를 기반으로 하는 서비스. 구성 파일을 사용할 수 없습니다.이미지 기반 서비스.

다음 예시AWS App Runner구성 파일. 일부는 최소 설정이며 필수 설정만 포함합니다. 모든 구성 파일 섹션을 포함하여 다른 항목이 완료되었습니다. 앱 러너 구성 파일에 대한 개요는 단원을 참조하세요.앱 러너 구성 파일.

구성 파일 예시

최소 구성 파일

최소한의 구성 파일을 사용하여 App Runner는 다음과 같은 가정을합니다.

- 빌드 또는 실행 중에는 사용자 지정 환경 변수가 필요하지 않습니다.
- 최신 런타임 버전이 사용됩니다.
- 기본 포트 번호 및 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
```

구성 파일

이 예에서는 관리되는 런타임에 모든 구성 키를 사용하는 방법을 보여 줍니다.

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"
```

특정 관리되는 런타임 구성 파일의 예는코드 기반 서비스.

앱 러너 구성 파일 참조 참조

> **Note**
>
> 구성 파일은 다음에만 적용할 수 있습니다.소스 코드를 기반으로 하는 서비스. 다음과 같이 구성 파일을 사용할 수 없습니다.이미지 기반 서비스.

이 항목은의 구문 및 의미에 대한 포괄적 인 참조 가이드입니다AWS App Runner구성 파일을 참조하세요. App Runner 구성 파일에 대한 개요는 단원을 참조하세요.앱 러너 구성 파일.

응용 프로그램 러너 구성 파일은 YAML 파일입니다. 이름 지정`apprunner.yaml`을 빌드하고 응용 프로그램 저장소의 루트 디렉터리에 배치합니다.

구조 개요

응용 프로그램 러너 구성 파일은 YAML 파일입니다. 이름 지정`apprunner.yaml`을 빌드하고 응용 프로그램 저장소의 루트 디렉터리에 배치합니다.

앱 러너 구성 파일에는 다음과 같은 주요 부분이 포함되어 있습니다.

- 위쪽 섹션— 최상위 키를 포함합니다.
- 빌드 섹션— 빌드 단계를 구성합니다.
- 실행 섹션— 런타임 스테이지를 구성합니다.

위쪽 섹션

파일 맨 위의 키는 파일 및 서비스 런타임에 대한 일반 정보를 제공합니다. 다음과 같은 키를 사용할 수 있습니다.

- `version`–필수 사항. 앱 러너 구성 파일 버전입니다. 가장 좋은 방법은 최신 버전을 사용하는 것입니다.

  구문

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`–필수 사항. 응용 프로그램에서 사용하는 런타임의 이름입니다. 현재 다음 런타임을 사용할 수 있습니다.

  `python3`, `nodejs12`

  > **Note**
  > 관리되는 런타임의 명명 규칙은*<language-name> <major-version>*.

  구문

  ```
  runtime: runtime-name
  ```

  Example

  ```
  runtime: python3
  ```

빌드 섹션

빌드 섹션은 App Runner 서비스 배포의 빌드 단계를 구성합니다. 빌드 명령과 환경 변수를 지정할 수 있습니다. 빌드 명령이 필요합니다.

섹션은 다음과 같이 시작합니다.`build:`키를 사용하며 다음과 같은 하위 키가 있습니다.

- `commands`–필수 사항. 다양한 빌드 단계에서 App Runner가 실행하는 명령을 지정합니다. 다음과 같은 하위 키를 포함합니다.
  - `pre-build`–선택 사항입니다. 빌드 전에 실행하는 명령입니다. 예를 들어,npm종속성 또는 테스트 라이브러리.
  - `build`–필수 사항. 응용 프로그램을 빌드하기 위해 응용 프로그램 러너가 실행하는 명령입니다. 예를 들어,pipenv.
  - `post-build`–선택 사항입니다. 빌드 후에 가 실행하는 명령입니다. 예를 들어, Maven을 사용하여 빌드 결과물을 JAR 또는 WAR 파일에 패키지할 수 있으며, 테스트를 실행할 수도 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 단계에 대한 사용자 지정 환경 변수를 지정합니다. 이름-값 스칼라 매핑으로 정의됩니다. 빌드 명령에서 이름을 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 응용 프로그램 배포의 컨테이너 실행 단계를 구성합니다. 런타임 버전, 시작 명령, 네트워크 포트 및 환경 변수를 지정할 수 있습니다.

섹션은 다음과 같이 시작합니다.`run:`키를 사용하며 다음과 같은 하위 키가 있습니다.

- `runtime-version`–선택 사항입니다. App Runner 서비스에 대해 잠글 런타임 버전을 지정합니다.

  기본적으로 주 버전만 잠겨 있습니다. App Runner는 모든 배포 또는 서비스 업데이트에서 런타임에 사용할 수 있는 최신 부 버전 및 패치 버전을 사용합니다. 주 버전과 부 버전을 지정하면 둘 다 잠기고 App Runner는 패치 버전만 업데이트합니다. 주 버전, 부 버전 및 패치 버전을 지정하면 서비스가 특정 런타임 버전에서 잠기고 App Runner는 이를 업데이트하지 않습니다.

  구문

```
run:
```

```
  runtime-version: major[.minor[.patch]]
```

Example

```
runtime: python3
run:
  runtime-version: 3.7
```

- command–필수 사항. App Runner가 응용 프로그램 빌드를 완료한 후 응용 프로그램을 실행하는 데 사용하는 명령입니다.

  구문

  ```
  run:
    command: command
  ```

- network–선택 사항입니다. 응용 프로그램이 수신 대기하는 포트를 지정합니다. 여기에는 다음이 포함됩니다.
  - port–선택 사항입니다. 이 번호를 지정하면 응용 프로그램이 수신 대기하는 포트 번호입니다. 기본값은 8080입니다.
  - env–선택 사항입니다. 지정된 경우 App Runner는 기본 환경 변수에 동일한 포트 번호를 전달하는 것 외에도 포트 번호를이 환경 변수의 컨테이너에 전달합니다.PORT. 다시 말하면,env로 설정하면 App Runner는 포트 번호를 두 개의 환경 변수로 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 변수의 정의입니다. 이름-값 스칼라 매핑으로 정의됩니다. 런타임 환경에서 이름으로이 변수를 참조 할 수 있습니다.

