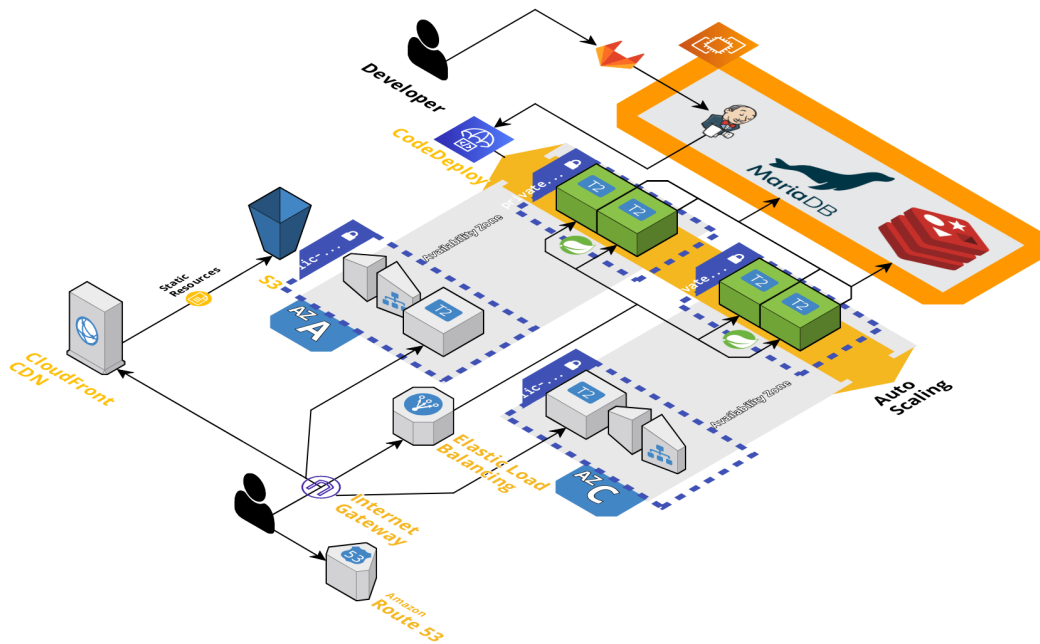


# 포팅 매뉴얼

## 1. 프로젝트 기술 스택

FE	<ul style="list-style-type: none"><li>• HTML5</li><li>• CSS3</li><li>• JavaScript</li><li>• React : 18.2.0</li><li>• Redux-toolkit : 1.8.6</li><li>• TypeScript : 4.8.4</li></ul>
BE	<p><b>Infra</b></p> <ul style="list-style-type: none"><li>• AWS EC2</li><li>• AWS VPC</li><li>• AWS S3</li><li>• AWS Cloudfront</li><li>• AWS IAM</li><li>• AWS ACM</li><li>• AWS Route 53</li><li>• AWS CodeDeploy</li><li>• Docker 20.10.21</li><li>• Jenkins 2.377</li></ul> <p><b>Development</b></p> <ul style="list-style-type: none"><li>• Java 17.0.5</li><li>• Spring boot 2.7.4</li><li>• spring-data-jpa 2.7.4</li><li>• spring-data-redis 2.7.4</li><li>• hibernate-core-5.6.11.Final</li><li>• querydsl-core:5.0.0</li><li>• projectlombok:1.18.24</li><li>• jbcrypt:0.4</li><li>• spring-cloud-starter-aws:2.2.6.RELEASE</li><li>• jjwt:0.11.5</li><li>• jarkarta.validation:2.0.2</li></ul> <p><b>Test</b></p> <ul style="list-style-type: none"><li>• junit-jupiter:5.8.2</li><li>• mockito-core:4.5.1</li><li>• Apache JMeter 5.5</li></ul> <p><b>DB</b></p> <ul style="list-style-type: none"><li>• mariadb 10.11.1</li></ul>

## 2. 서버 아키텍처



본 프로젝트의 아키텍처는 위와 같습니다. 각 서버 리소스는 특정 포트로 식별 가능하며 접근할 수 있습니다.

