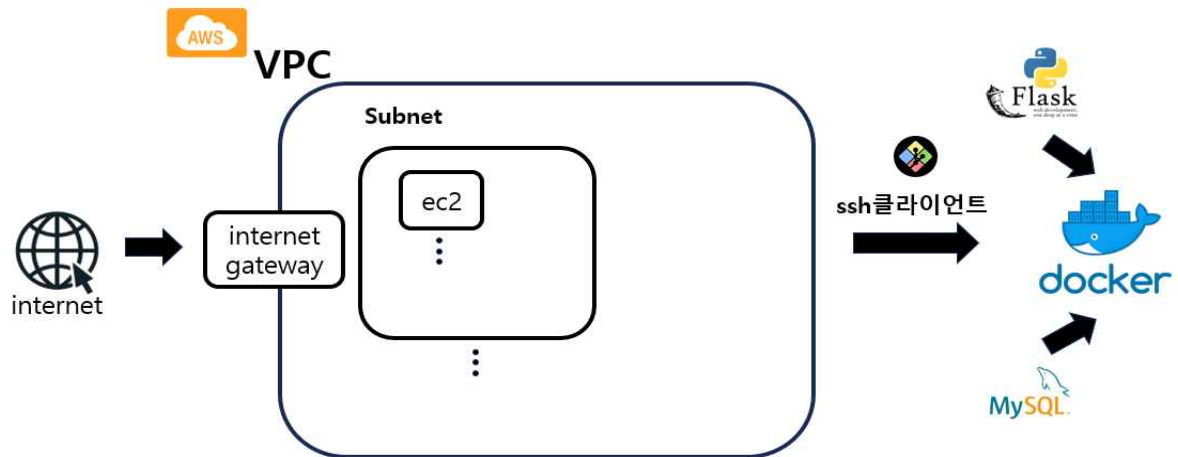


# 배포기술서

## 1. < 서버 아키텍처 구조 >



AWS에서 VPC와 subnet을 생성하고 그 안에 ec2 instance를 생성한다.  
instance와 연결할 ssh클라이언트로써 git bash를 사용한다.  
생성된 가상환경에 docker를 설치하고 이를 이용해 배포한다.

## 2. < instance 설정 >

VPC와 subnet을 따로 생성하지 않고 instance를 만들면서 default 값으로 생성한다.

이름을 설정한다

### 인스턴스 시작 정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 머신 또는 인스턴스를 생성할 수 있습니다. 아래의 간단한 단계에 따라 빠르게 시작할 수 있습니다.

#### 이름 및 태그 정보

이름

추가 태그 추가

본 프로젝트에서는 AWS의 12개월 무료사용을 이용하기 위해 프리티어 옵션들을 선택한다.

**Quick Start**

Amazon Linux  
aws

macOS  
Mac

Ubuntu  
ubuntu

Windows  
Microsoft

Red Hat  
Red Hat

SUSE Li  
SUS

더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type  
ami-086cae3329a3f7d75 (64비트(x86)) / ami-0f8536fc6ad2ba267 (64비트(Arm))  
가상화: hvm   ENA enabled: true   루트 디바이스 유형: ebs

프리 티어 사용 가능

▼ 인스턴스 유형 [정보](#) | [Get advice](#)

인스턴스 유형

t2.micro  
패밀리: t2   1 vCPU   1 GiB 메모리   현재 세대: true  
온디맨드 RHEL 기본 요금: 0.0744 USD 시간당  
온디맨드 Linux 기본 요금: 0.0144 USD 시간당  
온디맨드 SUSE 기본 요금: 0.0144 USD 시간당  
온디맨드 Windows 기본 요금: 0.019 USD 시간당

프리 티어 사용 가능

☒ 모든 세대

[인스턴스 유형 비교](#)

소프트웨어가 사전 설치된 AMI에는 추가 비용이 적용됩니다.

ssh 클라이언트로 putty를 사용할 것이 아니기 때문에 key는 .pem 형식으로 설정한다.

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

key

이름에는 최대 255개의 ASCII 문자를 포함할 수 있습니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.

