

도커 환경의 MYSQL부분 정리하기

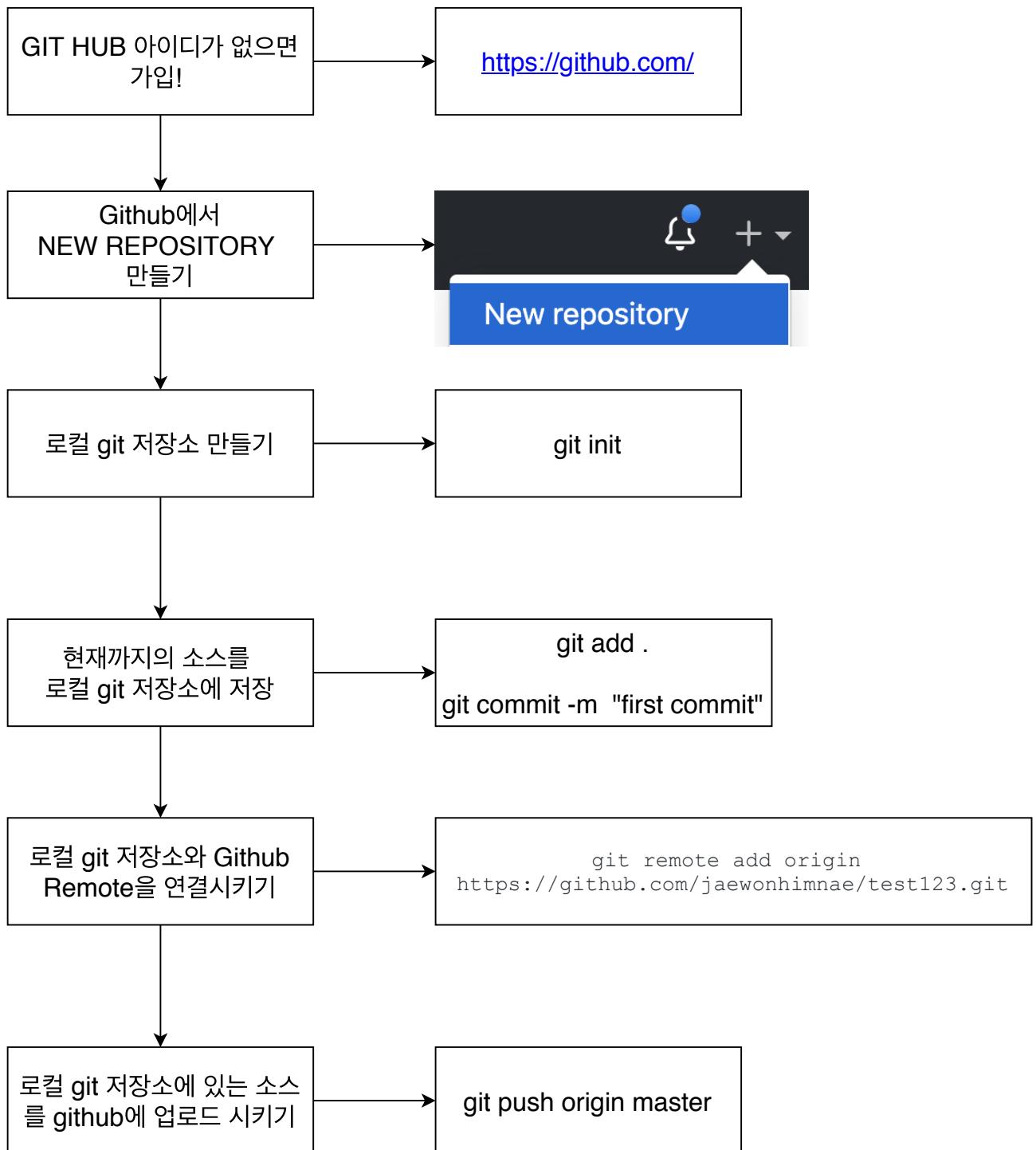
docker-compose.yml
에서 mysql부분
지워주기

```
# mysql:  
#   build: ./mysql  
#   restart: unless-stopped  
#   container_name: app_mysql  
#   ports:  
#     - "3306:3306"  
#   volumes:  
#     - ./mysql/mysql_data:/var/lib/mysql  
#     - ./mysql/sqls:/docker-entrypoint-initdb.d/  
#   environment:  
#     MYSQL_ROOT_PASSWORD: johnahn  
#     MYSQL_DATABASE: myapp
```

mysql 폴더를 지워주기
(하지만 이미 docker-
compose.yml에서 mysql
부분을 지워줬기에 안지워
도 작동은 안된다.)