각 서버의 포트 번호는 다음과 같습니다. (도커 배포 기준)

서버	HTTP 포트	HTTPS 포트
jenkins	8088	-
redis	6379	-
mariadb	3306	-

## 3. 프로젝트 빌드 방법 (로컬 서버)

### 3.1. Gitlab에서 프로젝트 클론하기

1. 작업할 공간에 폴더를 하나 생성합니다.
2. 생성한 폴더를 열고 해당 위치에서 **Git Bash** 를 열어줍니다. (**CMD** 와 같은 다른 터미널도 상관없습니다!)
3. `git clone https://lab.ssafy.com/s07-blockchain-contract-sub2/S07P22D102.git` 를 터미널에 입력해줍니다.
4. gitlab에서 내려받은 파일이 생깁니다. 앞으로 해당 파일이 위치한 폴더를 **root directory** 라고 하겠습니다. 이후 작업 공간으로 가서 빌드 과정을 수행해주시면 됩니다.

## 3.2. 스프링부트 WAS 빌드

### 3.2.1. gradle로 직접 빌드하는 방법 (CMD 버전)

1. Win + R 을 누르고 cmd 를 입력하고 확인 버튼을 누릅니다. 그러면 명령 프롬프트 창을 띄울 수 있습니다.
2. 백엔드 작업 공간으로 이동해줍니다. 저의 경우에는 백엔드 작업 공간이 C:\Users\multicampus\Desktop\final-project\BE\shall-we-meet-then 입니다. 앞에 cd 명령어를 붙이면 해당 디렉토리로 이동할 수 있습니다.
3. 그 후 gradle 을 이용하여 빌드해줍니다. cmd 에 gradlew clean build -x test 명령어를 입력합니다. 그러면 서버 내부에서 진행하는 테스트 코드를 수행한 후 빌드 파일이 생깁니다.

```
C:\Users\multicampus\Desktop\final-project\BE\shall-we-meet-then>gradlew clean build -x test
BUILD SUCCESSFUL in 7s
6 actionable tasks: 6 executed
C:\Users\multicampus\Desktop\final-project\BE\shall-we-meet-then>
```

4. cd build/libs 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 java -jar shall-we-meet-then-0.0.1-SNAPSHOT.jar 명령어를 입력해줍니다.
5. 위의 과정을 마치면 로컬 환경에서 서버 빌드 및 배포가 되었습니다.

### 3.2.2. gradle로 직접 빌드하는 방법 (IntelliJ)

1. 인텔리제이를 통해 해당 프로젝트를 열어줍니다.