키 페어 유형

☒ RSA

RSA 암호화된 프라이빗 및 퍼블릭 키 페어

☐ ED25519

ED25519 암호화된 프라이빗 및 퍼블릭 키 페어

프라이빗 키 파일 형식

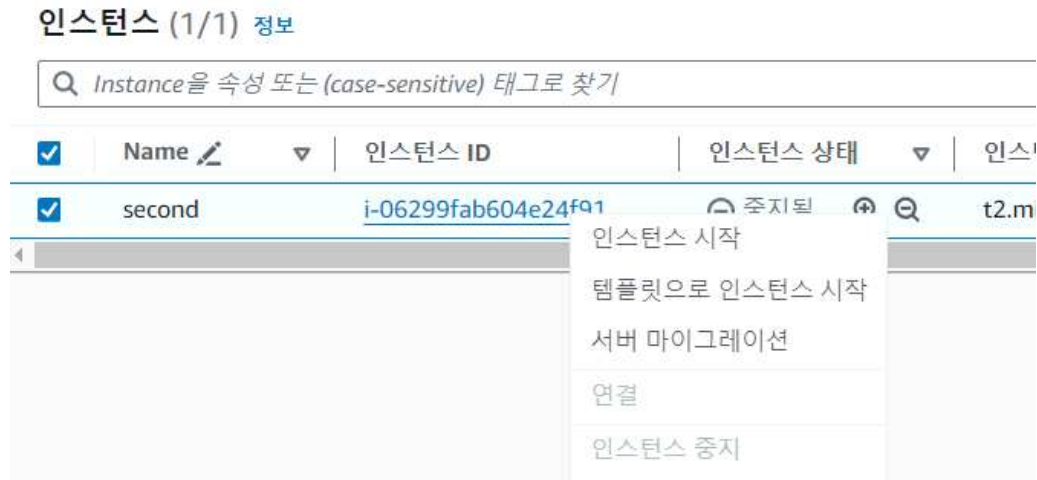
☒ .pem

OpenSSH와 함께 사용

☐ .ppk

Putty와 함께 사용

생성된 instance를 우클릭하여 실행(활성화) 시킨다.



연결할 ssh 클라이언트로는 git bash를 사용한다.

우선 자신이 받은 key 파일이 있는 디렉토리로 이동

```
Administrator@DESKTOP-622RD6I MINGW64 ~  
$ cd /d/key  
  
Administrator@DESKTOP-622RD6I MINGW64 /d/key  
$ |
```

해당 코드의 -----자리에 instance의 public ip주소값을 넣어 실행

```
Administrator@DESKTOP-622RD6I MINGW64 /d/key  
$ ssh -i "key.pem" ubuntu@-----|
```

### 3. < ubuntu 기반 도커 설치 >

1. 우분투(Ubuntu) 운영체제가 사용하는 소프트웨어 정보를 담고 있는 목록을 최신

상태로 업데이트

```
sudo apt-get update
```

## 2. Docker 설치를 위한 필수 패키지를 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg  
lsb-release
```

## 3. Docker 공식 GPG 키를 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg  
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

## 4. Docker 레포지토리를 등록

```
echo \  
"deb [arch=amd64  
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  
https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >  
/dev/null
```

## 5. 패키지 인덱스를 업데이트

```
sudo apt-get update
```

## 6. Docker를 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin
```

## 7. Docker가 제대로 설치되었는지 확인

`sudo docker version`

## 4. < 기본적인 docker 명령어 >

현재 실행 중인 컨테이너 확인

`docker ps`

가상환경에 존재 하는 모든 컨테이너 목록

`docker ps -a`

존재하는 이미지 목록 확인

`docker images`

컨테이너 삭제

`docker rm 컨테이너ID or 이름` (컨테이너ID는 앞 3자리만 사용해도 가능하다)

이미지 삭제

`docker rmi imageID or 이름`

이미지 다운로드

`docker pull 이미지_이름[:태그]`

## 5. < 필요한 라이브러리 설치 및 실행 >

컨테이너를 실행

`docker start 컨테이너ID`

실행 중인 컨테이너에서 해당 이미지 실행

```
docker exec -it 컨테이너ID /bin/bash
```

github에서 repository주소를 복사

명령창에 다음 코드 입력

```
git clone 주소
```

필요한 라이브러리 설치

```
pip install -r requirements.txt
```

플라스크 실행

```
flask run
```

...

< 데이터베이스 연동은 현재 진행 중이며 추후 업데이트 예정 >