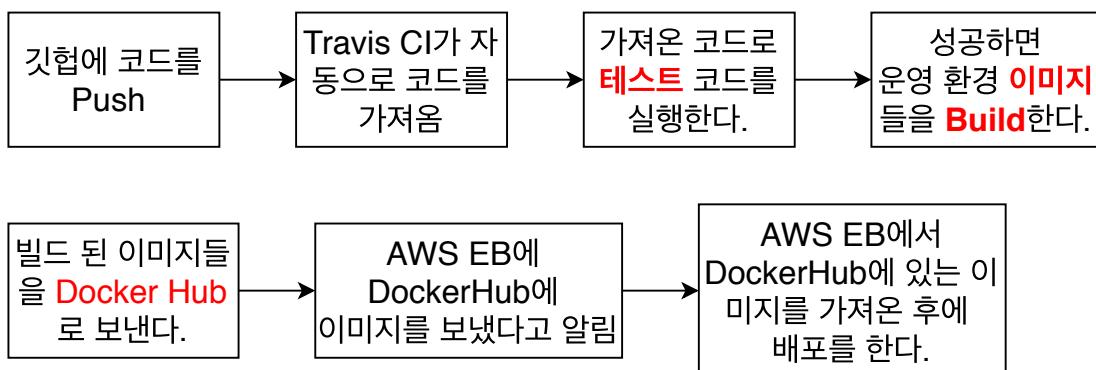
Github에 소스 코드 올리기

Steps

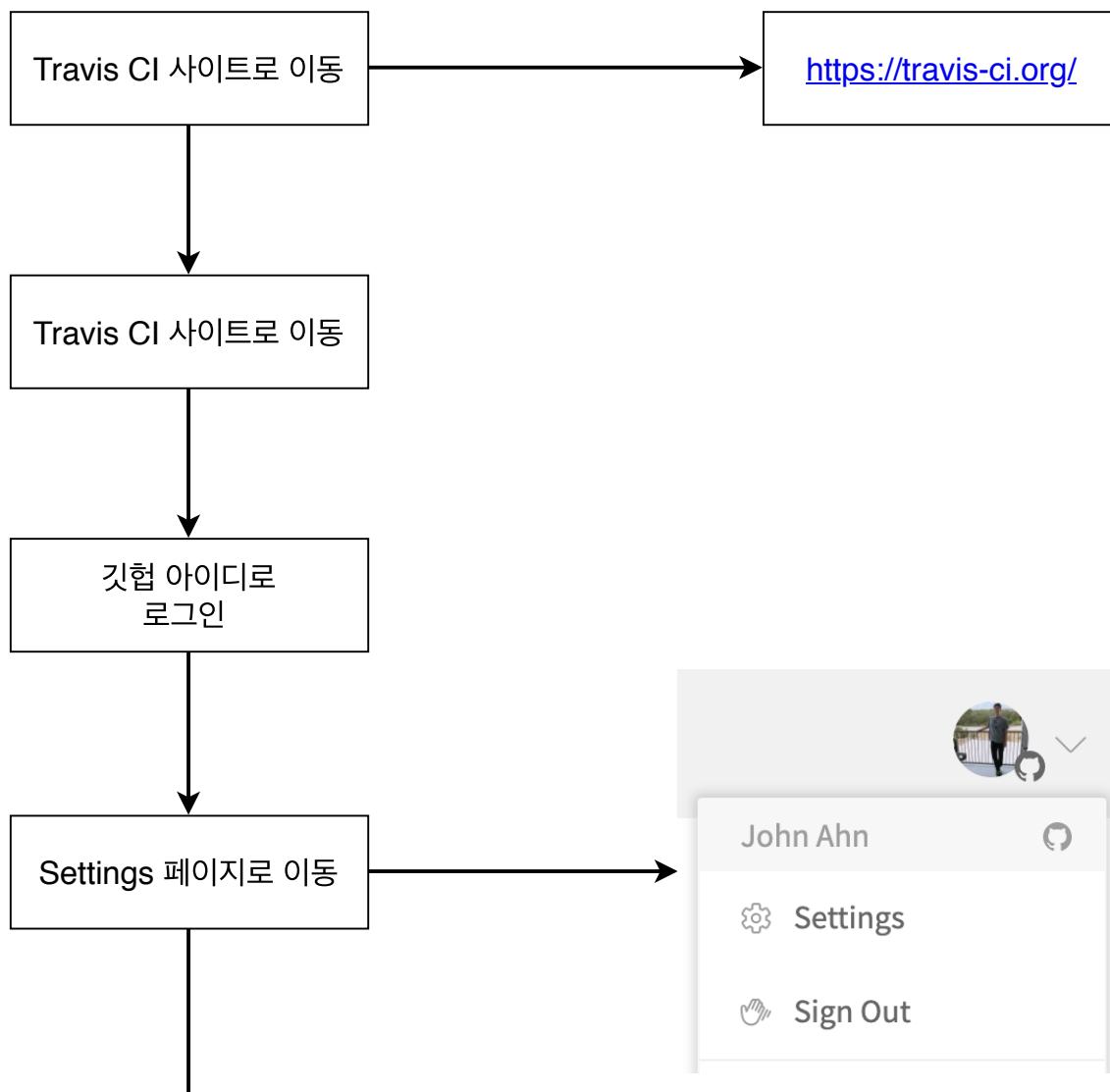


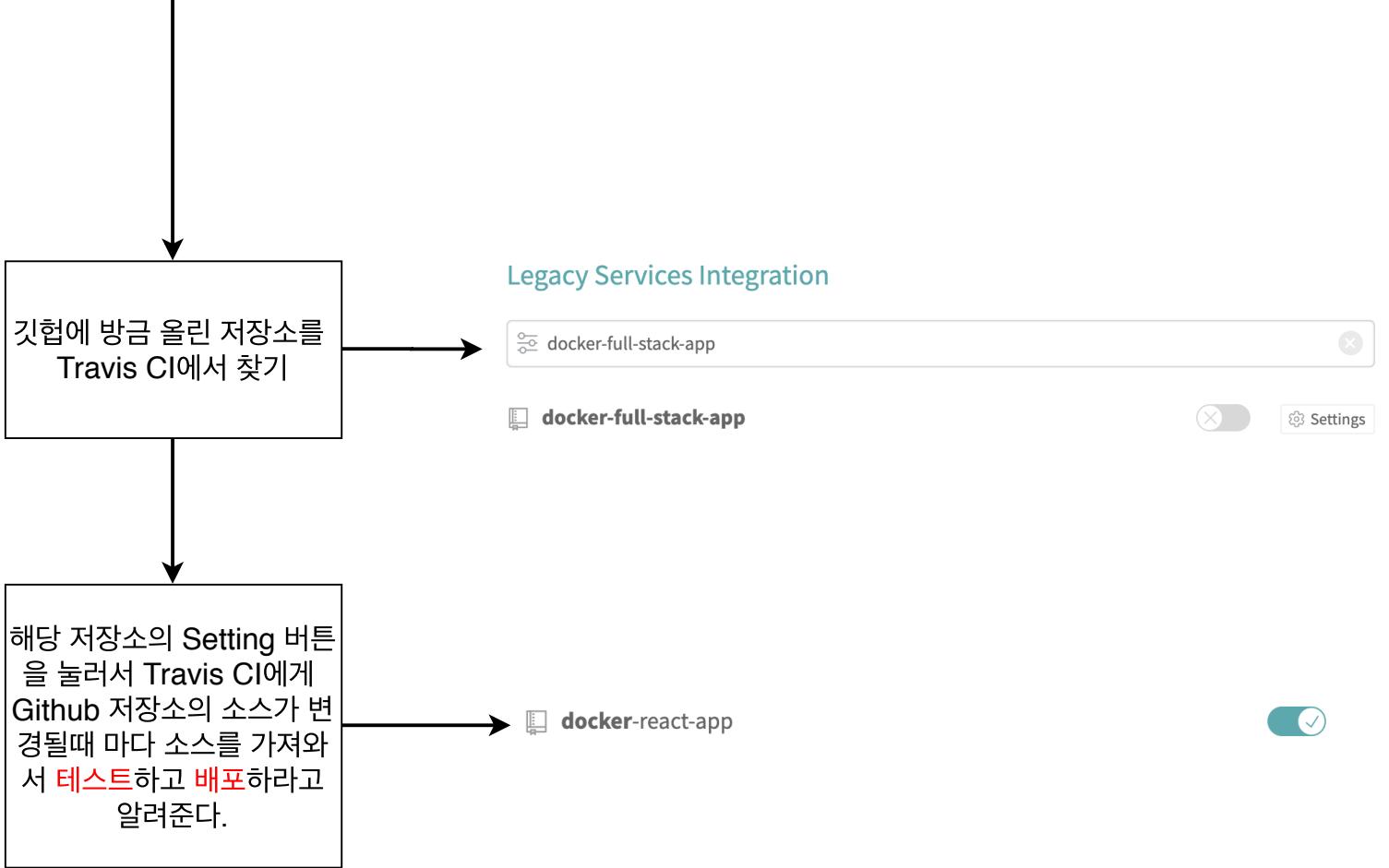
Travis CI Steps

Travis CI 할것들 Steps



순서

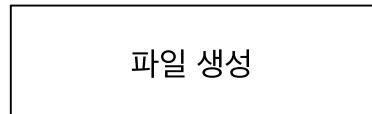




travis.yml 파일 작성(이미지 빌드까지)

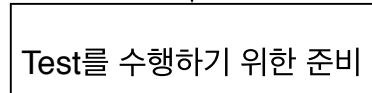
어떻게 작성하나요?

간단하게



구체적으로

.travis.yml

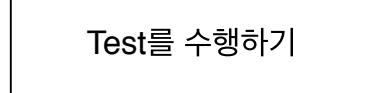


1

앱을 도커 환경에서
실행하고 있으니
Travis Ci에게
도커 환경으로 만들 것이라고
선언 해주기

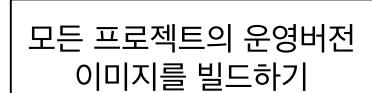
2

구성된 도커 환경에서
Dockerfile.dev를 이용해서
도커 이미지 생성



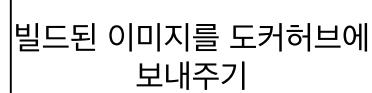
3

생성된 테스트 이미지를 이
용해서 테스트 수행하기



4

테스트가 성공했다면
하나하나의 프로젝트의
운영버전 이미지를 빌드하는
설정을 해주기.



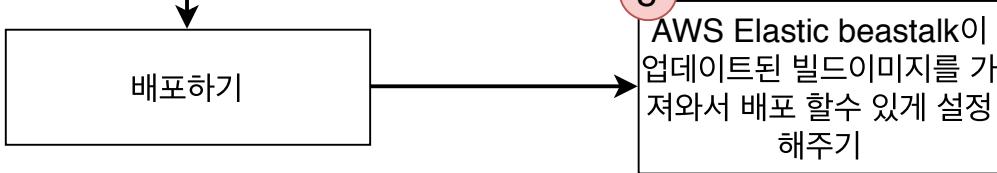
5

도커 허브에 빌드 된 파일을
넣어주기 위해서 도커 허브
에 로그인하기

6

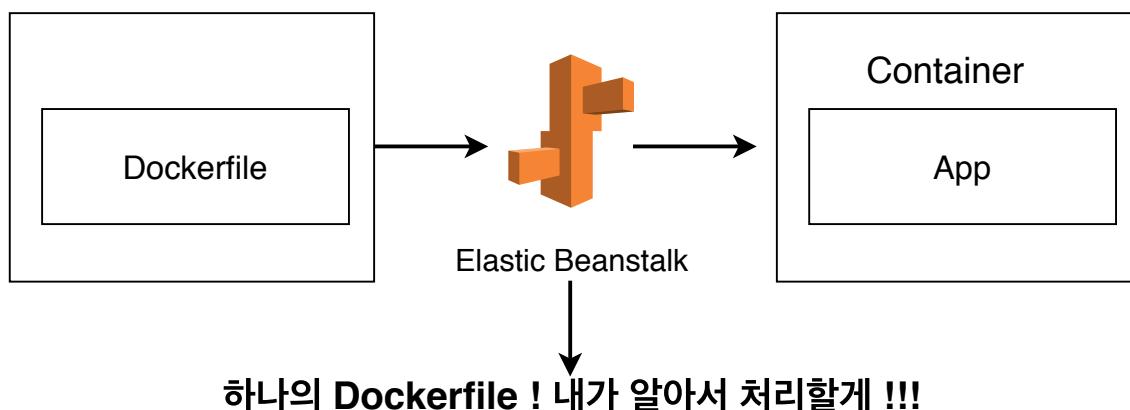
빌드된 이미지를 도커허브에
보내주기

8

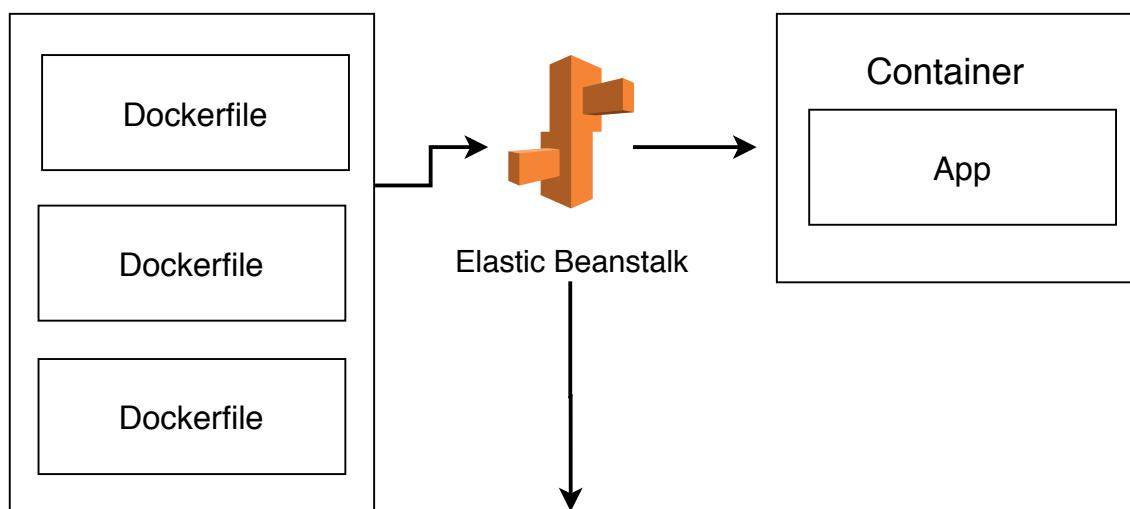


Dockerrun.aws.json에 대해서

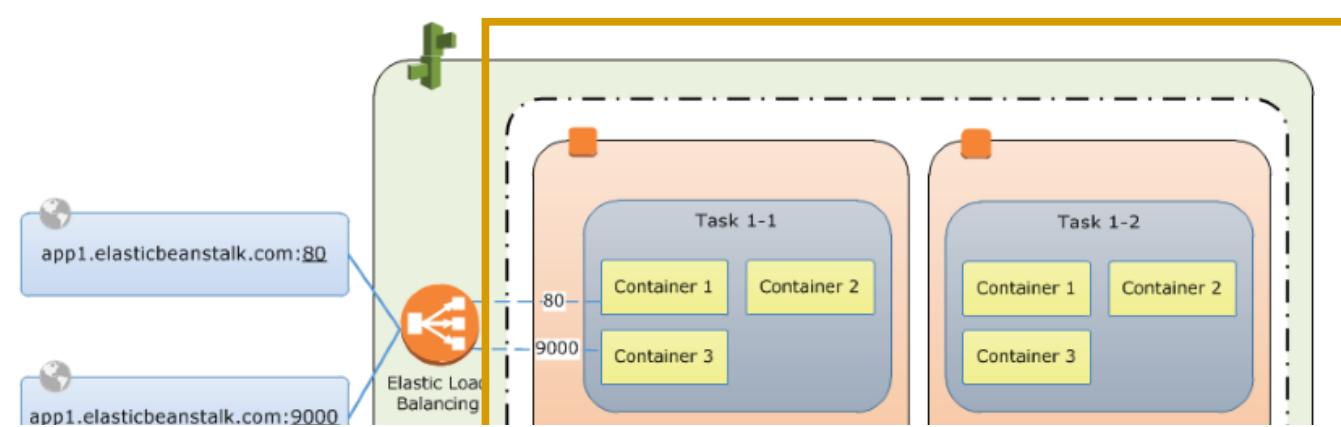
Dockerfile 하나 (React App)