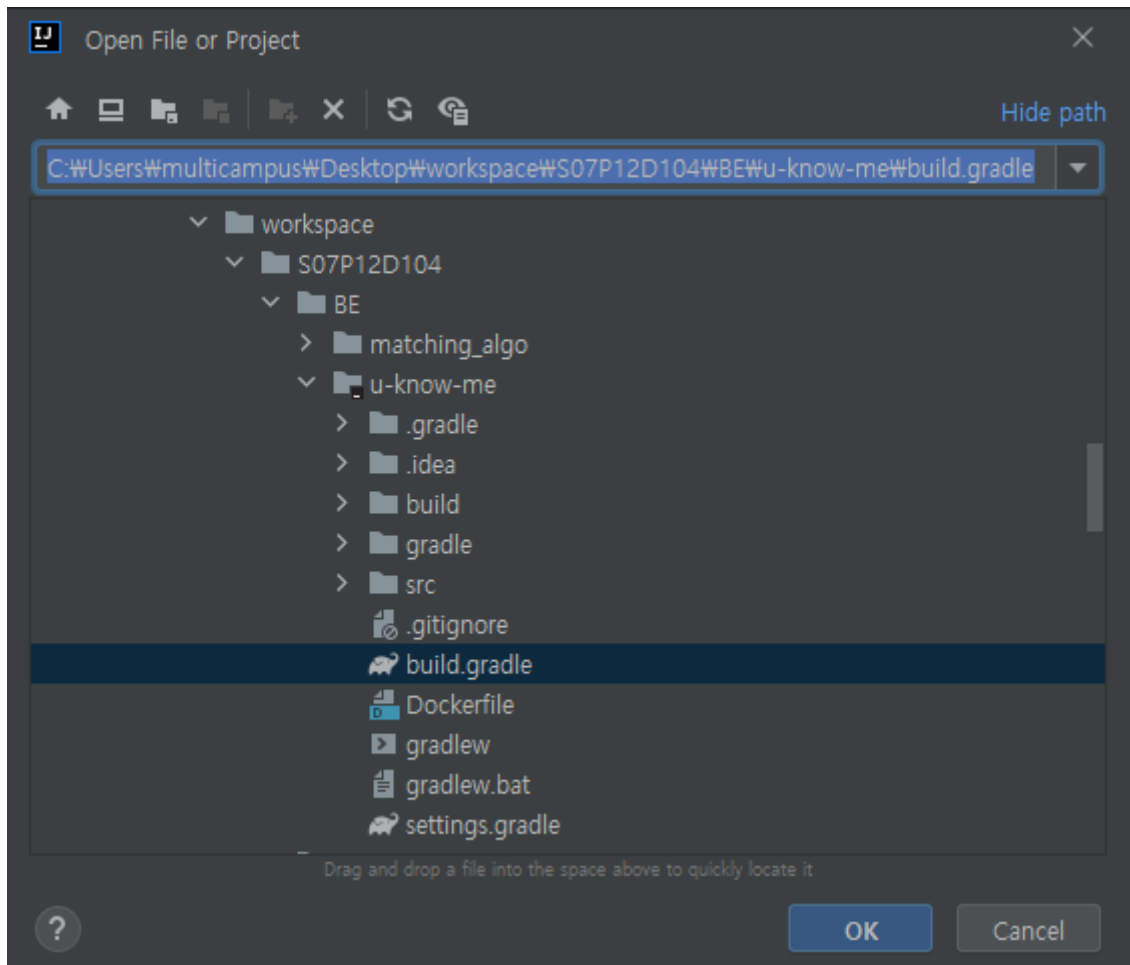


그림 3) 오픈 프로젝트를 통해 백엔드 프로젝트 열기

2. **Alt + F12** 를 눌러 터미널을 열어줍니다.

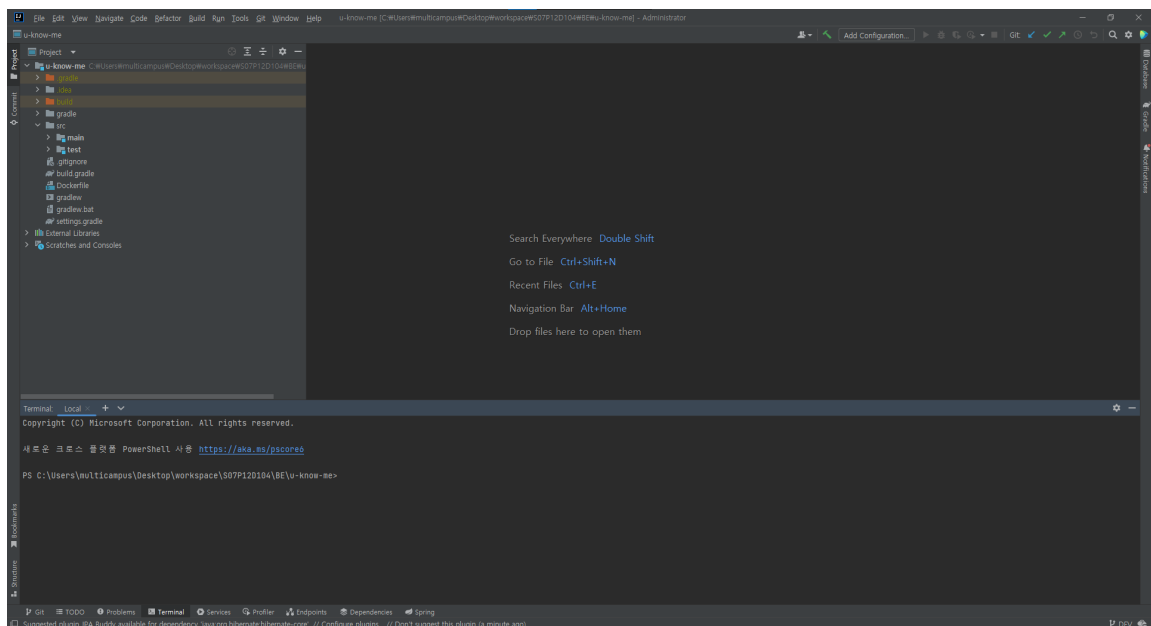


그림 4) 터미널을 연 상태의 IntelliJ

3. 터미널에 `./gradlew clean build` 명령어를 입력합니다.
4. `BUILD SUCCESSFUL` 이 뜨면 `cd build/libs` 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 `java -jar shall-we-meet-then-0.0.1-SNAPSHOT.jar` 명령어를 입력해줍니다.