구문

```
run:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
    - …
```

Example

```
run:
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

앱 러너 API

이AWS App Runner응용 프로그램 프로그래밍 인터페이스 (API) 는 앱 러너 서비스에 요청을 만들기위한 RESTful API입니다. API를 사용하여 앱 Runner 리소스를 생성, 나열, 설명, 업데이트 및 삭제할 수 있습니다.AWS 계정.

응용 프로그램 코드에서 API를 직접 호출하거나AWS의 SDK 명령줄 스크립트의 경우AWS CLI를 사용하여 앱 러너 서비스를 호출합니다. 에 대한 자세한 내용AWS개발자 도구에 대한 자세한 내용은구축해야 할 도구AWS.

전체 API 참조 정보는AWS App RunnerAPI 참조.

앱 러너의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안–AWS는 실행되는 인프라 보호를 책임집니다.AWS서비스 제공AWS 클라우드.AWS또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. 에 적용되는 규정 준수 프로그램에 대해 알아보려면AWS App Runner참조,AWS규정 준수 프로그램 제공 범위 내 서비스.
- 클라우드에서의 보안— 귀하의 책임은AWS서비스를 사용할 수 있습니다. 또한 귀하는 데이터의 민감도, 회사 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 주제에서는 보안 및 규정 준수 목적에 맞게 App Runner를 구성하는 방법을 보여줍니다. 또한 다른 방법을 배우려면AWS서비스를 통해 App Runner 리소스를 모니터링하고 보호하는 데 도움이 됩니다.

주제

- 앱 러너의 데이터 보호
- 앱 러너의 ID 및 액세스 관리
- 앱 러너에서 로깅 및 모니터링
- 앱 러너에 대한 규정 준수 확인
- 앱 러너의 복원성
- AWS App Runner의 인프라 보안
- 앱 러너의 구성 및 취약성 분석
- 앱 Runner 보안 모범 사례

앱 러너의 데이터 보호

이AWS 공동 책임 모델의 데이터 보호에 대해AWS App Runner. 이 모델에서 설명한 바와 같이AWS는 모든 인프라를 실행하는 글로벌 인프라를 보호할 책임이 있습니다.AWS클라우드. 이 인프라에서 호스

팅되는 콘텐츠에 대한 제어를 유지하는 것은 사용자의 책임입니다. 이 콘텐츠에는 보안 구성 및 관리 작업이 포함되어 있습니다.AWS사용하는 서비스에 대해 자세히 설명합니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 단원을 참조하십시오.AWS공동 책임 모델 및 GDPRR의 블로그 게시물은AWS보안 블로그.

데이터 보호를 위해 데이터를 보호하려면 데이터를 보호하는 것이 좋습니다.AWS계정 자격 증명을 사용하고 개별 사용자 계정을 설정합니다.AWS Identity and Access Management(IAM). 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정마다 멀티 팩터 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2 이상을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정합니다.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마십시오. 여기에는 앱 Runner 또는 기타 사용 시 작업이 포함됩니다.AWS콘솔, API,AWS CLI또는AWSSDK. App Runner 또는 기타 서비스에 입력하는 모든 데이터는 진단 로그에 포함하기 위해 선택될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마십시오.

다른 앱 Runner 보안 주제는 단원을 참조하십시오.보안.

주제

- 암호화를 사용하여 데이터 보호
- 인터네트워크 트래픽 개인 정보 보호
- VPC 엔드포인트에서 앱 러너 사용

암호화를 사용하여 데이터 보호

AWS App Runner는 지정한 저장소에서 응용 프로그램 소스 (소스 이미지 또는 소스 코드) 를 읽고 서비스에 배포하기 위해 저장합니다. 자세한 내용은 아키텍처 및 개념 단원을 참조하세요.

데이터 보호는 데이터를 보호하는 것을 의미하지만전송 중 전송 중(앱 러너에서 그리고 앱 러너로 이동) 및휴식 시에 저장되어있는 동안AWS데이터 센터).

데이터 보호에 대한 자세한 내용은 단원을 참조하십시오.the section called “데이터 보호”.

다른 앱 Runner 보안 주제는 단원을 참조하십시오.보안.

전송 중 데이터 암호화

TLS (전송 계층 보안) 를 사용하여 연결을 암호화하거나 클라이언트 측 암호화 (개체가 전송되기 전에 암호화된 위치) 를 사용하여 전송 중에 데이터를 보호할 수 있습니다. 두 방법 모두 애플리케이션 데이터를 보호하는 데 유효합니다 연결을 보호하려면 애플리케이션, 개발자 및 관리자 및 최종 사용자가 개체를 보내거나 받을 때마다 TLS를 사용하여 연결을 암호화하십시오. 앱 러너는 TLS를 통한 트래픽을 수신하도록 애플리케이션을 설정합니다.

클라이언트 측 암호화는 배포를 위해 App Runner에 제공하는 소스 이미지 또는 코드를 보호하는 올바른 방법이 아닙니다. 앱 Runner는 애플리케이션 소스에 액세스해야 하므로 암호화될 수 없습니다. 따라서 개발 또는 배포 환경과 App Runner 간의 연결을 보호하세요.

저장된 암호화 및 키 관리

애플리케이션의 유휴 데이터를 보호하기 위해 애플리케이션 소스 이미지 또는 소스 번들에서 저장된 모든 복사본을 암호화하십시오. 앱 Runner 서비스를 만들 때 고객 마스터 키 (CMK) 를 제공할 수 있습니다. 제공하면 App Runner는 제공된 키를 사용하여 소스를 암호화합니다. 사용자가 제공하지 않으면 앱 Runner는AWS대신 관리형 CMK

App Runner 서비스 만들기 매개 변수에 대한 자세한 내용은CreateService. 에 대한 자세한 내용은 AWS Key Management Service(AWS KMS참조) 의 경우AWS Key Management Service개발자 안내서.

인터네트워크 트래픽 개인 정보 보호

App Runner는 Amazon Virtual Private Cloud (Amazon VPC) 를 사용하여 App Runner 애플리케이션의 리소스 간 경계를 만들고 리소스, 온프레미스 네트워크 및 인터넷 간의 트래픽을 제어합니다.

Amazon VPC 보안에 대한 자세한 내용은 단원을 참조하십시오.Amazon VPC의 인터네트워크 트래픽 프라이버시의Amazon VPC 사용 설명서.

VPC 엔드포인트를 사용하여 App Runner에 대한 요청을 보호하는 방법에 대한 자세한 내용은 단원을 참조하십시오.the section called “VPC 엔드포인트”.

데이터 보호에 대한 자세한 내용은 단원을 참조하십시오.the section called “데이터 보호”.

다른 앱 Runner 보안 주제는 단원을 참조하십시오.보안.

VPC 엔드포인트에서 앱 러너 사용

귀하의AWS응용 프로그램을 통합할 수 있음AWS App Runner다른 서비스와의 서비스AWS에서 실행 중인Amazon Virtual Private Cloud (VPC). 애플리케이션의 일부는 VPC 내에서 App Runner에 요청할 수 있습니다. 예를 들어 사용할 수 있습니다.AWS CodePipeline를 사용하여 앱 러너 서비스에 지속적으로 배포할 수 있습니다. 응용 프로그램의 보안을 향상시키는 한 가지 방법은 이러한 App Runner 요청 (및 요청을 다른AWS서비스) 를 VPC 엔드포인트를 통해

VPC 엔드포인트를 통해 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결을 필요로 하지 않고 AWS PrivateLink 구동 지원 AWS 서비스 및 VPC 엔드포인트 서비스에 비공개로 연결할 수 있습니다.

VPC 리소스는 퍼블릭 IP 주소를 사용하여 App Runner 리소스와 상호 작용하지 않습니다. VPC 와 앱 Runner 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다. VPC 엔드포인트에 대한 자세한 내용은 단원을 참조하십시오.VPC 엔드포인트의AWS PrivateLink가이드.

앱 러너 지원AWS PrivateLink는 를 지원하여 App Runner에 대한 프라이빗 연결을 제공하고 트래픽이 퍼블릭 인터넷에 노출되지 않도록 합니다. 응용 프로그램에서 응용 프로그램을 사용하여 응용 프로그램 러너에 요청을 보낼 수 있도록하려면AWS PrivateLink에서 VPC 엔드포인트 유형을인터페이스 VPC 엔드포인트(인터페이스 끝점). 자세한 내용은 단원을 참조하십시오.인터페이스 VPC 엔드포인트 (AWS PrivateLink)의AWS PrivateLink가이드.

> **Note**
>
> App Runner 서비스의 웹 애플리케이션은 App Runner가 제공하고 구성하는 VPC 실행됩니다. 이 VPC 는 퍼블릭 인터넷에 연결되어 있습니다. App Runner는 프라이빗 VPC에서 애플리케이션을 실행하거나 VPC 엔드포인트를 생성하는 것을 지원하지 않습니다.

앱 러너용 VPC 엔드포인트 설정

VPC에서 앱 Runner 서비스에 대한 인터페이스 VPC 엔드포인트를 생성하려면인터페이스 엔드포인트 생성프로시저AWS PrivateLink가이드. 용서비스 이름를 선택하고`com.amazonaws.region.apprunner`.

VPC 네트워크 개인 정보 보호

> **Important**
>
> App Runner에 VPC 엔드포인트를 사용한다고 해서 VPC의 모든 트래픽이 인터넷 외부에 유지되는 것은 아닙니다. VPC 는 공개 (인터넷 연결) 일 수 있으며 솔루션의 일부 부분은 VPC 엔드포인트를 사용하여AWSAPI 호출입니다. 예,AWS서비스에서 퍼블릭 엔드포인트를 사용하여 다른 서비스를 호출할 수 있습니다. VPC 솔루션에 트래픽 프라이버시가 필요한 경우 이 섹션을 참조하십시오.