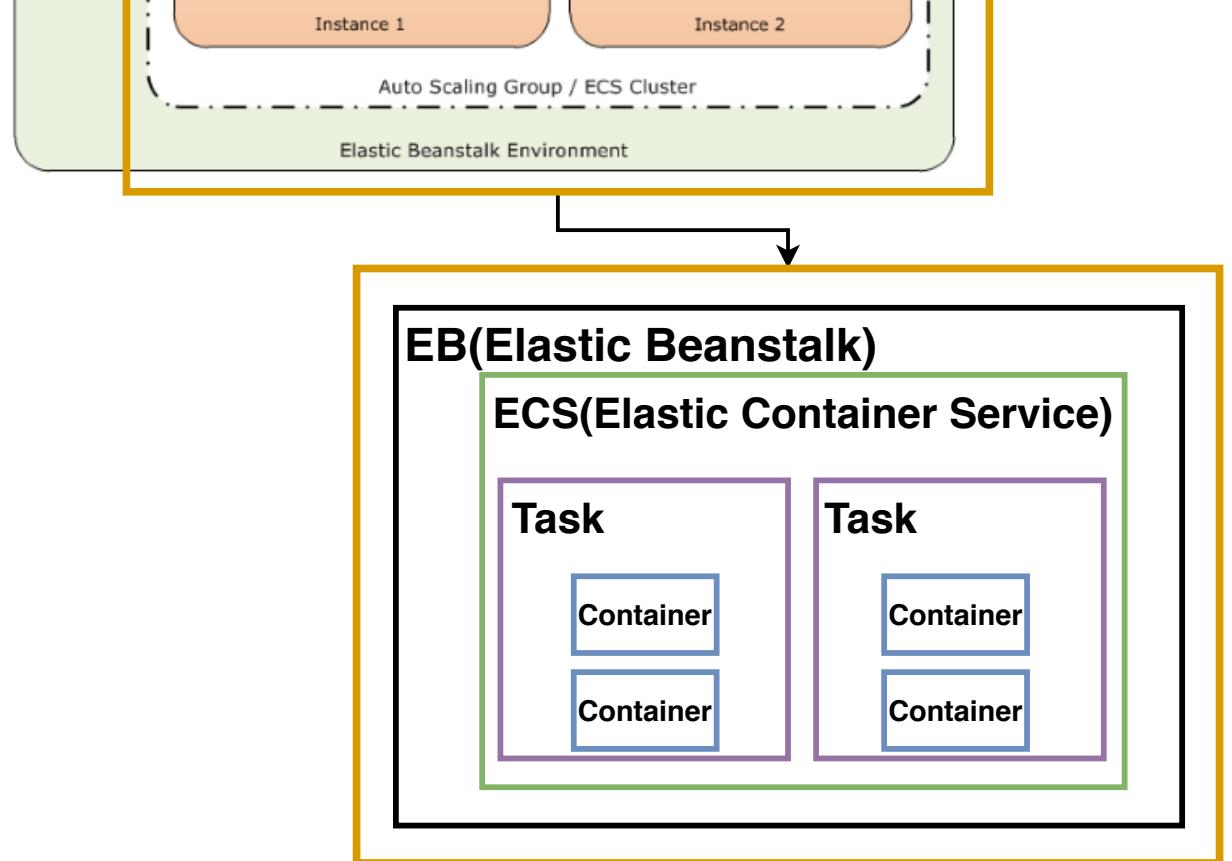


Dockerfile 여러개 (Full Stack App)

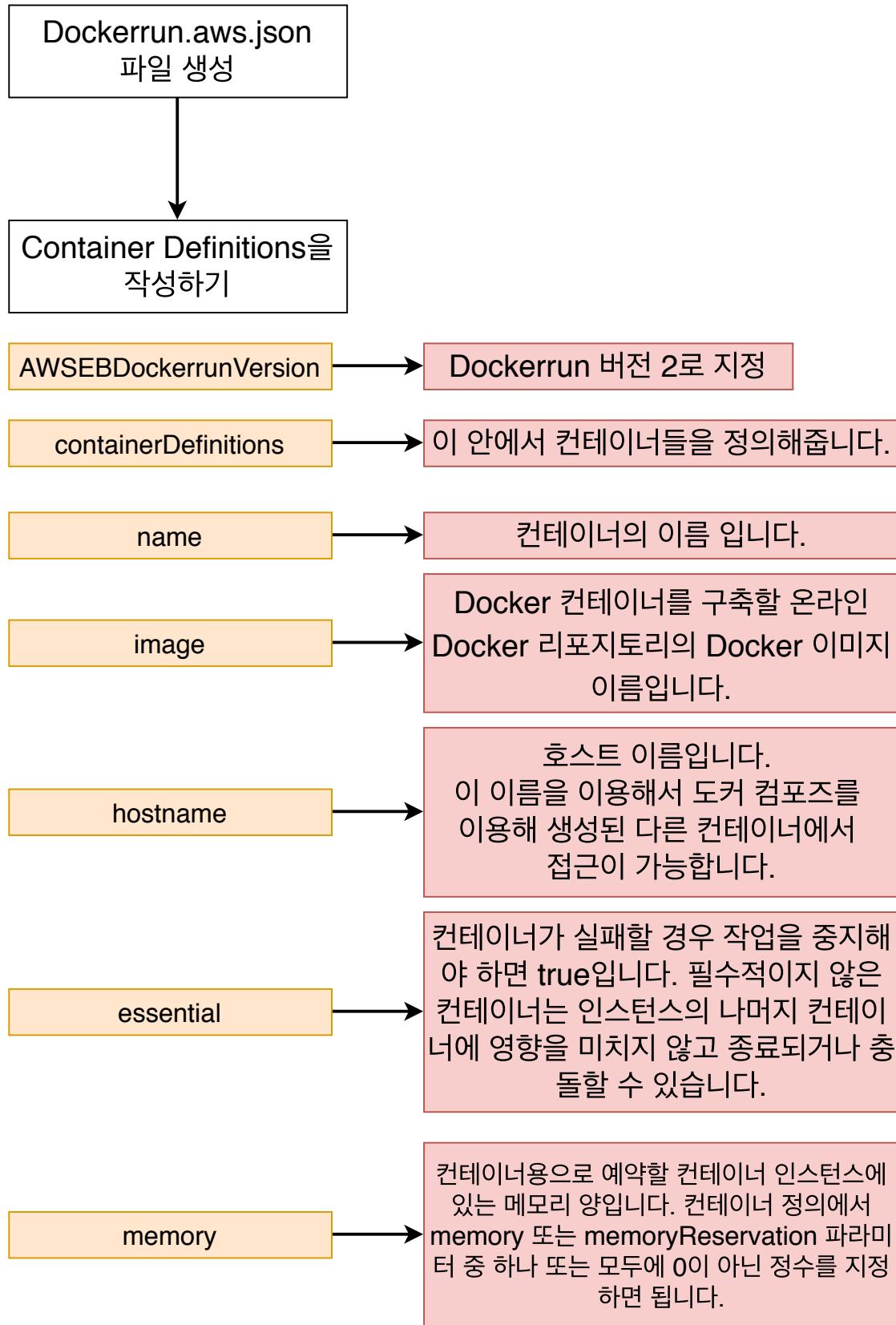


AWS 제공하는 도표로 좀 더 깊게 이해하기





Dockerrun.aws.json 작성하기



portMappings

컨테이너에 있는 네트워크 지점을 호스트에 있는 지점에 매핑합니다.

links

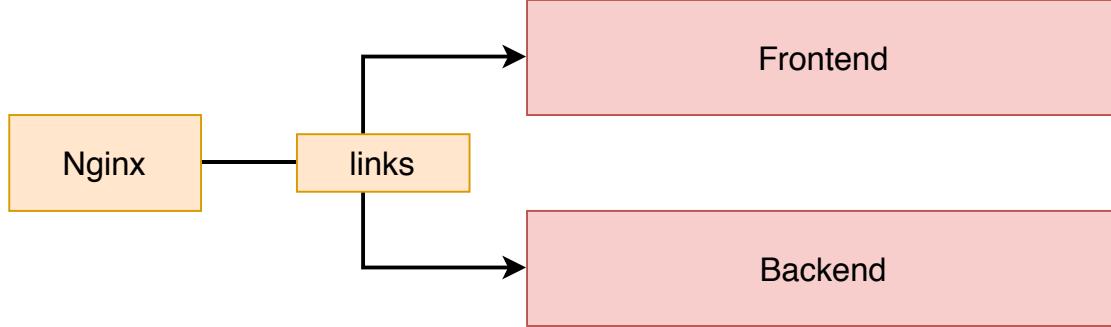
연결할 컨테이너의 목록입니다.
연결된 컨테이너는 서로를 검색하고
안전하게 통신할 수 있습니다.