## 4. 프로젝트 빌드 방법 (운영 서버)

### 4.1. VSCode를 이용한 ssh 접속

#### 1. Remote - SSH 설치

vscode에서 `ctrl + shift + x` 를 누르면 extensions 탭으로 넘어갈 수 있습니다. 해당 화면에서 검색창에 `ssh` 를 검색하면 `Remote - SSH` 라는 extension이 나오는데 `install` 버튼을 눌러 다운로드 받아줍니다.

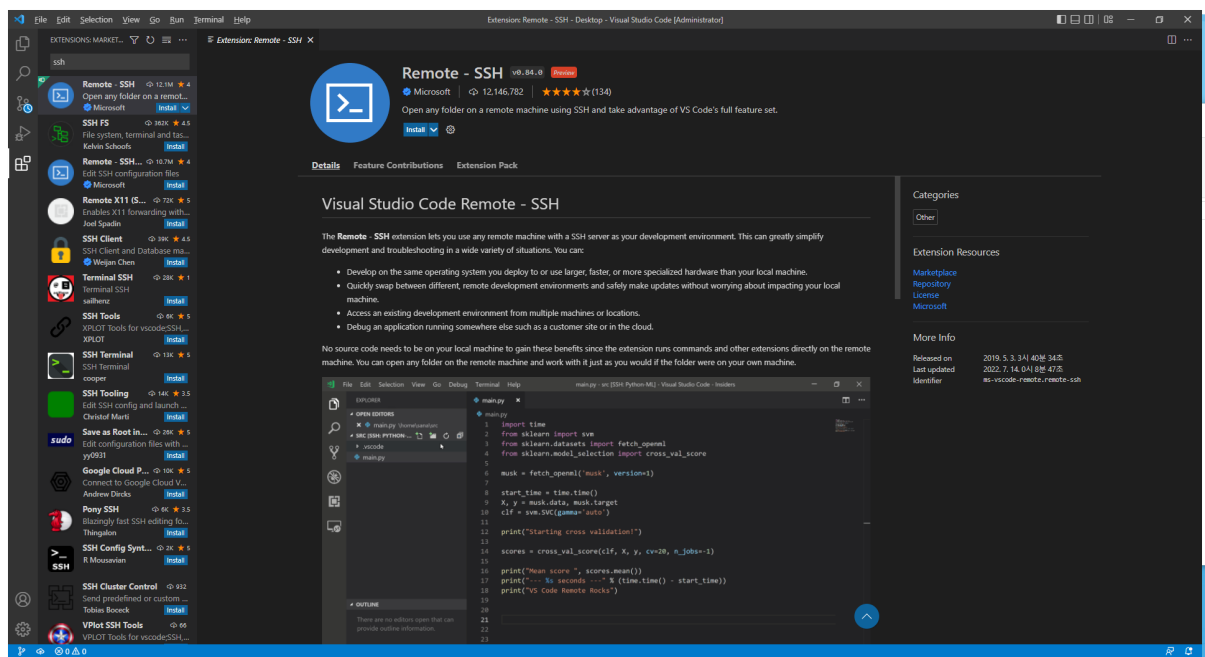


그림 5) extensions 탭

#### 2. SSH 설정 파일 등록

원래 터미널에서 `ssh -i 계정명@IP주소` 로 연결할 수 있지만 계속 터미널에 명령어를 입력하기는 번거로우니 설정 파일을 등록해줍니다.

우선 `f1` 버튼을 눌러 `ssh` 를 검색합니다. 그리고 `Remote-SSH:Open SSH Configuration File...` 이라는 탭을 선택합니다.

