

App Runner

!!CodeBuild, ECR 세팅이 선행되어야합니다!!

The screenshot shows the AWS App Runner service configuration interface. It is divided into two main sections: '소스 및 배포' (Deployment Source and Configuration) and '서비스 구성' (Service Configuration).

소스 및 배포

- 소스 (Deployment Source):**
 - 리포지토리 유형 (Repository Type):** 선택한 항목은 'Amazon ECR'입니다.
 - Amazon ECR:** 선택된 ECR에 저장된 컨테이너 이미지를 사용하여 서비스를 배포합니다.
 - 컨테이너 이미지 URL (Container Image URL):** 서비스 할 수 있는 이미지에 대한 URL을 입력하거나, Amazon ECR 계정의 이미지를 찾아보세요.
725511698765.dkr.ecr.ap-northeast-2.amazonaws.com/pet-web:latest
 - 찾아보기 (Search):** 버튼
- 배포 설정 (Deployment Configuration):**
 - 배포 트리거 (Deployment Triggers):** 선택한 항목은 '자동'입니다.
 - ECR 액세스 역할 (ECR Access Role):** 선택한 항목은 '기존 서비스 IAM 사용'입니다.
 - 기본 서비스 IAM 역할 (Default Service IAM Role):** 선택한 항목은 'AppRunnerECRAccessRole'입니다.

서비스 구성 (Service Configuration)

- 서비스 설정 (Service Settings):**
 - 서비스 이름 (Service Name):** 선택한 항목은 'pet-web'입니다.
 - 기상 CPU (Memory Type):** 선택한 항목은 '1 vCPU'입니다.
 - 기상 메모리 (Memory Size):** 선택한 항목은 '2 GB'입니다.
- 환경 변수 - 선택 사항 (Environment Variables - Optional):**
 - 환경 변수 추가 (Add Environment Variable):** 선택한 항목은 '환경 변수 추가'입니다.
 - 설명 (Description):** 최대 50자로 설명을 추가할 수 있습니다.
 - ▶ IAM 정책 템플릿 (IAM Policy Template):** 선택한 항목은 'AWS Lambda'입니다.
 - 포트 (Port):** 선택한 항목은 '8080'입니다.
- 추가 구성 (Additional Configuration):** 선택한 항목은 '추가 구성'입니다.

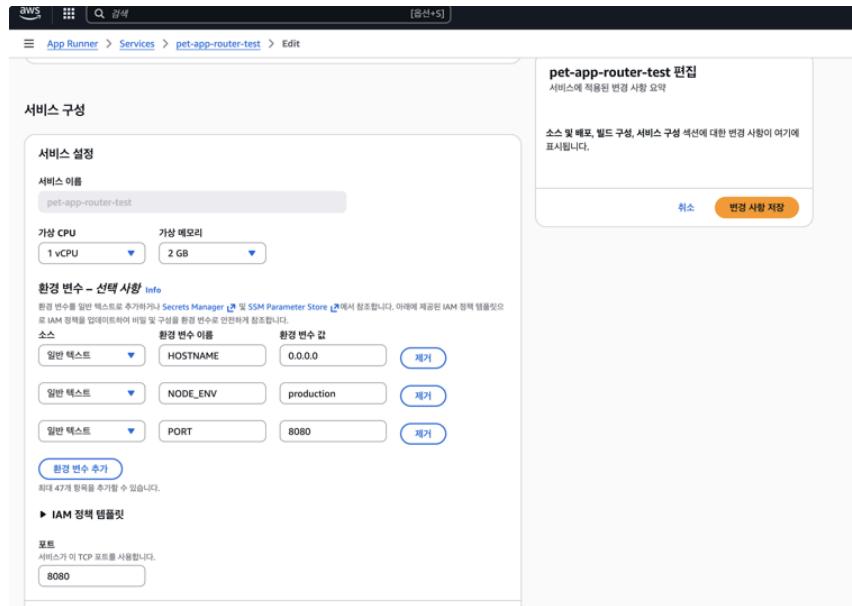
Nextjs standalone 설정으로 인한 포트 설정

왜 하는가?

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

1. app runner에 환경변수 넣기

a. app ruuner > 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
```