Nginx

links

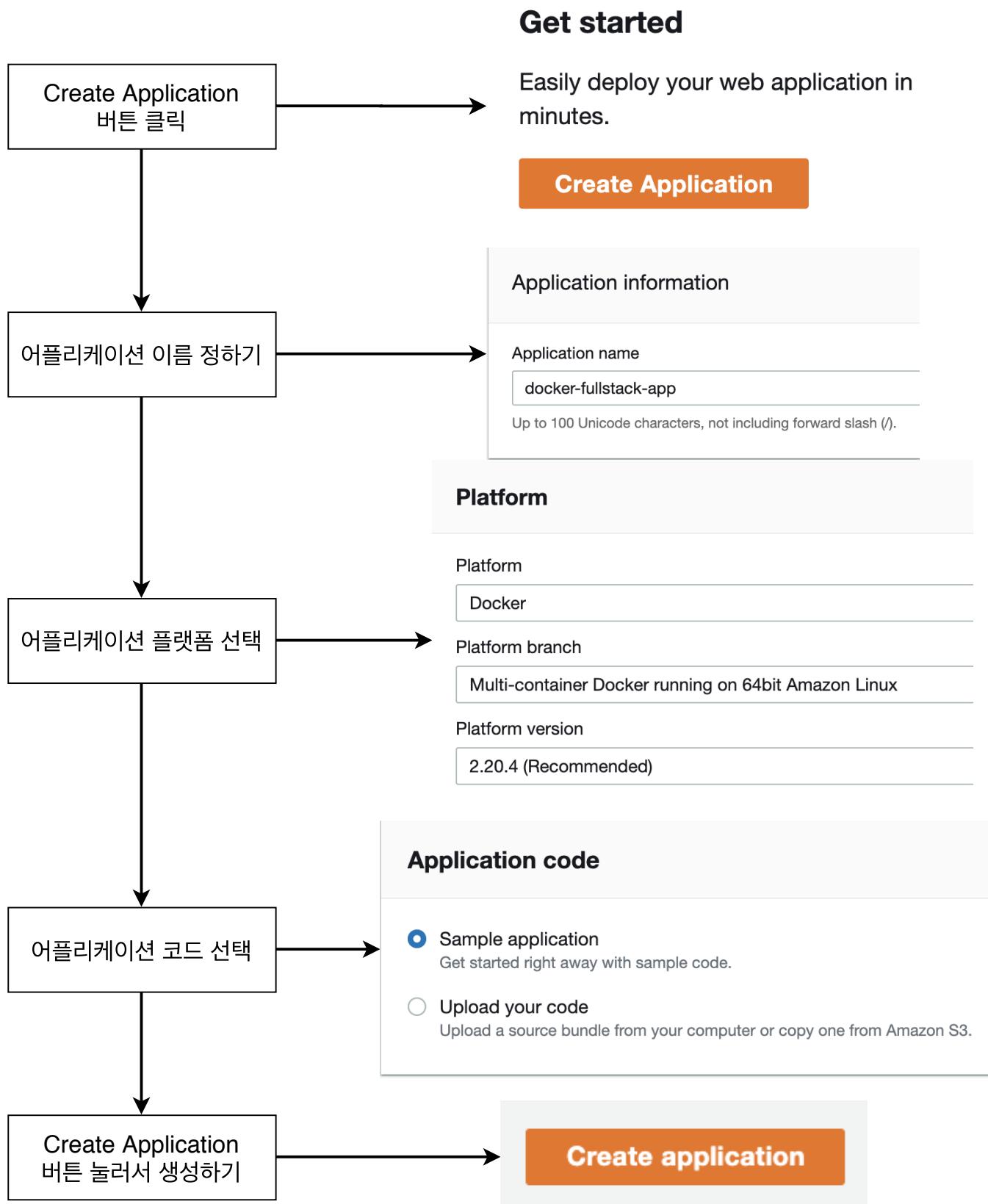
Frontend

Backend

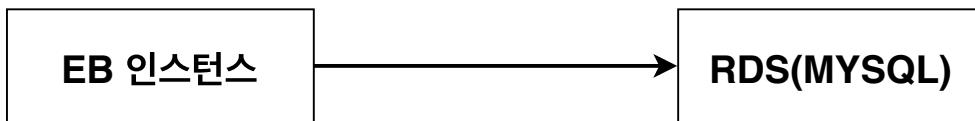


다중 컨테이너 앱을 위한 Elastic Beanstalk 환경 생성

어플리케이션 만드는 순서

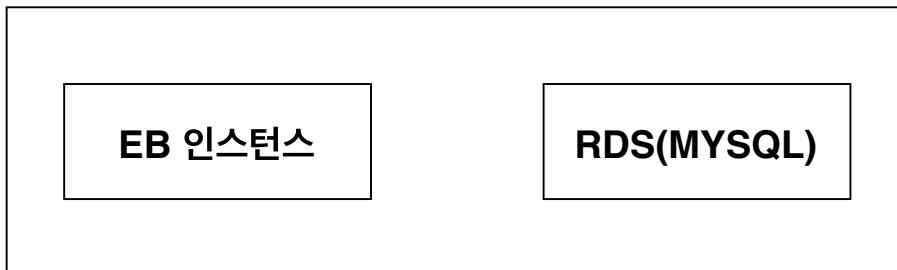


VPC(virtual private cloud)와 Security Group 설정하기

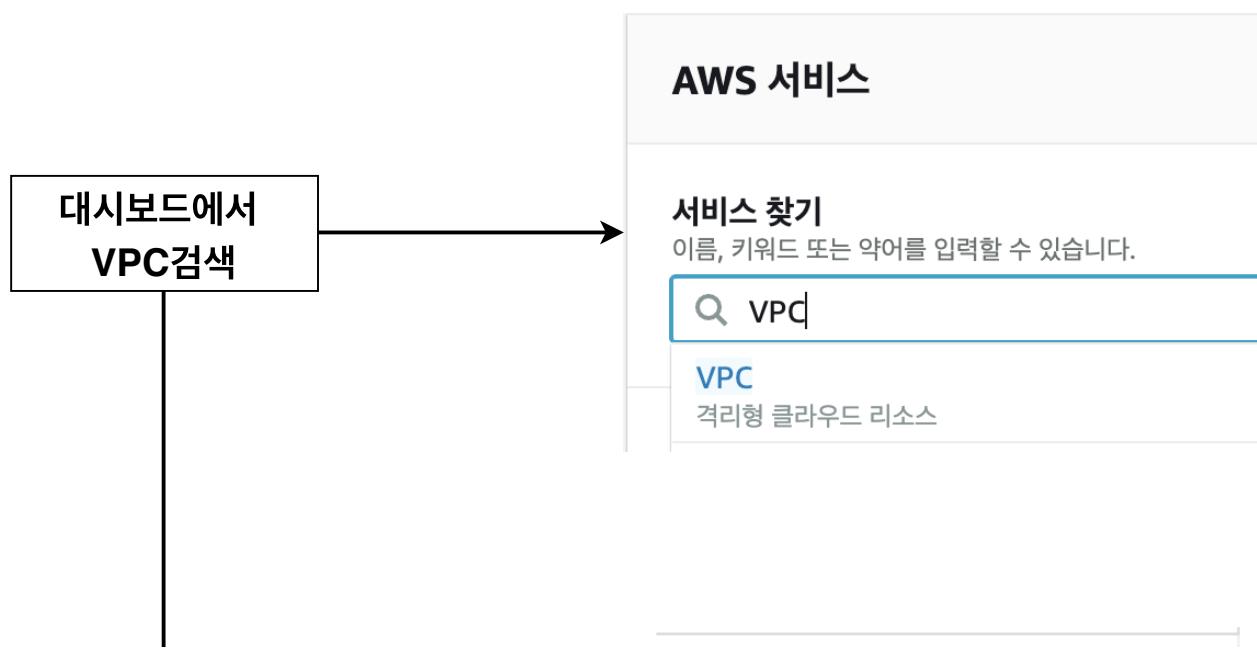


기본적으로 이렇게 연결이
되어있지 않아서 통신을 할
수 없기 때문에 따로 설정을
해줘서 연결시켜줘야 합니
다.