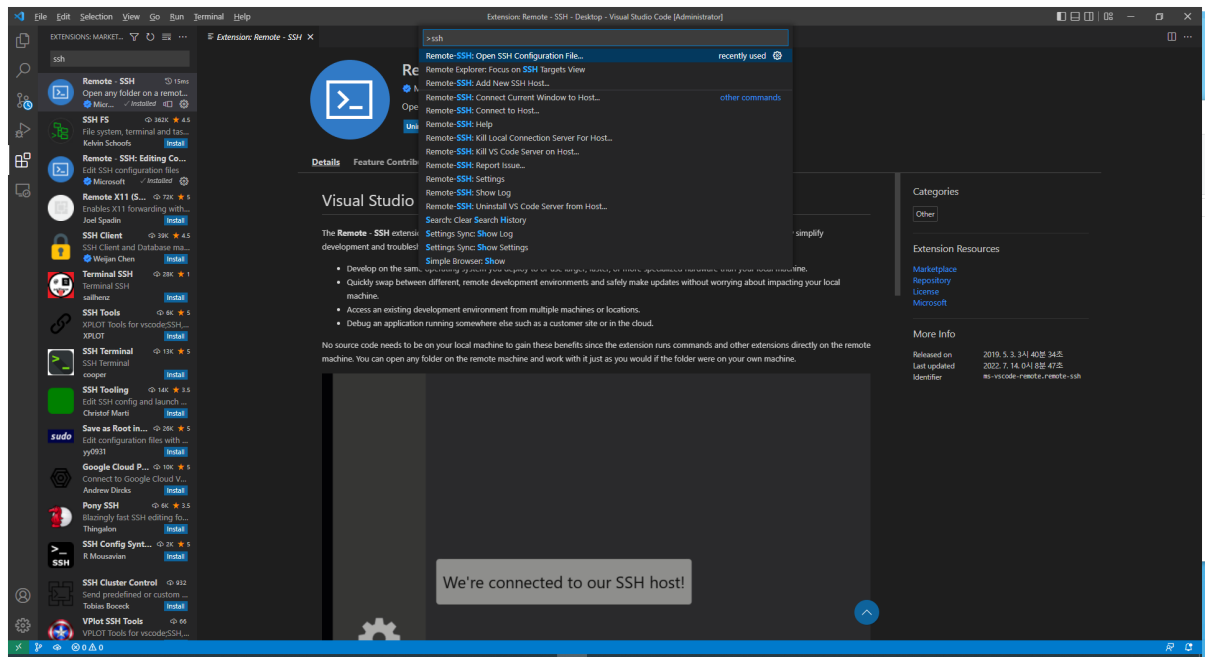


그림 6) ssh 검색

위의 버튼을 눌렀다면 같은 자리에 SSH 구성 파일 리스트가 나열됩니다. 여기서 `C:\Users\<계정명>\.ssh\config` 를 선택합니다.

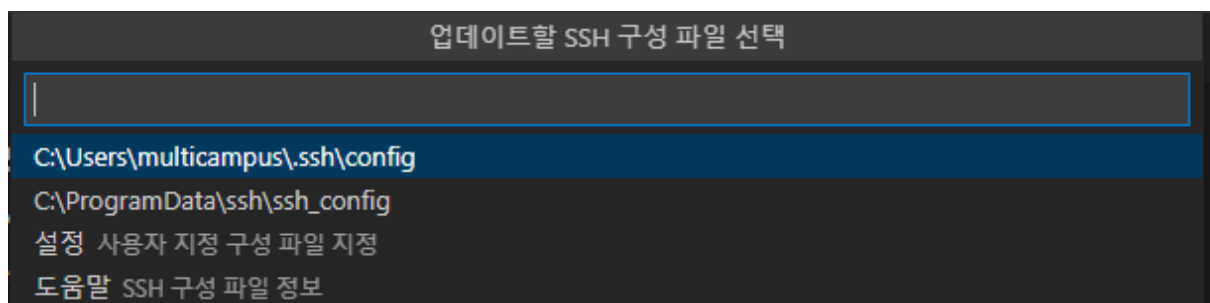


그림 7) SSH 구성 파일 리스트

그러면 SSH 구성 파일이 열립니다.