VPC 네트워크 트래픽의 프라이버시를 보호하려면 다음을 고려하십시오.

- DNS 이름 활성화- 응용 프로그램의 일부가 인터넷을 통해 App Runner에 요청을 보낼 수 있습니다.`apprunner.region.amazonaws.com`퍼블릭 엔드포인트입니다. 퍼블릭 인터넷 액세스로 VPC 구성한 경우 이러한 요청은 사용자에게 아무런 표시 없이 성공합니다. 이를 방지하려면 엔드포인트 생성 중에 [Enable DNS name]을 활성화해야 합니다(기본적으로 true). 그러면 퍼블릭 서비스 엔드포인트를 인터페이스 VPC 엔드포인트에 매핑하는 DNS 항목이 VPC에 추가됩니다.
- 추가 서비스에 대한 VPC 엔드포인트 구성- 솔루션이 다른AWS서비스. 예,AWS CodePipeline에 요청을 보낼 수 있습니다.AWS CodeBuild. 이러한 서비스에 대해서도 VPC 엔드포인트를 구성하고 이러한 엔드포인트에서 DNS 이름을 사용하도록 설정합니다.

엔드포인트 정책을 사용하여 VPC 엔드포인트로 액세스 제어

기본적으로 VPC 엔드포인트는 연결된 서비스에 대한 모든 액세스를 허용합니다. 앱 Runner용 VPC 엔드포인트를 생성하거나 수정할 때엔드포인트 정책를 추가합니다.

엔드포인트 정책은AWS Identity and Access Management엔드포인트에서 지정된 서비스로의 액세스를 제어하는 (IAM) 리소스 정책입니다. 엔드포인트 정책은 엔드포인트에만 해당됩니다. 사용자 환경에 적용될 수 있는 사용자 또는 인스턴스 IAM 정책과는 별개이며 이를 재정의하거나 대체하지 않습니다. VPC 엔드포인트 정책을 작성 및 사용하는 방법에 대한 자세한 내용은 단원을 참조하십시오.VPC 엔드포인트로 서비스에 대한 액세스 제어의AWS PrivateLink가이드.

다음 예제에서는 VPC 엔드포인트에서 App Runner로의 읽기 전용 액세스를 허용합니다. VPC 엔드포인트를 사용할 때의 최소 액세스 권한입니다. 보안 주체는 다른 정책을 통해 추가 권한을 가질 수 있습니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

앱 러너의 ID 및 액세스 관리

AWS Identity and Access Management(IAM) 은AWS관리자가 액세스 권한을 안전하게 제어할 수 있도록 지원하는 서비스AWS있습니다. IAM 관리자가 사용자를 제어할 수 있습니다.인증된(로그인) 및권한(권한이 있음) 을 사용하여 앱 러너 리소스를 사용할 수 있습니다. IAM은AWS추가 비용없이 사용할 수 있는 서비스.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

주제

- Audience
- 자격 증명을 통한 인증
- 정책을 사용하여 액세스 관리
- IAM과 앱 러너가 작동하는 방식
- 앱 러너 자격 증명 기반 정책 예제
- 앱 러너에 서비스 연결 역할 사용
- AWS App Runner의 AWS 관리형 정책
- 앱 러너 자격 증명 및 액세스 문제 해결

Audience

사용 방법AWS Identity and Access Management(IAM) 는 앱 러너에서 수행하는 작업에 따라 달라집니다.

서비스 사용자— App Runner 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 App Runner 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. 앱 러너의 기능에 액세스할 수 없는 경우앱 러너 자격 증명 및 액세스 문제 해결.

서비스 관리자— 회사에서 앱 러너 리소스를 책임지고 있는 경우 앱 러너에 대한 전체 액세스를 가지고 있을 것입니다. 직원이 어떤 App Runner 기능과 리소스에 액세스해야 하는지를 결정하는 작업은 서비스 관리자의 책임입니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 App Runner에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 단원을 참조하십시오.IAM과 앱 러너가 작동하는 방식.

IAM 관리자IAM 관리자는 App Runner에 대한 액세스 권한 관리 정책을 작성하는 방법에 대해 자세히 알아보려고 할 수 있습니다. IAM에서 사용할 수 있는 App Runner 자격 증명 기반 정책 예제를 보려면 단원을 참조하십시오.앱 러너 자격 증명 기반 정책 예제.

자격 증명을 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. 사용 하 여 로그인에 대 한 자세한 내용은AWS Management Console단원을 참조하십시오.에 로그인합니다.AWS Management ConsoleIAM 사용자 또는 루트 사용자로의IAM 사용 설명서.

수행해야 할 사항인증된에 로그인합니다.AWS) 을AWS계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 구성할 수 있습니다. 회사의 Single Sign-On 인증을 사용하거나 Google 또는 Facebook을 사용하여 로그인할 수도 있습니다. 이러한 경우 관리자는 이전에 IAM 역할을 사용하여 자격 증명 연동을 설정한 것입니다. 다른 회사의 자격 증명을 사용하여 AWS에 액세스하면 간접적으로 역할을 가정하는 것입니다.

에 직접 로그인하려면AWS Management Console의 경우 루트 사용자 이메일 주소 또는 IAM 사용자 이름과 함께 암호를 사용하십시오. 에 액세스할 수 있습니다.AWS루트 사용자 또는 IAM 사용자 액세스 키를 프로그래밍 방식으로 사용할 수 있습니다.AWSSDK 및 명령줄 도구를 사용하여 암호화 방식으로 요청에 서명할 수 있는 자격 증명을 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 이렇게 하려면 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 사용합니다. 요청 인증에 대한 자세한 내용은 단원을 참조하십시오.서명 버전 4 서명 프로세스의AWS일반 참조.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS는 멀티 팩터 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 다음을 참조하세요.멀티 팩터 인증 (MFA) 사용AWS의IAM 사용 설명서.

AWS 계정 루트 사용자

AWS 계정을 처음 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 SSO(Single Sign-In) ID로 시작합니다. 이 자격 증명을 호출할 수 있는AWSaccount루트 사용자에서 액세스할 수 있으며 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업, 심지어 관리 작업의 경우에도 루트 사용자를 사용하지 마실 것을 강력히 권장합니다. 대신, IAM 사용자를 처음 생성할 때만 루트 사용자를 사용하는 모범 사례를 준수합니다. 그런 다음 루트 사용자 자격 증명을 안전하게 보관하고 몇 가지 계정 및 서비스 관리 작업을 수행할 때만 사용합니다.

IAM 사용자 및 그룹

한IAM 사용자내 신원입니다AWS단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 계정입니다. IAM 사용자에게 사용자 이름과 암호 또는 액세스 키 세트와 같은 장기 자격 증명이 있을 수 있습니다. 액세스 키를 생성하는 방법은 단원을 참조하십시오.IAM 사용자에 대한 액세스 키 관리의IAM 사용 설명서. IAM 사용자의 액세스 키를 생성할 때는 키 페어를 보고 안전하게 저장해야 합니다. 향후에 보안 액세스 키를 복구할 수 없습니다. 그 대신 새 액세스 키 페어를 생성해야 합니다.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 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 자격 증명만 제공합니다. 자세한 내용은 다음을 참조하세요.IAM 사용자를 만들어야 하는 경우 (역할이 아님)의IAM 사용 설명서.

IAM 역할

한IAM 역할내 신원입니다AWS계정에 특정 권한을 가지고 있는지 확인합니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. IAM 역할을 임시로 수임할 수 있는 경우AWS Management Console에 의해역할 전환. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할을 사용하는 방법에 대한 자세한 내용은 단원을 참조하십시오.IAM 역할 사용의IAM 사용 설명서.

임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 임시 IAM 사용자 권한 - IAM 사용자는 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 연합된 사용자 액세스— IAM 사용자를 생성하는 대신 의 기존 자격 증명을 사용할 수 있습니다.AWS Directory Service, 엔터프라이즈 사용자 디렉터리 또는 웹 자격 증명 공급자에 대한 자세한 내용은 단원을 참조하십시오. 이 사용자를 연합된 사용자라고 합니다. AWS에서는 자격 증명 공급자를 통해 액세스가 요청되면 연합된 사용자에게 역할을 할당합니다. 연동 사용자에 대한 자세한 내용은 단원을 참조하십시오.연동 사용자 및 역할의IAM 사용 설명서.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 교차 계정 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 역할을 프록시로 사용하는 대신 리소스에 정책을 직접 연결할 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 단원을 참조하십시오.IAM 역할이 리소스 기반 정책과 어떻게 다른지의IAM 사용 설명서.