AP-Northeast-2에 할당된 기본 VPC



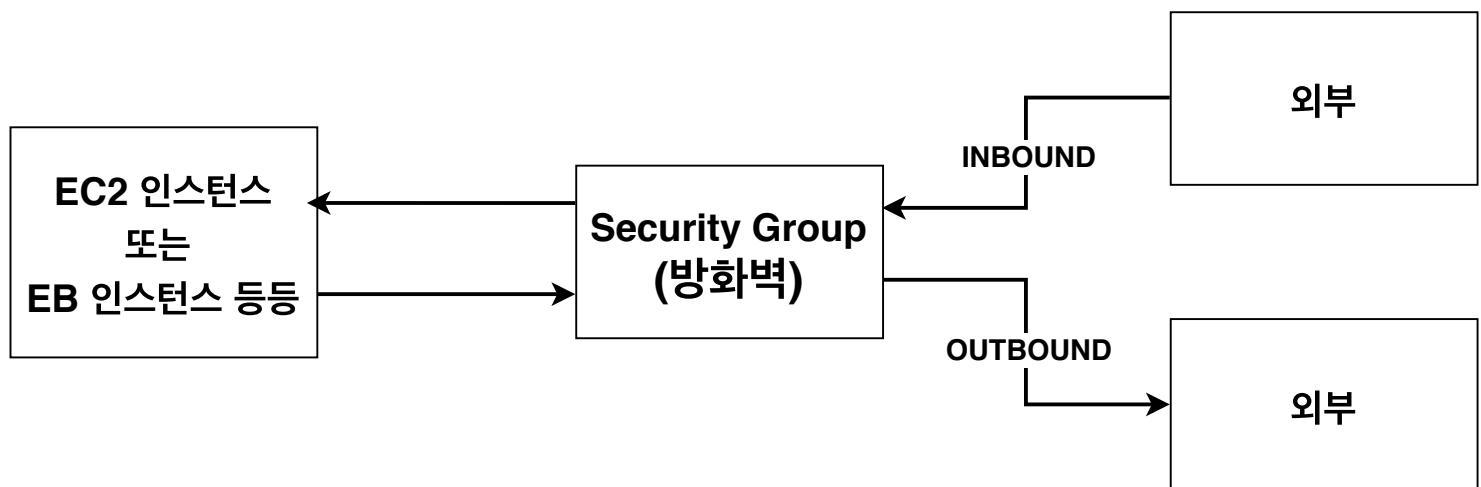
EB를 생성할때 할당되는 기본 VPC를 보기 위해서는



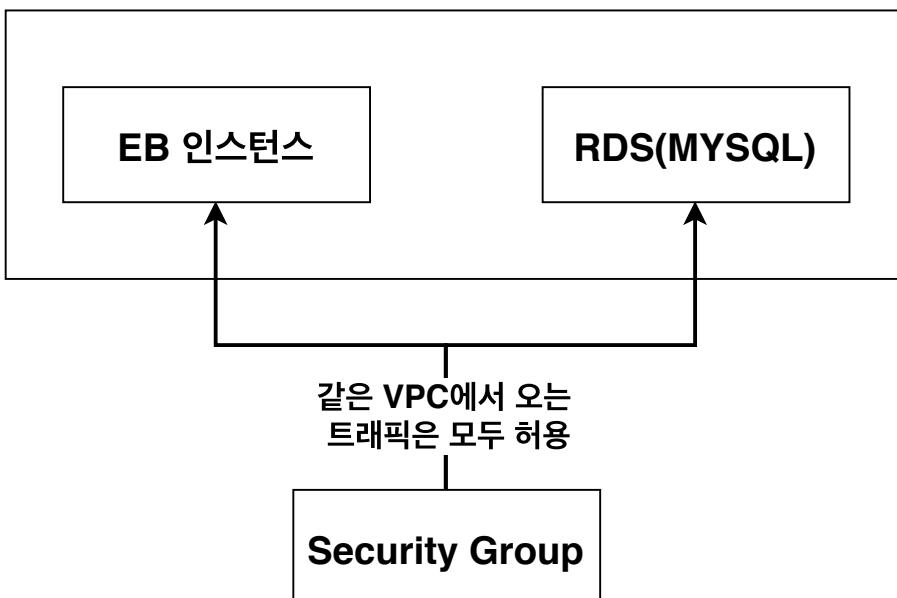
VPC 대시보드 New



Security Group(보안 그룹)이 무엇인지 알아볼게요.



AP-Northeast-2에 할당된 기본 VPC

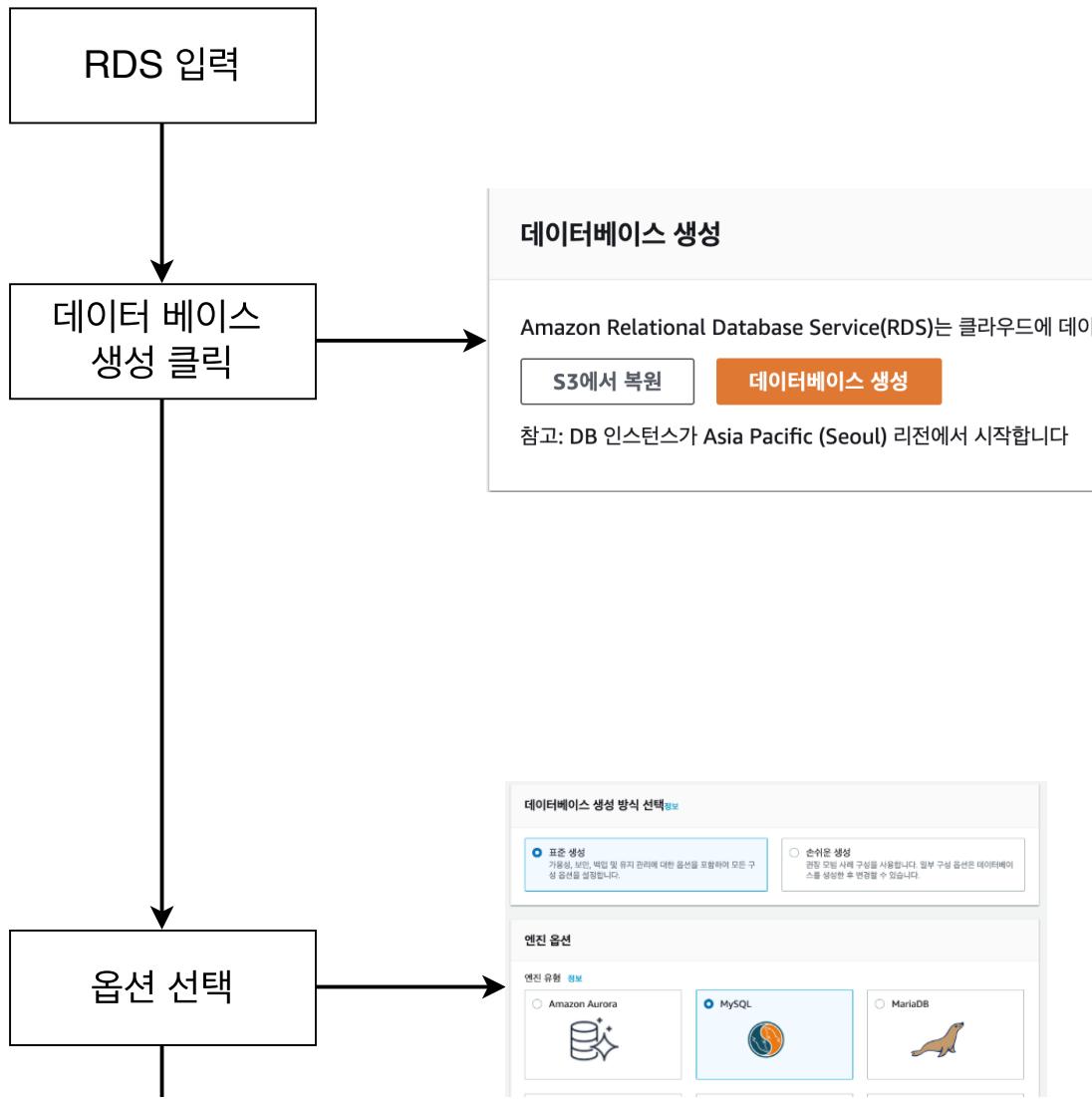


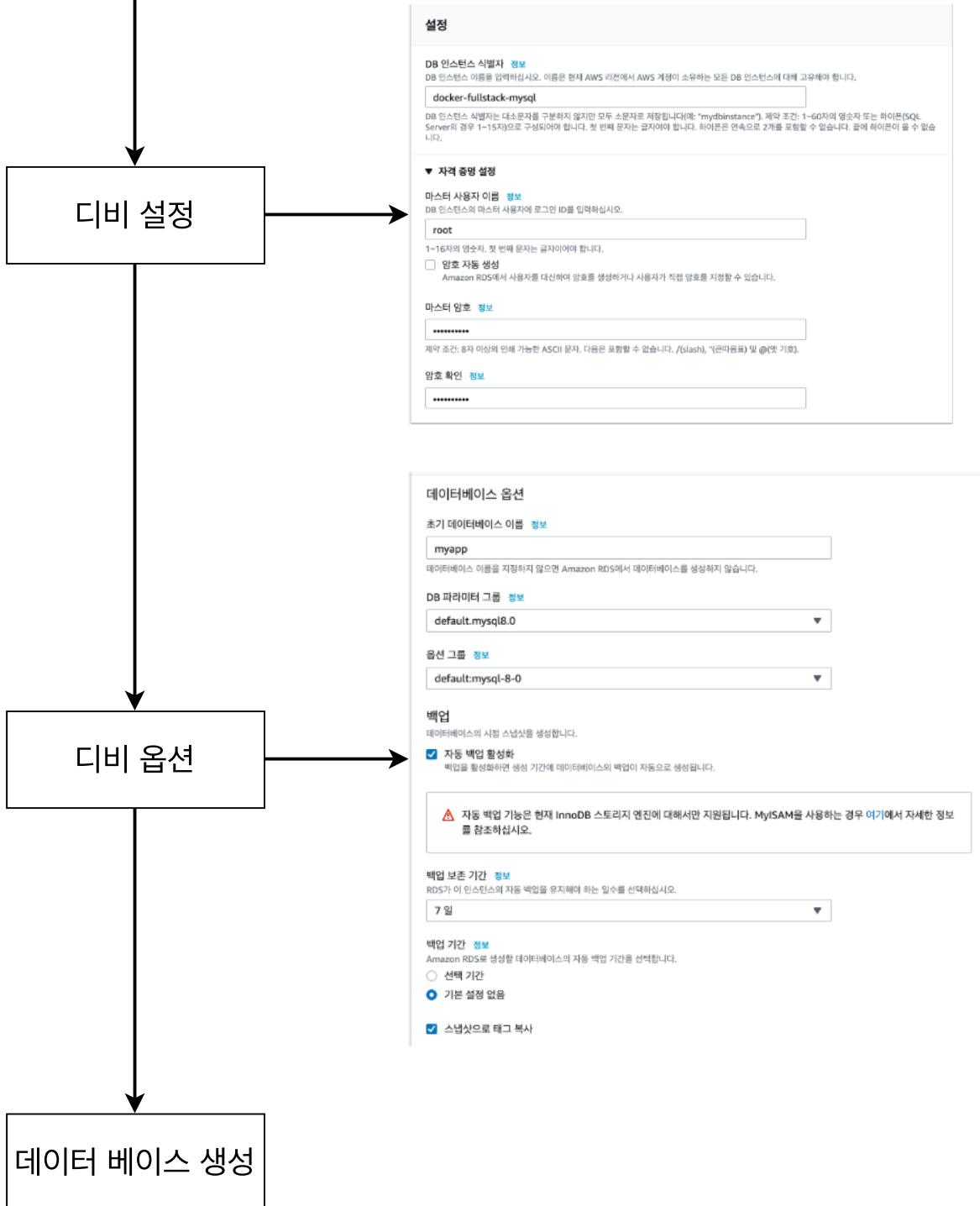
MYSQL을 위한 AWS RDS 생성하기

```
backend:  
  build:  
    dockerfile: Dockerfile.dev  
    context: ./backend  
  volumes:  
    - /app/node_modules  
    - ./backend:/app  
  environment:  
    MYSQL_HOST: mysql  
    MYSQL_USER: root  
    MYSQL_ROOT_PASSWORD: johnahn777  
    MYSQL_DATABASE: myapp  
    MYSQL_PORT: 3306
```