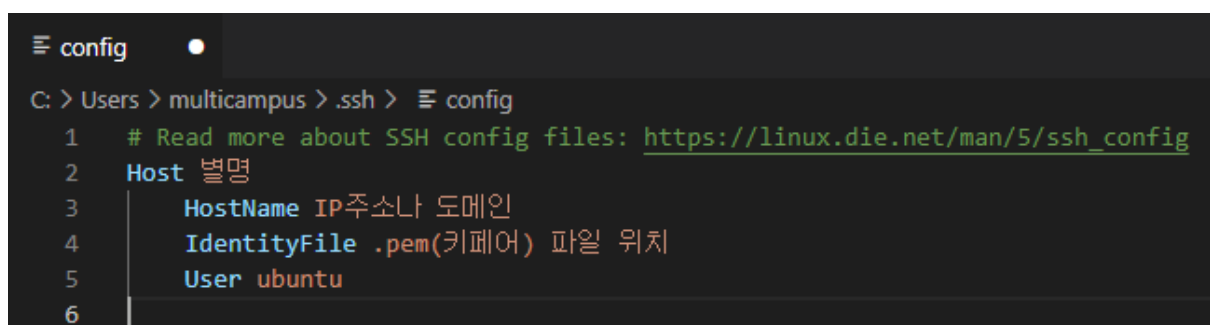


그림 8) SSH 구성 파일

각 요소에 대해 자세히 알아보겠습니다.

- **Host** : **Remote SSH** 의 이름을 설정해주면 됩니다. (해당 인스턴스가 무엇인지 알기 쉽게 이름을 정합니다.)
- **HostName** : AWS EC2 인스턴스의 **public IP** 나 도메인을 적으면 됩니다.
- **IdentityFile** : 현재 **.pem** 파일이 저장되어있는 위치를 작성하면 됩니다.
- **User** : 계정 이름을 설정한다. 우리는 **ubuntu** 를 사용합니다.
- **Port** : 기본값인 22번 포트가 아니라 다른 포트로 ssh 접근을 한다면 입력해줍니다.

### 3. SSH 세션 접속

입력을 다하고 저장한 후 좌측 탭에서 **Remote Explorer** 탭으로 이동합니다.

**SSH TARGET** 에 config에서 설정한 Host명으로 아이콘이 하나 생깁니다. Host명 우측의 폴더 아이콘을 클릭합니다.

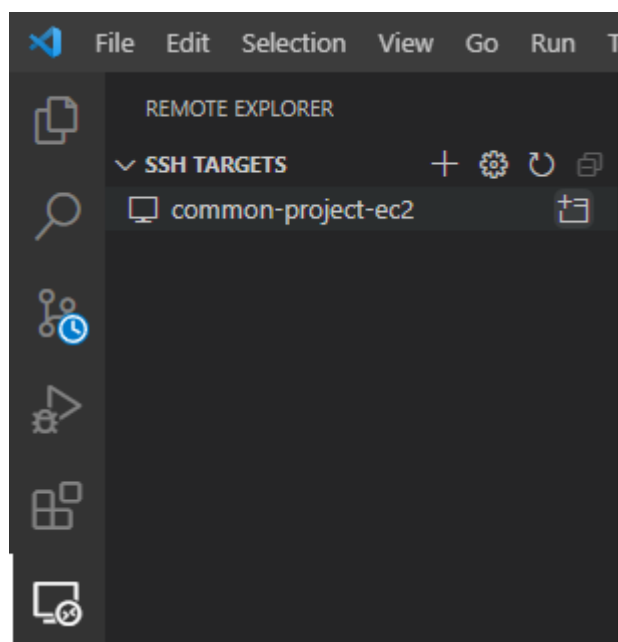


그림 9) Remote Explorer

그럼 새 vscode 창이 뜨면서 Linux, Windows, macOS를 선택하는 창이 나옵니다. 우리는 우분투를 사용하기 때문에 리눅스를 선택합니다.