- 교차 서비스 액세스— 약간AWS서비스는 다른AWS서비스. 예를 들어 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.
  - 보안 주체 권한— IAM 사용자 또는 역할을 사용하여AWS를 사용하는 경우 보안 주체로 간주됩니다. 정책은 보안 주체에게 권한을 부여합니다. 일부 서비스를 사용할 때는 다른 서비스에서 다른 작업을 트리거하는 작업을 수행할 수 있습니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. 작업에 정책에서 추가 종속 작업이 필요한지 여부를 확인하려면 단원을 참조하십시오.에 사용되는 작업, 리소스 및 조건 키AWS App Runner의서비스 승인 참조.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM 역할입니다. 서비스 역할은 해당 계정 내에서만 액세스를 제공하며 다른 계정의 서비스에 대한 액세스를 부여하는 데 사용할 수 없습니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 단원을 참조하십시오.역할을 만들어 권한을 위임할 AWSservice의IAM 사용 설명서.
  - 서비스 연결 역할— 서비스 연결 역할은 서비스 역할의 한 유형입니다.AWS서비스에 로그인합니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션— IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 있는 애플리케이션의 임시 자격 증명을 관리하고AWS CLI또는AWSAPI 요청. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 그 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 만들어야 합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램

이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 단원을 참조하십시오.IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여의IAM 사용 설명서.

IAM 역할을 사용할지 아니면 IAM 사용자를 사용할지에 대한 자세한 내용은IAM 역할을 만들어야 하는 경우 (사용자가 아님)의IAM 사용 설명서.

정책을 사용하여 액세스 관리

에서 액세스를 제어합니다.AWS정책을 생성하고 IAM 자격 증명 또는AWS있습니다. 정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. 루트 사용자 또는 IAM 사용자로 로그인하거나 IAM 역할을 수임할 수 있습니다. 그런 다음 요청을 할 때AWS에서는 관련 자격 증명 기반 또는 리소스 기반 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는지 여부를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 내용은 단원을 참조하십시오.JSON 정책 개요의IAM 사용 설명서.

관리자는 다음을 사용할 수 있습니다AWS누가 무엇에 액세스 할 수 있는지를 지정하는 JSON 정책입니다. 즉,보안 주체수행 할 수 있습니다작업란 무엇입니까?리소스, 그리고 무엇에조건.

모든 IAM 개체(사용자 또는 역할)는 처음에는 권한이 없습니다. 다시 말해, 기본적으로 사용자는 아무 작업도 수행할 수 없으며, 자신의 암호를 변경할 수도 없습니다. 사용자에게 작업을 수행할 권한을 부여하기 위해 관리자는 사용자에게 권한 정책을 연결해야 합니다. 또한 관리자는 의도한 권한을 가지고 있는 그룹에 사용자를 추가할 수 있습니다. 관리자가 그룹에 권한을 부여하면 그룹의 모든 사용자가 해당 권한을 받습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

자격 증명 기반 정책

자격 증명 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 있는지를 제어합니다. 자격 증명 기반 정책을 만드는 방법을 알아보려면 단원을 참조하십시오.IAM 정책 생성의IAM 사용 설명서.

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 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에게 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS 관리형 정책과 고객 관리

형 정책이 포함되어 있습니다. 관리형 정책을 사용할지 아니면 인라인 정책을 사용할지를 알아보려면 단원을 참조하십시오.관리형 정책과 인라인 정책의 선택의IAM 사용 설명서.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 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 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연합된 사용자 또는AWS서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 을 사용할 수 없습니다.AWS리소스 기반 정책에서 IAM의 관리형 정책을 생성합니다.

ACL(액세스 제어 목록)

ACL(액세스 제어 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3,AWS WAFAmazon VPC 는 ACL을 지원하는 서비스의 예입니다. ACL에 대한 자세한 내용은 단원을 참조하십시오.ACL(액세스 제어 목록) 개요의Amazon Simple Storage Service 개발자 안내서.

기타 정책 유형

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책이 IAM 개체(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 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 내용은 단원을 참조하십시오.IAM 엔터티에 대한 권한 경계의IAM 사용 설명서.
- 서비스 제어 정책(SCP)— SCP는 조직 또는 조직 단위 (OU) 에 최대 권한을 지정하는 JSON 정책입니다.AWS Organizations.AWS Organizations는 그룹화하고 중앙에서 관리할 수 있는 서비스입니다.AWS계정이 있는지 확인합니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을

임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 를 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다.AWS계정 루트 사용자를 참조하십시오. Organizations 및 SCP에 대한 자세한 내용은 단원을 참조하십시오.SCP 작동 방식의AWS Organizations사용 설명서.

- 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 역할의 자격 증명 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 단원을 참조하십시오.세션 정책의IAM 사용 설명서.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 자세한 방법은 다음과 같습니다.AWS여러 정책 유형이 관련될 때 요청을 허용할지 여부를 결정합니다. 자세한 내용은 단원을 참조하십시오.정책 평가 로직의IAM 사용 설명서.

IAM과 앱 러너가 작동하는 방식

IAM을 사용하여 에 대한 액세스를 관리하기 전에AWS App Runner의 경우 App Runner에서 사용할 수 있는 IAM 기능을 이해해야 합니다. 앱 러너 및 기타 방법을 상위 수준에서 보려면AWS서비스는 IAM과 함께 작동합니다.AWSIAM과 함께 작업하는 서비스의IAM 사용 설명서.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

주제

- 앱 러너 자격 증명 기반 정책
- 앱 러너 리소스 기반 정책
- 앱 러너 태그 기반 권한 부여
- 앱 러너 사용자 권한
- IAM 앱 러너 역할

앱 러너 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. App Runner는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 단원을 참조하십시오.IAM JSON 정책 요소 참조의IAM 사용 설명서.

Actions

관리자는 다음을 사용할 수 있습니다AWS누가 무엇에 액세스 할 수 있는지를 지정하는 JSON 정책입니다. 즉,보안 주체수행 할 수 있습니다작업란 무엇입니까?리소스, 그리고 무엇에조건.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWS API 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함시킵니다.

App Runner의 정책 작업은 작업 앞에 접두사를 사용합니다. `apprunner:`. 예를 들어 Amazon EC2를 사용하여 Amazon EC2 인스턴스를 실행할 수 있는 권한을 부여합니다.`RunInstances`API 작업을 사용하는 경우`ec2:RunInstances`작업을 정책에 포함해야 합니다. 정책 설명에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. App Runner는 이 서비스로 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 작업을 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 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "apprunner:Describe*"
```