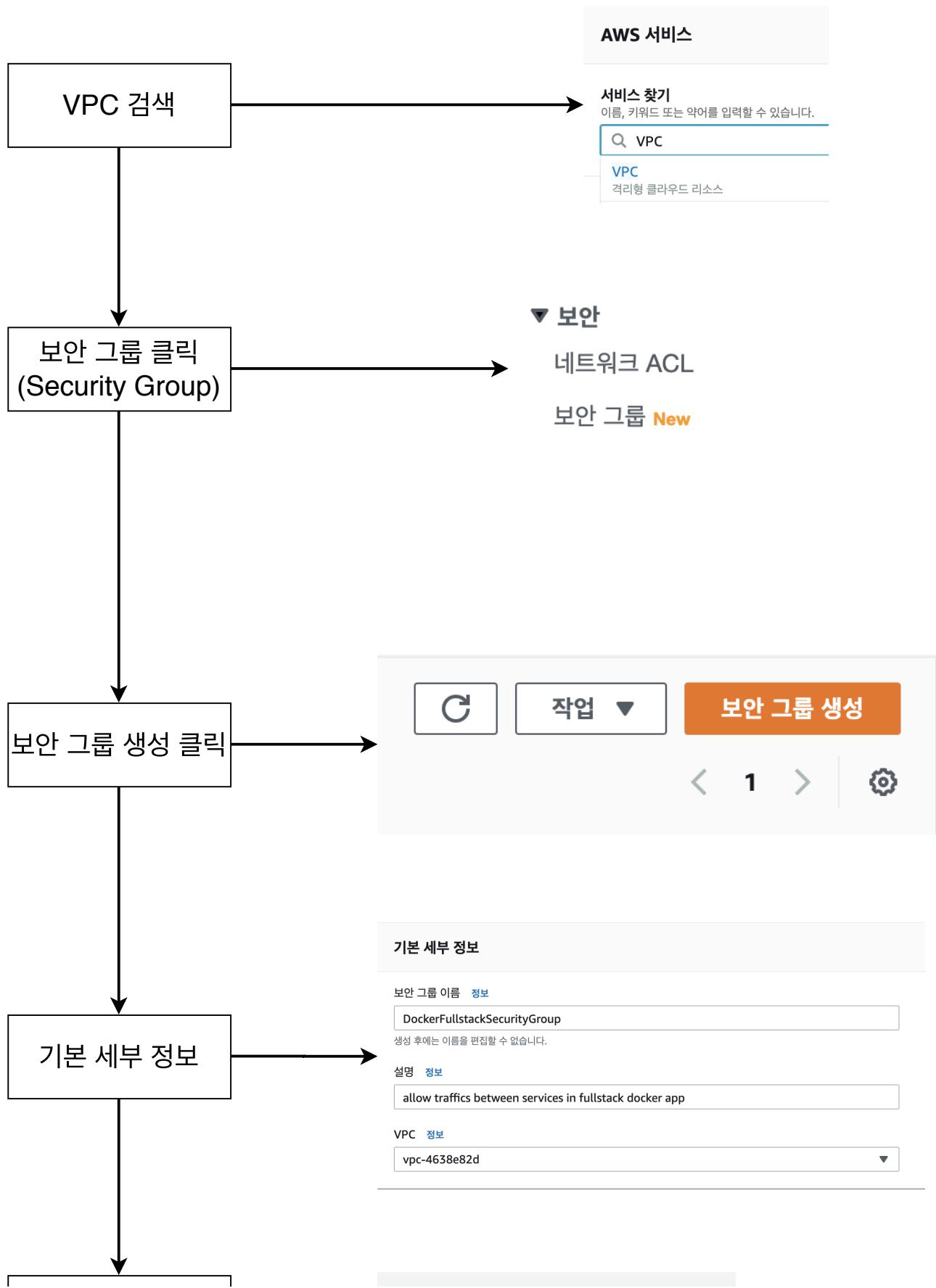
```
const mysql = require("mysql");  
const pool = mysql.createPool({  
  connectionLimit: 10,  
  host: process.env.MYSQL_HOST,  
  user: process.env.MYSQL_USER,  
  password: process.env.MYSQL_ROOT_PASSWORD,  
  database: process.env.MYSQL_DATABASE,  
  port: process.env.MYSQL_PORT  
});  
exports.pool = pool;
```

이제는 RDS를 생성해보겠습니다.





