

App Runner

!!CodeBuild, ECR 세팅이 선행되어야합니다!!

APP RUNNER 서비스의 소스 및 배포 설정을 진행합니다.

소스 및 배포

소스

리포지토리 유형

☒ 컨테이너 레지스트리
컨테이너 레지스트리에 저장된 컨테이너 이미지를 사용하여 서비스를 배포합니다.

☐ 소스 코드 리포지토리
소스 리포지토리에 호스팅된 코드를 사용하여 서비스를 배포합니다.

공급자

☒ Amazon ECR
Amazon ECR에 저장된 컨테이너 이미지.

☐ Amazon ECR 퍼블릭
컨테이너 이미지는 공공영역, 오픈 소스 프로젝트 및 커뮤나티 개발자가 공개적으로 공유합니다.

컨테이너 이미지 URI

역제스할 수 있는 이미지에 대한 URI를 입력하거나, Amazon ECR 계정의 이미지를 찾아주세요.
725511698765.dkr.ecr.ap-northeast-2.amazonaws.com/pet-web:latest

찾아보기

배포 설정

배포 트리거

☐ 수동
App Runner 콘솔 또는 AWS CLI를 사용하여 각 배포를 직접 시작합니다.

☒ 자동
App Runner는 레지스트리를 모니터링하고 각 이미지 푸시에 서비스의 새 버전을 배포합니다.

ECR 액세스 역할

이 역할은 ECR에 액세스할 수 있는 App Runner 권한을 부여합니다. 사용자 지정 역할을 생성하려면 IAM 콘솔 [여기](#)로 이동합니다.

☐ 새 서비스 역할 생성

☒ 기존 서비스 역할 사용

계정에서 IAM 역할을 선택합니다. 목적에는 신뢰할 수 있는 역할만 기재됩니다.

AppRunnerECRAccessRole

서비스 설정

서비스 이름

pet-web

가용 CPU

1 vCPU

가용 메모리

2 GB

환경 변수 - 선택 사항

환경 변수를 위한 키/값 쌍을 추가하거나 Secrets Manager 또는 SSM Parameter Store에서 참조합니다. 이렇게 제공된 IAM 정책 템플릿으로 IAM 정책을 업데이트하여 비밀 및 구성을 환경 변수로 안전하게 참조합니다.
환경 변수가 구성되지 않았습니다.

환경 변수 추가

최대 50개 항목을 추가할 수 있습니다.

IAM 정책 템플릿

포트

서비스가 이 TCP 포트를 사용합니다.
8080

추가 구성

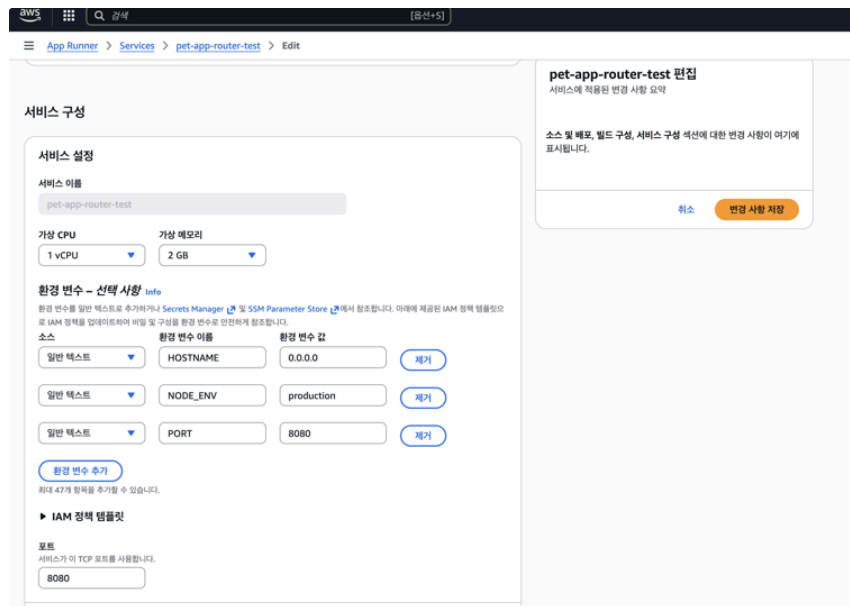
Nextjs standalone 설정으로 인한 포트 설정

왜 하는가?

- standalone을 해야 이미지 크기가 줄어든다 그래야 배포 시간이 단축됨.
- 근데 standalone을 하면 원래 자동으로 포트가 변경되는게 안됨. 그래서 아래와 같이 강제로 포트 지정을 해야함.

1. app runner에 환경변수 넣기

- a. app runner > 배포요소 선택 > 구성 > 편집버튼클릭 > 서비스 구성 > 아래와같이 포트 설정



2. server.js

```

1  /* eslint-disable @typescript-eslint/no-require-imports */
2  // server.js
3  const { createServer } = require('http')
4  const { parse } = require('url')
5  const next = require('next')
6
7  const dev = process.env.NODE_ENV !== 'production'
8  const hostname = process.env.HOSTNAME || '0.0.0.0'
9  const port = parseInt(process.env.PORT, 10) || 8080
10
11 const app = next({ dev, hostname, port })
12 const handle = app.getRequestHandler()
13
14 app.prepare().then(() => {
15   createServer(async (req, res) => {
16     try {
17       const parsedUrl = parse(req.url, true)
18       await handle(req, res, parsedUrl)
19     } catch (err) {
20       console.error('Error occurred handling', req.url, err)
21       res.statusCode = 500
22       res.end('internal server error')
23     }
24   })
25   .once('error', (err) => {
26     console.error(err)
27     process.exit(1)
28   })
29   .listen(port, hostname, () => {
30     console.log(`Ready on http://${hostname}:${port}`)
31   })
32 })
33 
```

3. apprunner.yml

```

1  version: 1.0
2  runtime: nodejs20
3  build:
4    commands:
5      build:
6        - npm ci --cache .npm --prefer-offline
7        - npm run build
8  run:
9    runtime-version: 20

```

```
10  command: node server.js
11  network:
12    port: 8080
13    env:
14      - name: PORT
15        value: "8080"
16      - name: HOSTNAME
17        value: "0.0.0.0"
18      - name: NODE_ENV
19        value: "production"
20
```