앱 러너 작업 목록을 보려면 단원을 참조하십시오.에서 정의한 작업AWS App Runner의서비스 승인 참조.

Resources

관리자는 다음을 사용할 수 있습니다AWS누가 무엇에 액세스 할 수 있는지를 지정하는 JSON 정책입니다. 즉,보안 주체수행 할 수 있습니다작업란 무엇입니까?리소스, 그리고 무엇에조건.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource` 또는 `NotResource` 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을

사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우 와일드카드(*)를 사용하여 명령문이 모든 리소스에 적용됨을 나타냅니다.

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

앱 러너 리소스에는 다음과 같은 ARN 구조가 있습니다.

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

ARL의 형식에 대한 자세한 내용은 단원을 참조하십시오.Amazon 리소스 이름 (ARN) 및AWS서비스 네임스페이스의AWS일반 참조.

예를 들어, 지정하려면my-service서비스에 대한 자세한 내용은 다음 ARN 사용합니다.

```
"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/*"
```

리소스를 생성하기 위한 작업과 같은 일부 App Runner 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우 와일드카드(*)를 사용해야 합니다.

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

App Runner 리소스 유형 및 해당 ARN의 목록을 보려면 단원을 참조하십시오.에서 정의한 리소스 AWS App Runner의서비스 승인 참조. 각 리소스의 ARN 지정할 수 있는 작업을 알아보려면 단원을 참조하십시오.에서 정의한 작업AWS App Runner.

조건 키

관리자는 다음을 사용할 수 있습니다AWS누가 무엇에 액세스 할 수 있는지를 지정하는 JSON 정책입니다. 즉,보안 주체수행 할 수 있습니다작업란 무엇입니까?리소스, 그리고 무엇에조건.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 선택 사항입니다. 같음이나 미만 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 AWS는 논리적 AND 작업을 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 OR 작업을 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 단원을 참조하십시오.IAM 정책 요소: 변수 및 태그의IAM 사용 설명서.

AWS에서는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 항목을 보려면AWS전역 조건 키에 대한 자세한 내용은AWS글로벌 조건 컨텍스트 키의IAM 사용 설명서.

앱 러너는 일부 전역 조건 키를 사용하도록 지원합니다. 모든 항목을 보려면AWS전역 조건 키에 대한 자세한 내용은AWS전역 조건 컨텍스트 키의IAM 사용 설명서.

앱 러너는 서비스별 조건 키 집합을 정의합니다. 또한 App Runner는 조건 키를 사용하여 구현되는 태그 기반 액세스 제어를 지원합니다. 세부 정보는 the section called “앱 러너 태그 기반 권한 부여” 단원을 참조하십시오.

앱 러너 조건 키 목록을 보려면 단원을 참조하십시오.의 조건 키AWS App Runner의서비스 승인 참조. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 단원을 참조하십시오.에서 정의한 작업AWS App Runner.

Examples

App Runner 자격 증명 기반 정책의 예를 보려면 단원을 참조하십시오.앱 러너 자격 증명 기반 정책 예제.

앱 러너 리소스 기반 정책

App Runner는 리소스 기반 정책을 지원하지 않습니다.

앱 러너 태그 기반 권한 부여

앱 러너 리소스에 태그를 연결하거나 요청에서 태그를 앱 러너에 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 리소스 태그 지정에 대한 자세한 내용은 단원을 참조하십시오.the section called “Configuration”.

해당 리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하기 위한 자격 증명 기반 정책의 예제를 보려면 태그를 기반으로 하는 App Runner 서비스에 대한 액세스 제어 단원을 참조하십시오.

앱 러너 사용자 권한

App Runner를 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 단원을 참조하십시오.IAM 사용자에 대한 권한 변경의IAM 사용 설명서.

App Runner는 사용자에게 연결할 수 있는 2개의 관리형 정책을 제공합니다.

- `AWSAppRunnerReadOnlyAccess`— App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있는 권한을 부여합니다.
- `AWSAppRunnerFullAccess`— 모든 앱 러너 작업에 대한 권한을 부여합니다.

사용자 권한을 보다 세부적으로 제어하려면 사용자 지정 정책을 만들어 사용자에게 연결할 수 있습니다. 자세한 내용은 단원을 참조하십시오.IAM 정책 생성의IAM 사용 설명서.