Security Group 생성하기



우선 다른 옵션 체크
없이 보안 그룹 생성

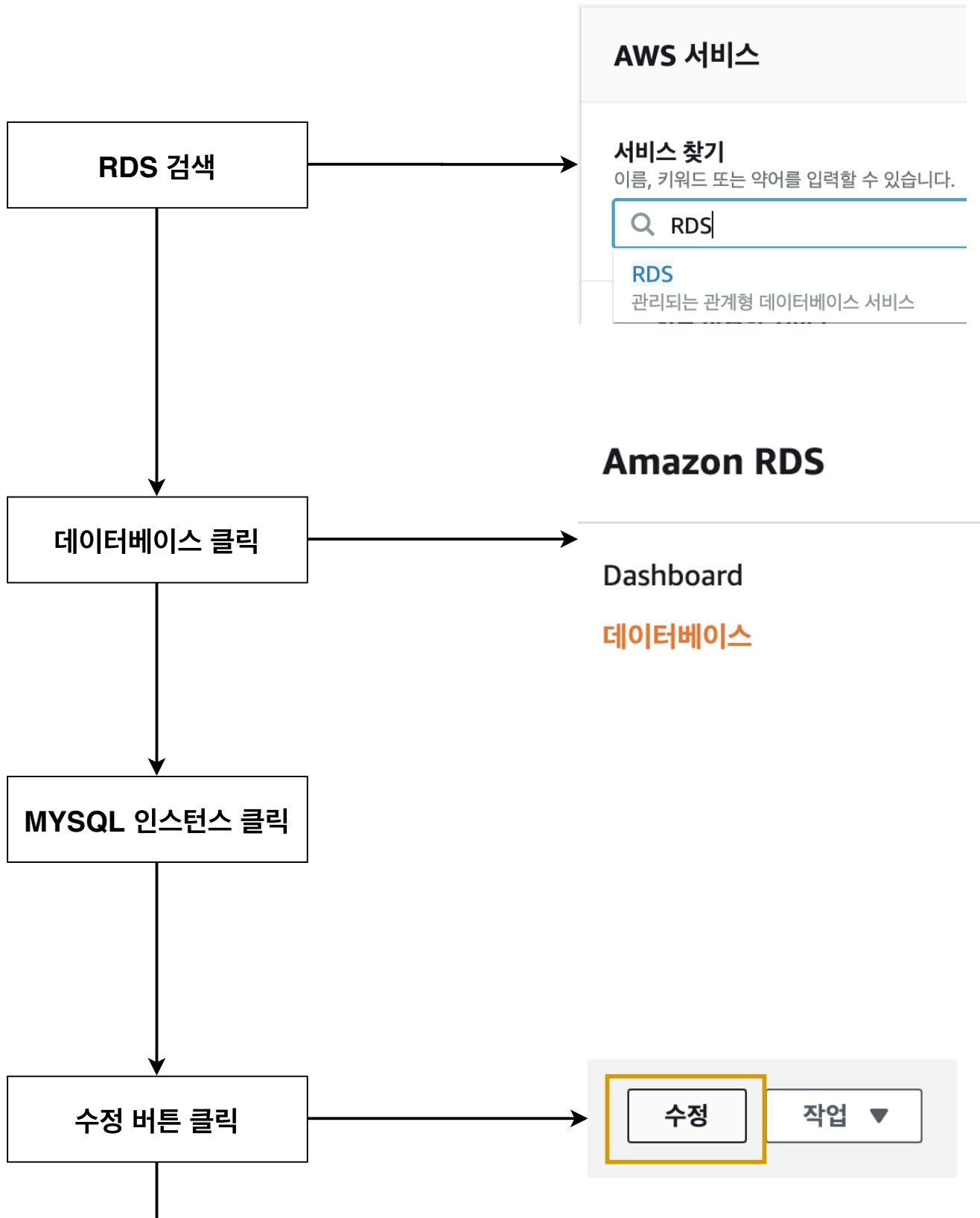
최소

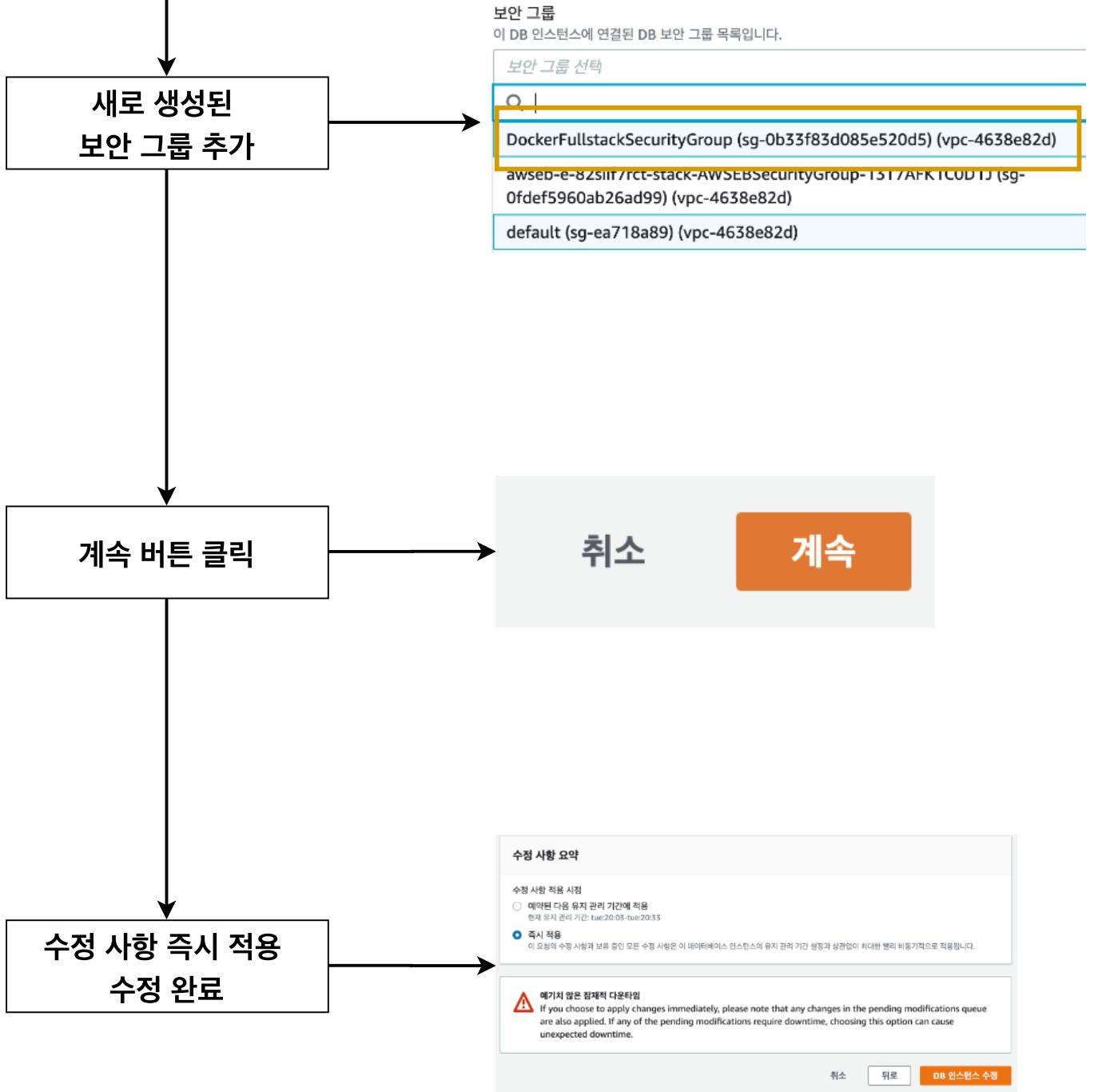
보안 그룹 생성

인바운드 규칙 설정

Security Group 적용하기

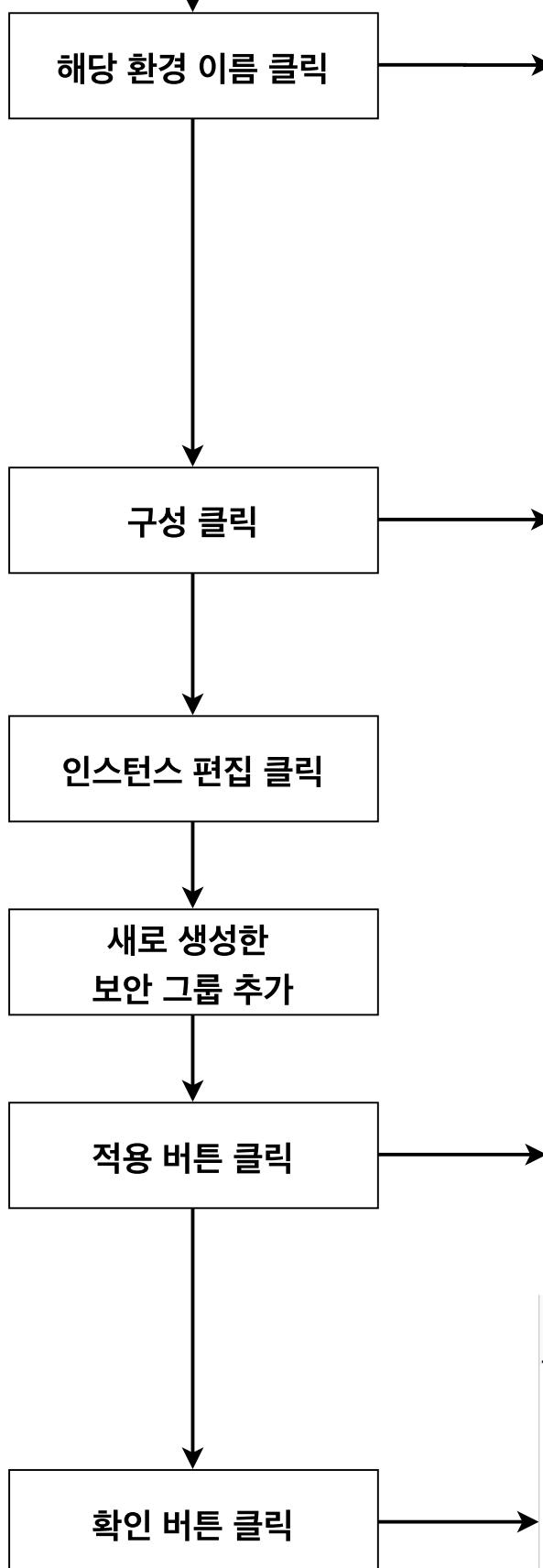
먼저 MySQL 인스턴스에 새로 생성된 보안 그룹 적용





먼저 EB 인스턴스에 새로 생성된 보안 그룹 적용





환경 이름 ▲ 상태 ▽ 애플리케이션 이름 ▽

DockerFullstackApp-env Ok docker-fullstack-app

▼ DockerFullstackApp-env

환경으로 이동 ↗

구성

취소 계속 적용

warnings (1) errors

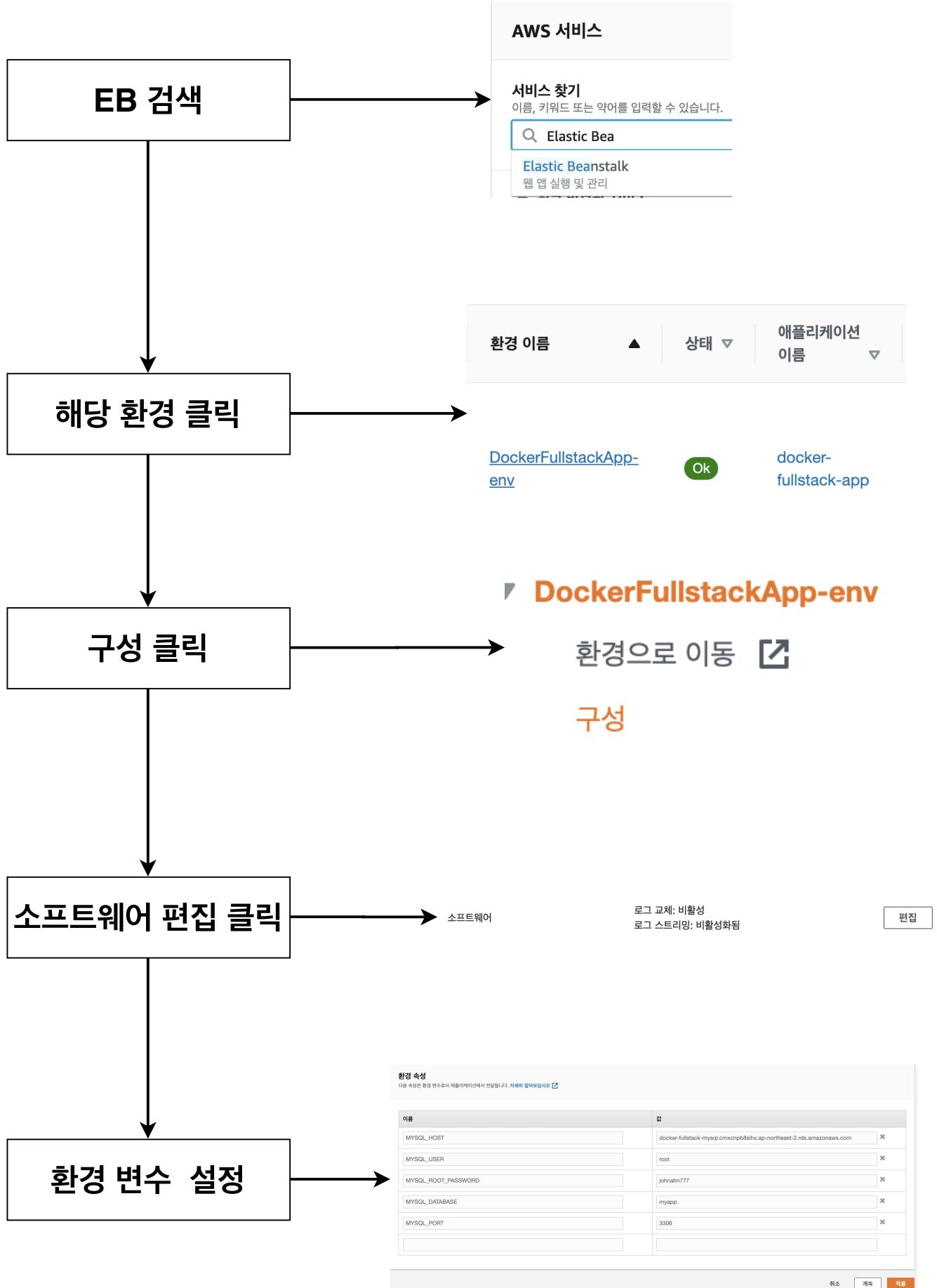
! Changes to option SecurityGroups settings will not take effect immediately. Each of your existing EC2 instances will be replaced and your new settings will take effect then.

```
aws:autoscaling:launchconfiguration:SecurityGroups "awseb-e-82siif7rct-stack-AWSEBSecurityGroup-13T7AFK1C0D1J" => "DockerFullstackSecurityGroup,awseb-e-82siif7rct-stack-AWSEBSecurityGroup-13T7AFK1C0D1J"
```