사용자 정책에 대한 예시는 단원을 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",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "apprunner.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "apprunner.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey",
        "kms:CreateGrant"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Administrative permission over AppRunner applications",
      "Effect": "Allow",
      "Action": "apprunner:*",
      "Resource": "*"
    }
  ]
}
```

IAM 앱 러너 역할

한IAM 역할는 내 엔티티입니다.AWS 계정특정 권한을 가지고 있습니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 제품이 다른 서비스의 리소스에 액세스하여 사용자 대신 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

앱 러너는 서비스 연결 역할을 지원합니다. App Runner 서비스 연결 역할을 생성하거나 관리하는 방법에 대한 자세한 내용은 단원을 참조하십시오.the section called “서비스 연결 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 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

앱 러너는 몇 가지 서비스 역할을 지원합니다.

액세스 역할

액세스 역할은 App Runner에서 계정의 Amazon Elastic Container Registry (Amazon ECR) 에 있는 이미지에 액세스하기 위해 사용하는 역할입니다. Amazon ECR의 이미지에 액세스하는 데 필요하며 Amazon ECR 공용에서는 필요하지 않습니다. Amazon ECR의 이미지를 기반으로 서비스를 생성하기 전에 IAM을 사용하여 서비스 역할을 생성하고`AWSAppRunnerServicePolicyForECRAccess`관리형 정책을 포함할 수 있습니다. 그런 다음 호출 할 때이 역할을 App Runner에 전달할 수 있습니다CreateServiceAPI 사용AuthenticationConfiguration구조체의 멤버 (SourceConfiguration매개 변수) 또는 App Runner 콘솔을 사용하여 서비스를 만들 때 사용할 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 생성하는 경우`"Resource": "*"`의 경우`ecr:GetAuthorizationTokenaction`. 토큰은 액세스 권한이 있는 Amazon ECR 레지스트리에 액세스하는 데 사용할 수 있습니다.

액세스 역할을 만들 때 App Runner 서비스 보안 주체를 선언하는 신뢰 정책을 추가해야 합니다.`build.apprunner.amazonaws.com`신뢰할 수 있는 엔터티로.

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 콘솔은 자동으로 액세스 역할을 만들어 새 서비스에 대해 선택할 수 있습니다. 콘솔에는 계정의 다른 역할도 나열되며, 원하는 경우 다른 역할을 선택할 수 있습니다.

인스턴스 역할

인스턴스 역할은 App Runner가 사용 권한을 제공하는 데 사용하는 선택적 역할입니다.AWS응용 프로그램 코드가 호출하는 서비스 동작입니다. App Runner 서비스를 생성하기 전에 IAM을 사용하여 애플리케이션 코드에 필요한 권한을 가진 서비스 역할을 생성합니다. 그런 다음이 역할을 응용 프로그램 러

너에 전달할 수 있습니다CreateServiceAPI를 사용하거나 앱 러너 콘솔을 사용하여 서비스를 만들 때 사용할 수 있습니다.

인스턴스 역할을 만들 때 App Runner 서비스 보안 주체를 선언하는 신뢰 정책을 추가해야 합니다.`tasks.apprunner.amazonaws.com`신뢰할 수 있는 엔터티로.

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 콘솔에 계정의 역할이 나열되며 이 목적을 위해 만든 역할을 선택할 수 있습니다.

서비스 생성에 대한 자세한 내용은 단원을 참조하십시오.the section called “생성”.

> **Note**
>
> 인스턴스 역할이 제공해야 하는 권한은 전적으로 응용 프로그램에 따라 다릅니다. 코드가 아무것도 호출하지 않을 수 있습니다.AWSAPI를 사용할 수 있으며, 이 경우 App Runner에 인스턴스 역할을 제공 할 필요가 없습니다.
> 코드가 호출 할 경우AWSAPI를 사용하면 어떤 호출인지 예측할 방법이 없습니다. 따라서 인스턴스 역할에 대한 관리형 정책은 제공하지 않습니다. 인스턴스 역할에 필요한 권한을 명시적으로 포함하거나 사용자 지정 정책을 생성하여 인스턴스 역할에 사용해야 합니다.

앱 러너 자격 증명 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 생성하거나 수정할 수 있는 권한이 없습니다.AWS App Runner있습니다. 또한 AWS Management Console, AWS CLI 또는 AWS API를 사용해 작업을 수행할 수 없습니

다. 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 자격 증명 기반 정책을 생성하는 방법을 알아보려면 단원을 참조하십시오.JSON 탭에서 정책 만들기의IAM 사용 설명서.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

주제

- 정책 모범 사례
- 사용자 정책
- 태그를 기반으로 하는 App Runner 서비스에 대한 액세스 제어

정책 모범 사례

자격 증명 기반 정책은 매우 강력합니다. 이 정책은 계정에서 사용자가 App Runner 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- 사용 시작하기AWS관리형 정책— 앱 러너 사용을 빠르게 시작하려면AWS관리형 정책을 사용하여 직원에게 필요한 권한을 부여합니다. 이 정책은 이미 계정에서 사용할 수 있으며 AWS에 의해 유지 관리 및 업데이트됩니다. 자세한 내용은 단원을 참조하십시오.에서 권한 사용 시작AWS관리형 정책의IAM 사용 설명서.
- 최소 권한 부여 - 사용자 지정 정책을 생성할 때 작업을 수행하는 데 필요한 권한만 부여합니다. 최소한의 권한 조합으로 시작하여 필요에 따라 추가 권한을 부여합니다. 처음부터 권한을 많이 부여한 후 나중에 줄이는 방법보다 이 방법이 안전합니다. 자세한 내용은 단원을 참조하십시오.최소 권한 부여의IAM 사용 설명서.
- 중요한 작업에 대해 MFA 활성화 - 보안을 강화하기 위해 IAM 사용자가 중요한 리소스 또는 API 작업에 액세스할 때 Multi-Factor Authentication(MFA)을 사용하도록 합니다. 자세한 내용은 단원을 참조하십시오.멀티 팩터 인증 (MFA) 사용AWS의IAM 사용 설명서.
- 보안 강화를 위해 정책 조건 사용 – 실제로 가능한 경우 자격 증명 기반 정책이 리소스에 대한 액세스를 허용하는 조건을 정의합니다. 예를 들어 요청을 할 수 있는 IP 주소의 범위를 지정하도록 조건을 작성할 수 있습니다. 지정된 날짜 또는 시간 범위 내에서만 요청을 허용하거나, SSL 또는 MFA를 사용해야 하는 조건을 작성할 수도 있습니다. 자세한 내용은 단원을 참조하십시오.IAM JSON 정책 요소: Condition의IAM 사용 설명서.

사용자 정책

App Runner 콘솔에 액세스하려면 IAM 사용자에게 최소한의 권한 집합이 있어야 합니다. 이러한 권한은 앱 Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다.AWS 계정. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다.

App Runner는 사용자에게 연결할 수 있는 2개의 관리형 정책을 제공합니다.

- `AWSAppRunnerReadOnlyAccess`— App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있는 권한을 부여합니다.
- `AWSAppRunnerFullAccess`— 모든 앱 러너 작업에 대한 권한을 부여합니다.

사용자가 앱 러너 콘솔을 사용할 수 있도록 하려면 최소한`AWSAppRunnerReadOnlyAccess`관리형 정책을 사용자에 적용합니다. 연결할 수 있습니다.`AWSAppRunnerFullAccess`관리형 정책을 대신 사용하거나 특정 추가 권한을 추가하여 사용자가 리소스를 생성, 수정 및 삭제할 수 있도록 합니다. 자세한 내용은 단원을 참조하십시오.사용자에 권한 추가의IAM 사용 설명서.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 사용자가 수행하도록 허용하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

다음 예는 사용자 지정 사용자 정책을 보여줍니다. 사용자 지정 사용자 정책을 정의하기 위한 시작 지점으로 사용할 수 있습니다. 예제를 복사하거나 작업을 제거하고 리소스의 범위를 좁히고 조건을 추가합니다.

예: 콘솔 및 연결 관리 사용자 정책

이 예제 정책은 콘솔 액세스를 활성화하고 연결 생성 및 관리를 허용합니다. 앱 러너 서비스 생성 및 관리를 허용하지 않습니다. 소스 코드 자산에 대한 App Runner 서비스 액세스를 관리하는 역할을하는 사용자에게 연결할 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 예제에서는 일부 리소스 속성 또는 작업 매개 변수에 종속된 조건부 사용 권한을 보여 줍니다.

이 예제 정책은 App Runner 서비스를 만들 수 있지만prod.

```
{
  "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 다음과 같은 앱 러너 서비스를 업데이트할 수 있도록preprod라는 Auto Scaling 구성을 사용하는 경우에만preprod.

```
{
  "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 하는 App Runner 서비스에 대한 액세스 제어

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 App Runner 리소스에 대한 액세스를 제어할 수 있습니다. 이 예에서는 App Runner 서비스 삭제를 허용하는 정책을 생성할 수 있는 방법을 보여 줍니다. 하지만 `Owner` 서비스 태그가 해당 사용자의 사용자 이름 값을 가지고 있는 경우에만 권한이 부여됩니다. 이 정책은 콘솔에서 이 작업을 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 사용자를`richard-roe`가 App Runner 서비스 삭제를 시도하면 서비스에 태그가 지정되어야 합니다.`Owner=richard-roe`또는`owner=richard-roe`. 그렇지 않으면 액세스가 거부됩니다. 조건 키 이름은 대소문자를 구분하지

않기 때문에 태그 키 `Owner`는 `Owner` 및 `owner` 모두와 일치합니다. 자세한 내용은 단원을 참조하십시오.IAM JSON 정책 요소: Condition의IAM 사용 설명서.

앱 러너에 서비스 연결 역할 사용

AWS App Runner은 다음을 사용합니다.AWS Identity and Access Management(IAM)서비스 연결 역할. 서비스 연결 역할은 App Runner에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 App Runner에서 사전 정의하며 서비스에서 다른 제품을 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.AWS서비스를 사용자 대신 사용합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 App Runner를 더 쉽게 설정할 수 있습니다. App Runner에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한 App Runner만 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 객체에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 App Runner 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 단원을 참조하십시오.AWSIAM과 함께 작업하는 서비스가지고 있는 서비스를 찾아보세요.예의서비스 연결 역할열을 클릭합니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 [예(Yes)] 링크를 선택합니다.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

앱 러너에 대한 서비스 연결 역할 권한

앱 러너는 라는 서비스 연결 역할을 사용합니다.AWS서비스롤포래프러너.

이 역할은 App Runner에서 다음 작업을 수행할 수 있도록 허용합니다.

- Amazon CloudWatch Logs로 로그를 푸시합니다.
- Amazon CloudWatch Events 규칙을 생성하여 Amazon Elastic Container Registry (Amazon ECR) 이미지 푸시를 구독합니다.

AWSServiceRoleForAppRunner 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `apprunner.amazonaws.com`

AWSServiceRoleForAppRunner 서비스 연결 역할의 권한 정책에는 앱 러너가 사용자를 대신해 작업을 완료하는 데 필요한 모든 권한이 포함되어 있습니다.

AppRunnerServiceRolePolicy

```
{
  "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:*"
      ]
    },
    {
      "Sid": "AllowPutRuleForManagedRules",
      "Effect": "Allow",
      "Action": [
        "events: PutRule",
        "events: PutTargets",
        "events: DeleteRule",
        "events: RemoveTargets",
        "events: DisableRule",
        "events: EnableRule"
      ],
      "Resource": "arn:aws:events:*:*:rule/apprunner-*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "apprunner.amazonaws.com",
          "events:source": "aws.ecr",
          "events:detail-type": "ECR Image Action"
        }
      }
    }
```

```
  ]
}
```

IAM 개체(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

앱 러너에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. 응용 프로그램 러너 서비스를 만들 때AWS Management Console,AWS CLI, 또는AWSAPI를 사용하면 App Runner가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 앱 러너 서비스를 생성할 때 앱 러너는 서비스 연결 역할을 다시 생성합니다.

앱 러너에 대한 서비스 연결 역할 편집

앱 러너는 AWSServiceRoleForAppRunner 서비스 연결 역할을 편집할 수 없습니다. 서비스 연결 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.

앱 러너에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 이렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

앱 러너에서 이는 계정의 모든 앱 러너 서비스 삭제를 의미합니다. 앱 러너 서비스를 삭제하는 방법에 대한 자세한 내용은the section called “삭제”.

> **Note**
>
> 리소스를 삭제하려 할 때 App Runner 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

IAM을 사용하여 서비스 연결 역할을 수동으로 삭제하려면

IAM 콘솔을 사용하여AWS CLI, 또는AWSAPI를 사용하여 AWSServiceRoleForAppRunner 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

앱 러너 서비스 연결 역할이 지원되는 리전

App Runner는 서비스를 사용할 수 있는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 단원을 참조하십시오.AWS App Runner엔드포인트 및 할당량의AWS일반 참조.

AWS App Runner의 AWS 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 보다 쉽게AWS관리형 정책을 직접 작성하는 것보다 시간과 전문 지식이 필요합니다.IAM 고객 관리형 정책을 생성합니다.팀에게 필요한 권한만을 제공할 수 있는 서비스입니다. 빠르게 시작하려면 당사의AWS관리형 정책 이러한 정책은 일반적인 사용 사례를 다루며AWS계정에 로그인합니다. 에 대한 자세한 내용AWS관리형 정책에 대한 자세한 내용은AWS관리형 정책의IAM 사용 설명서.

AWS서비스 유지 관리 및 업데이트AWS관리형 정책 에서 권한을 변경할 수 없습니다.AWS관리형 정책 서비스는 때때로 추가 사용 권한을AWS관리형 정책을 사용하여 새로운 기능을 지원할 수 있습니다. 이 유형의 업데이트는 정책이 연결된 모든 자격 증명 (사용자, 그룹 및 역할) 에도 적용됩니다. 서비스는 업데이트 할 가능성이 가장 높습니다AWS관리형 정책은 새 기능이 시작되거나 새 작업을 사용할 수 있게 될 때 적용됩니다. 서비스에서 사용 권한을 제거 하지 않는AWS관리형 정책을 사용할 수 있으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한,AWS는 여러 서비스에 걸쳐 있는 직무에 관한 관리형 정책을 지원합니다. 예를 들어,ReadOnlyAccess AWS관리형 정책은 모든AWS서비스 및 리소스를 참조하십시오. 서비스에서 새 기능을 시작하면AWS는 새 작업 및 리소스에 대한 읽기 전용 권한을 추가합니다. 직무 정책의 목록과 설명은 단원을 참조하십시오.AWS직무에 관한 관리형 정책의IAM 사용 설명서.

앱 러너 업데이트AWS관리형 정책

업데이트에 대한 세부 정보를 봅니다.AWS이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 App Runner에 대한 정책을 관리합니다. 이 페이지의 변경 사항에 대한 자동 경보를 보려면 App Runner 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
앱 실행자서비스 역할정책— 새 정책	앱 러너는 앱 러너 서비스를 대신하여 Amazon CloudWatch 로그 및 Amazon CloudWatch Events 호출할 수 있도록 새로운 정책을 추가했습니다.	2021년 3월 1일
AWS앱 실행읽기 전용액세스— 새 정책	App Runner는 사용자가 App Runner 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 하는 새 정책을 추가했습니다.	2021년 3월 1일
AWS앱런너풀액세스— 새 정책	App Runner는 사용자가 모든 앱 러너 작업을 수행할 수 있도록 하는 새 정책을 추가했습니다.	2021년 3월 1일
AWS앱 실행자서비스 정책예측 액세스— 새 정책	App Runner가 계정의 Amazon Elastic Container Registry (Amazon ECR) 이미지에 액세스하도록 하는 새 정책을 추가했습니다.	2021년 3월 1일
앱 러너가 변경 내용 추적을 시작했습니다.	앱 러너는 변경 사항을 추적하기 시작했습니다.AWS관리형 정책	2021년 3월 1일

앱 러너 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 사용 시 공통적으로 발생할 수 있는 문제를 진단 및 수정하십시오.AWS App Runner및 IAM.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

주제

- 앱 러너에서 작업을 수행할 권한이 없음
- 관리자이며 다른 사용자가 App Runner에 액세스하도록 허용하려고 함
- 나는 외부의 사람을 허용하려고 내AWS 계정내 앱 러너 리소스에 액세스

앱 러너에서 작업을 수행할 권한이 없음

AWS Management Console에서 작업을 수행할 권한이 없다는 메시지가 나타나는 경우 관리자에게 지원을 요청하십시오. 관리자는 귀하를 제공한 사람입니다.AWS사용자 이름 및 암호를 입력합니다.

다음 예제 오류는 라는 IAM 사용자가`marymajor`콘솔을 사용하여 App Runner 서비스에 대한 세부 정보를 보려고 하지만`apprunner:DescribeService`권한.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

이 경우 Mary는 관리형 정책을 업데이트하도록 요청하여*my-example-service*리소스를 사용하여`apprunner:DescribeService`action.

관리자이며 다른 사용자가 App Runner에 액세스하도록 허용하려고 함

다른 사용자가 App Runner에 액세스하도록 하려면 액세스 권한이 필요한 사용자 또는 애플리케이션에 대한 IAM 개체 (사용자 또는 역할) 를 생성해야 합니다. 다른 사용자들은 해당 엔터티에 대한 자격 증명을 사용해 AWS에 액세스합니다. 그런 다음 App Runner에서 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다.

바로 시작하려면 단원을 참조하십시오.IAM 위임 사용자와 그룹 처음 생성의IAM 사용 설명서.

나는 외부의 사람을 허용하려고 내AWS 계정내 앱 러너 리소스에 액세스