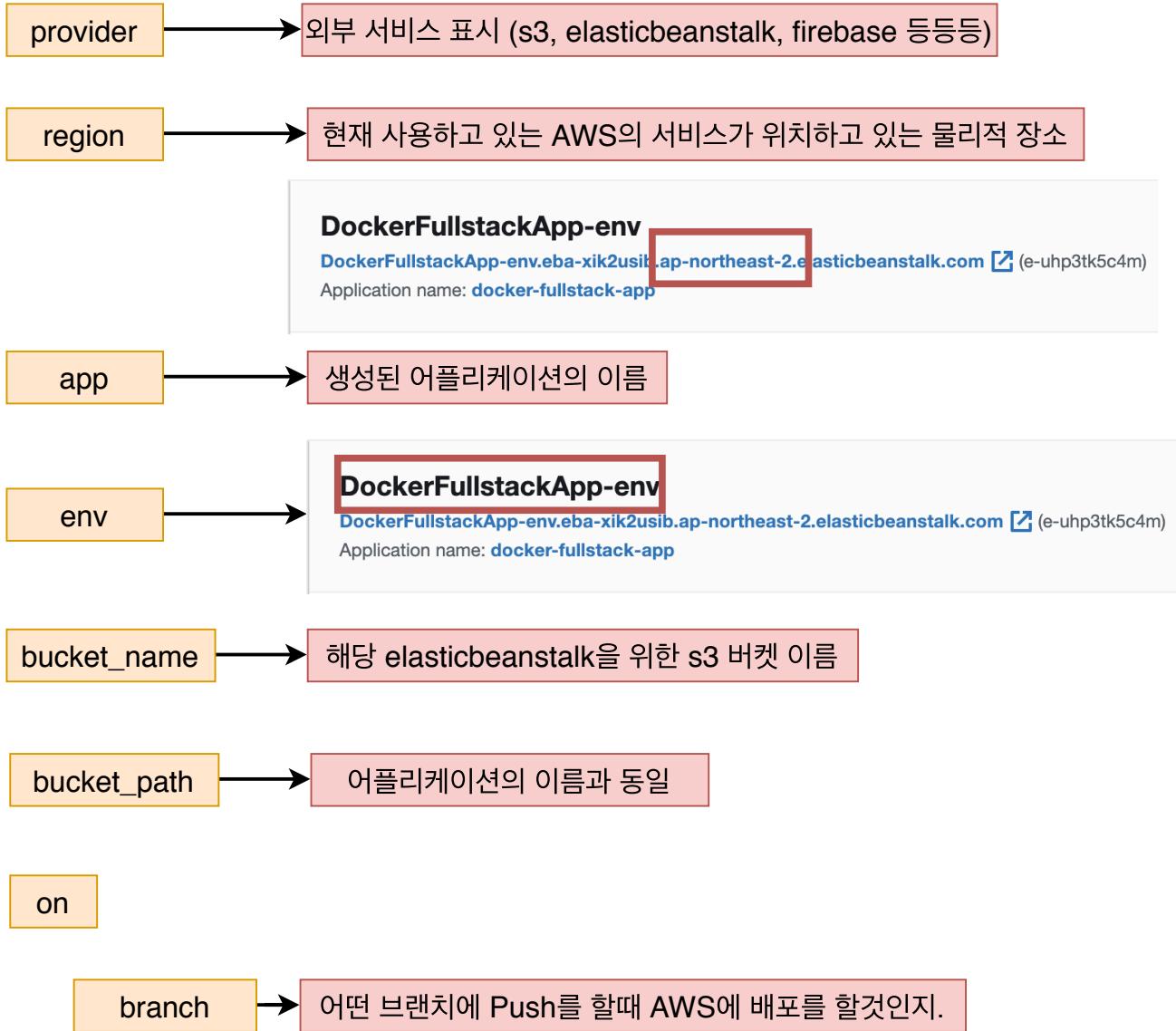
취소

확인

EB와 RDS 소통을 위한 환경 변수 설정하기

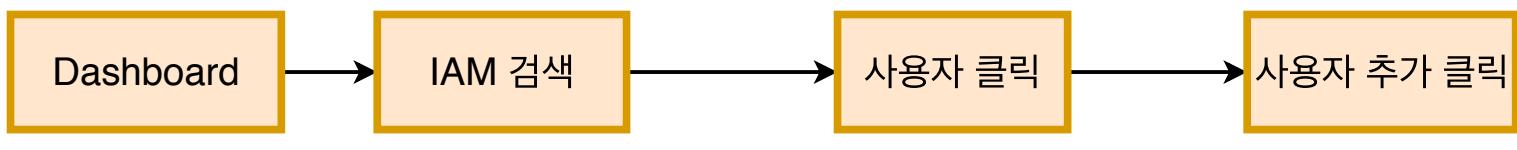
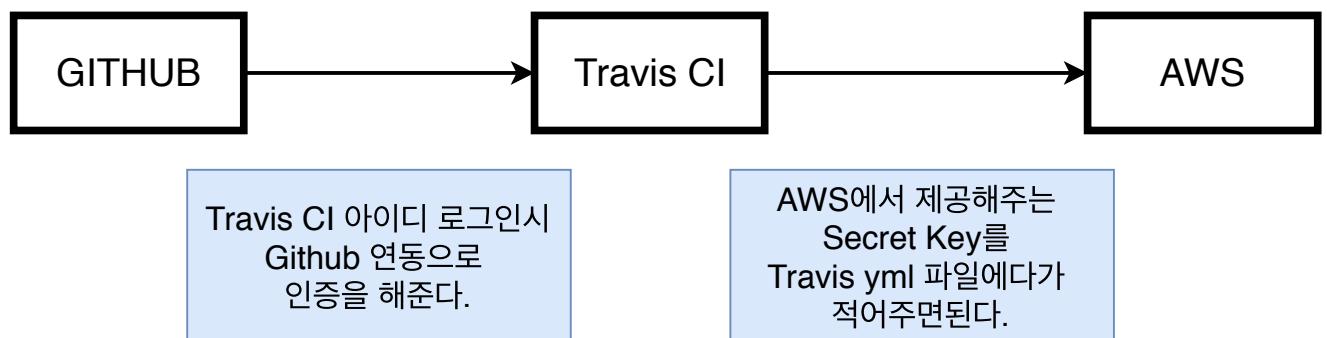


.travis.yml 파일 작성하기 (배포 부분)



Travis CI의 AWS 접근을 위한 API key 생성

소스 파일을 전달하기 위한 접근 요건



AWS 서비스

서비스 찾기
이름, 키워드 또는 약어를 입력할 수 있습니다.
IAM

IAM
AWS 리소스에 대한 액세스 관리

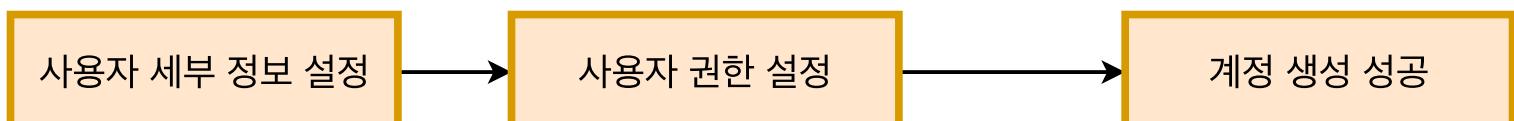
Identity and Access Management(IAM)

대시보드

액세스 관리

그룹

사용자



권한 설정

그룹에 사용자 추가
기존 사용자에서 권한 복사
기존 정책 직접 연결

정책 생성

정책 필터 <input> elasticbeanstalk

정책 이름	유형
AWSElasticBeanstalkCustomPlatformforEC2Role	AWS 관리형
AWSElasticBeanstalkPlatformPolicy	AWS 관리형

<input type="checkbox"/>	▶ AWSElasticBeanstalkEnhancedHealth	AWS 관리형
<input checked="" type="checkbox"/>	▶ AWSElasticBeanstalkFullAccess	AWS 관리형
<input type="checkbox"/>	▶ AWSElasticBeanstalkMulticontainerDocker	AWS 관리형
<input type="checkbox"/>	▶ AWSElasticBeanstalkReadOnlyAccess	AWS 관리형
<input type="checkbox"/>	▶ AWSElasticBeanstalkRoleCore	AWS 관리형

직접 API키를 Travis yml 파일에 적어 주면 노출이 되기 때문에 다른곳에 적고 그것을 가져와줘야한다.

Travis 웹사이트 해당 저장소 대쉬보드에 오기

설정 클릭

사진과 같이 AWS에서 받은 API 키들을 NAME과 VALUE에 적어서 넣어줍니다. 이곳에 넣어 주면 외부에서 접근을 할 수 없어 더욱 안전합니다.

Travis CI 웹사이트에서 보관중인 Key를 로컬 환경에서 가지고 올수 있게 travis yml 파일에서 설정을 해줍니다.

Environment Variables

Customize your build using environment variables. For secure tips on generating private keys read our documentation

AWS_ACCESS_KEY	*****	Available to all branches
AWS_SECRET_ACCESS_KEY	*****	Available to all branches
DOCKER_HUB_ID	*****	Available to all branches
DOCKER_HUB_PASSWORD	*****	Available to all branches

```

deploy:
  provider: elasticbeanstalk
  region: "ap-northeast-2"
  app: "docker-react-app"
  env: "DockerReactApp-env"
  bucket_name: "elasticbeanstalk-ap-northeast-2-972153559337"
  bucket_path: "docker-react-app"
  on:
    branch: master
    access_key_id: $AWS_ACCESS_KEY
  
```

```
secret_access_key: $AWS_SECRET_ACCESS_KEY
```