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스하는 데 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 ACL(액세스 제어 목록)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- App Runner에서 이러한 기능을 지원하는지 여부를 알아보려면IAM과 앱 러너가 작동하는 방식.
- 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면AWS소유하고 있는 계정에 대한 자세한 내용은IAM 사용자에게 액세스 권한 제공AWS소유한 계정의IAM 사용 설명서.
- 리소스에 대한 액세스 권한을 타사에 제공하는 방법을 알아보려면AWS계정에 대한 자세한 내용은에 대한 액세스 권한 제공AWS타사가 소유한 계정의IAM 사용 설명서.
- ID 페더레이션을 통해 액세스를 제공하는 방법을 알아보려면외부에서 인증된 사용자에게 액세스 권한 제공 (자격 증명 연동)의IAM 사용 설명서.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 단원을 참조하십시오.IAM 역할이 리소스 기반 정책과 어떻게 다른지의IAM 사용 설명서.

앱 러너에서 로깅 및 모니터링

모니터링은 다음과 같이 안정성, 가용성 및 성능을 유지 관리하는 중요한 부분입니다.AWS App Runner서비스를 제공합니다. 모든 부분에서 모니터링 데이터 수집AWS솔루션을 사용하면 오류가 발생할 경우 이를 더 간편하게 디버깅할 수 있습니다. 앱 러너는 여러AWS도구를 사용하면 App Runner 서비스를 모니터링하고 잠재적 인시던트에 대응할 수 있습니다.

Amazon CloudWatch 경보

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 서비스 지표를 감시할 수 있습니다. 지표가 지정된 기간 수에 대해 지정된 임계값을 초과하면 알림이 전송됩니다.

App Runner는 서비스 전체에 대한 다양한 메트릭과 웹 서비스를 실행하는 인스턴스 (확장 단위) 를 수집합니다. 자세한 내용은 지표 (CloudWatch) 단원을 참조하세요.

애플리케이션 로그

앱 러너는 애플리케이션 코드의 출력을 수집하여 Amazon CloudWatch Logs 로 스트리밍합니다. 이 출력의 내용은 당신에게 달려 있습니다. 예를 들어 웹 서비스에 대한 요청의 세부 레코드를 포함할 수 있습니다. 이러한 로그 레코드는 보안 및 액세스 감사에 유용할 수 있습니다. 자세한 내용은 로그 (CloudWatch Logs) 단원을 참조하세요.

AWS CloudTrail작업 로그

앱 러너와 통합되어AWS CloudTrail은 사용자, 역할 또는AWS앱 러너의 서비스. CloudTrail 은 앱 러너에 대한 API 호출을 모두 이벤트로 캡처합니다. CloudTrail 콘솔에서 가장 최근 이벤트를 볼 수 있으며, Simple Storage Service (Amazon S3) 버킷에 CloudTrail 이벤트를 지속적으로 전송할 수 있도록 추적을 생성할 수 있습니다. 자세한 내용은 API 작업 (CloudTrail) 단원을 참조하세요.

앱 러너에 대한 규정 준수 확인

타사 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 AWS App Runner의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

앱 러너 또는 기타AWS서비스가 특정 규정 준수 프로그램의 범위에 있는 경우AWS규정 준수 프로그램 제공 범위 내 서비스. 일반적인 내용은 AWS 규정 준수 프로그램을 참조하십시오.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하십시오.

AWS 서비스를 사용할 때 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS는 규정 준수를 지원하기 위해 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 기본 환경을 배포하기 위한 단계를 제공합니다.AWS보안 및 규정 준수에 중점을 두고 있습니다.
- HIPAA Security 및 규정 준수 확인— 이 백서에서는 회사가AWS를 사용해 HIPAA 규정 준수 애플리케이션을 생성할 수 있습니다.

> **Note**
>
> 일부 서비스는 HIPAA를 준수하지 않습니다.

- AWS규정 준수 리소스이 워크북 및 안내서 모음은 사용자의 업계와 위치에 적용될 수 있습니다.
- 규칙을 사용하여 리소스 평가의AWS Config개발자 안내서—AWS Config리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다.
- AWS Security Hub— 이AWS의 보안 상태에 대한 포괄적인 보기를 제공합니다.AWS이를 통해 보안 업계 표준 및 모범 사례를 준수하는지 확인할 수 있습니다.
- AWS Audit Manager— 이AWS서비스를 통해 지속적으로AWS를 사용하여 리스크를 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있습니다.

기타 앱 러너 보안 주제는보안.

앱 러너의 복원성

이AWS글로벌 인프라는AWS 리전및 가용 영역을 제공합니다.AWS 리전는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

에 대한 자세한 내용AWS 리전및 가용 영역에 대한 자세한 내용은AWS글로벌 인프라.

AWS App Runner은 사용자를 대신하여 AWS 글로벌 인프라 사용을 관리하고 자동화합니다. App Runner를 사용하면 제공의 가용성 및 내결함성 메커니즘의 이점을 누릴 수 있습니다.AWS제안입니다.

기타 앱 러너 보안 주제는보안.

AWS App Runner의 인프라 보안

관리형 서비스로서AWS App Runner에 의해 보호되는AWS전역 네트워크 보안 프로시저는Amazon Web Services 에 대하여 보안 프로세스 개요백서

다음을 사용합니다.AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 App Runner에 액세스합니다. 클라이언트가 TLS(전송 계층 보안) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 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 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

기타 App Runner 보안 주제는보안.

앱 러너의 구성 및 취약성 분석

AWS와 고객은 높은 수준의 소프트웨어 구성 요소 보안 및 규정 준수를 달성할 책임을 공유합니다. 자세한 내용은 AWS 공동 책임 모델을 참조하십시오.

기타 앱 러너 보안 주제는 단원을 참조하십시오.보안.

앱 Runner 보안 모범 사례

AWS App Runner은 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 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.

기타 앱 Runner보안.

예방 보안 모범 사례

예방적 보안 통제는 사고가 발생하기 전에 사고를 예방하려고 시도합니다.

최소 권한 액세스 구현

앱 RunnerAWS Identity and Access Management(IAM) 관리형 정책IAM 사용자및액세스 역할. 이러한 관리형 정책은 App Runner 서비스를 올바르게 작동하는 데 필요할 수 있는 모든 권한을 지정합니다.

애플리케이션에 우리의 관리형 정책의 모든 권한이 필요한 것은 아닙니다. 이러한 정책을 사용자 지정하여 사용자 및 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 있어서 중요한 부분입니다.AWS모니터링하는 데 도움이 되는 몇 가지 도구와 서비스를 제공합니다.AWS서비스.

다음은 모니터링 대상 항목의 예입니다:

- 앱 실행자에 대한 Amazon CloudWatch 지표— 주요 App Runner 지표와 애플리케이션의 사용자 지정 지표에 대한 경보를 설정합니다. 세부 정보는 지표 (CloudWatch) 단원을 참조하십시오.
- AWS CloudTrail항목— 가용성에 영향을 줄 수 있는 작업 추적`PauseService`또는`DeleteConnection`. 세부 정보는 API 작업 (CloudTrail) 단원을 참조하십시오.

AWS용어집

최신AWS용어에 대한 자세한 내용은AWS용어집의AWS일반 참조.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.