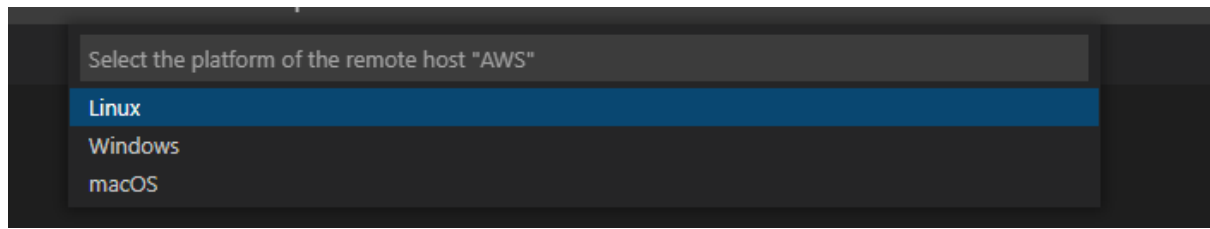


그림 10) AWS 플랫폼 선택창

그러면 SSH를 이용하여 AWS EC2 인스턴스를 vscode에서 편집할 수 있게 됩니다.

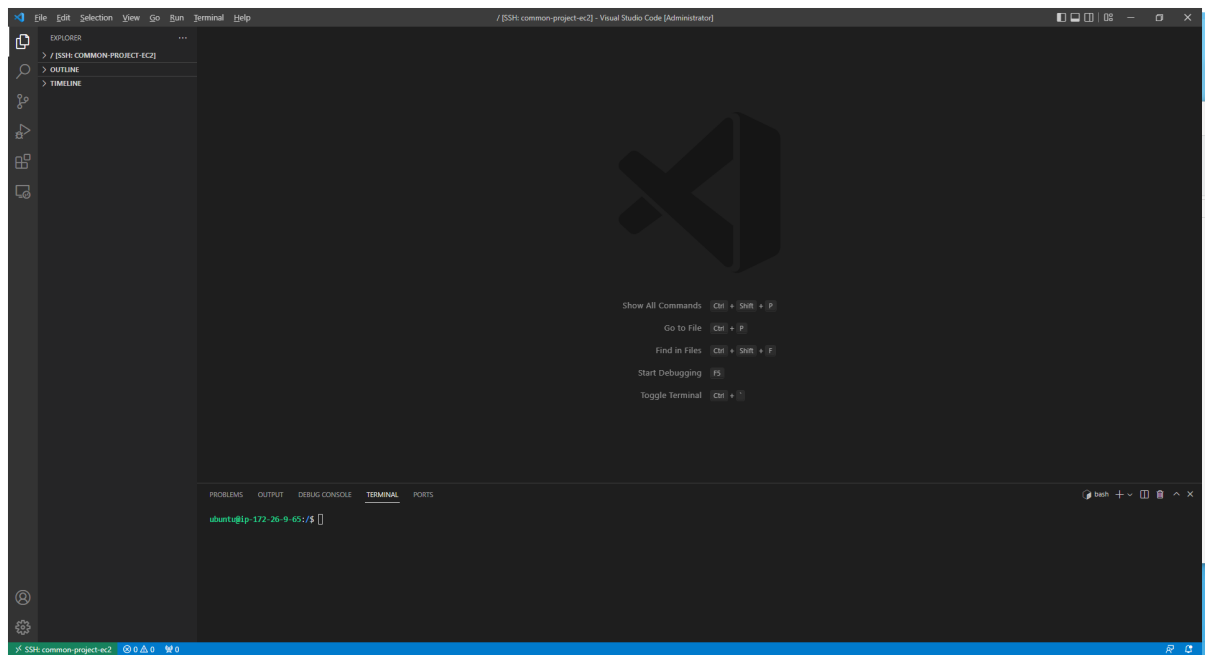


그림 11) Remote SSH에 연결된 모습

## 4.3. Jenkins 설정

### 4.3.1. 컨테이너 실행

우선 Jenkins 컨테이너를 서버에 올려줍니다. 명령어는 다음과 같습니다.

```
docker run --name jenkins-server -itd -p 8088:8080 -v /jenkins:/var/jenkins_home -u root jenkins/jenkins:lts
```

컨테이너를 올린 후 `{hostname}:8088` 에 접속한 후 조금 기다리면 패스워드를 입력해달라는 화면이 등장합니다.



# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



Continue

이 때 터미널에서 `docker logs jenkins-server` 를 입력하면 비밀번호를 확인할 수 있습니다.

```
*****
*****
*****

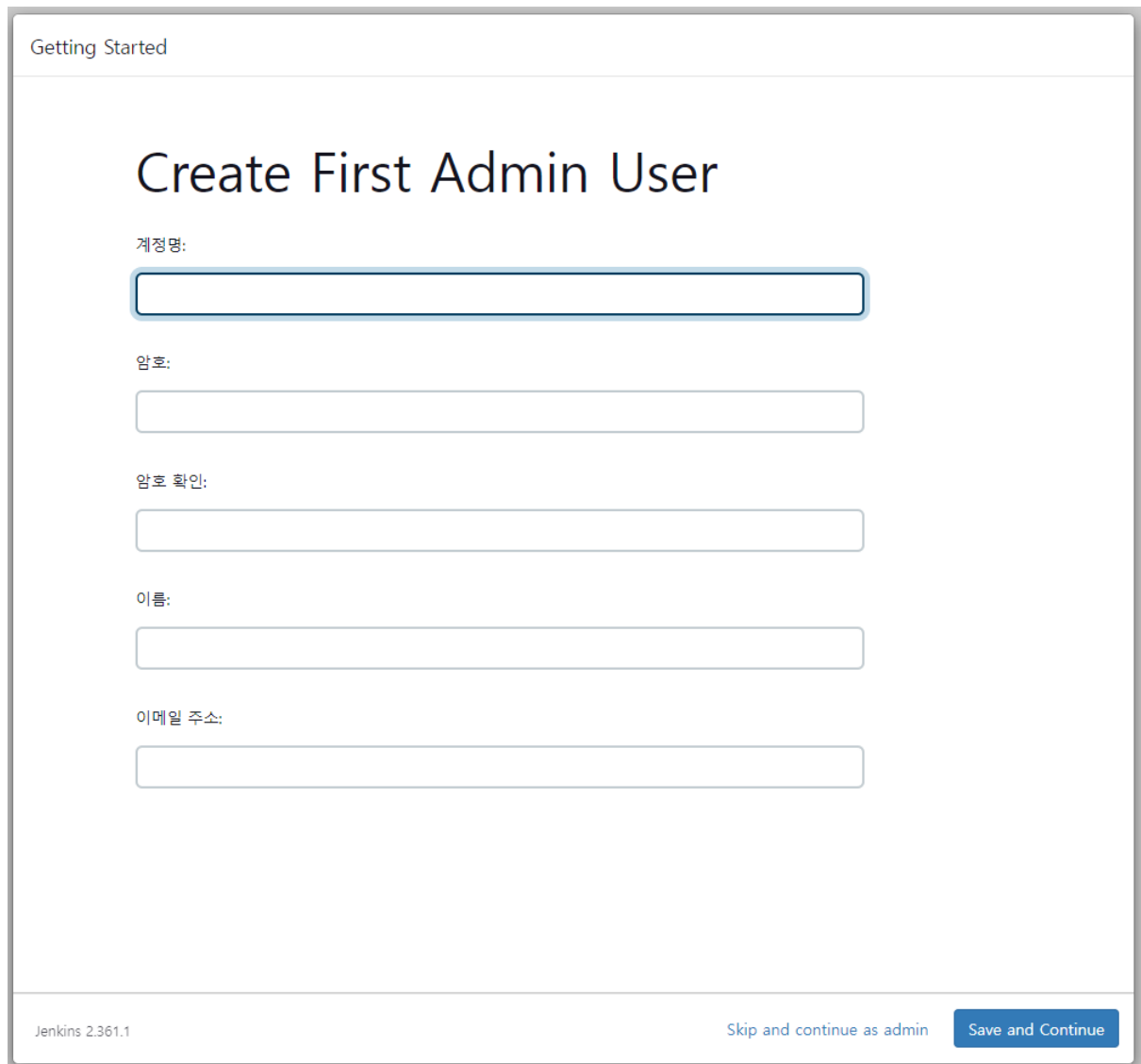
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

9598d100ec6f40e0a9d6474c1b008aa2

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

해당 비밀번호를 입력 후 **Install suggested plugins** 를 클릭한 후 플러그인을 다운로드 받아줍니다.

The image shows the 'Getting Started' screen of Jenkins 2.361.1. The main heading is 'Create First Admin User'. Below the heading, there are five input fields with labels in Korean: '계정명:' (Username), '암호:' (Password), '암호 확인:' (Confirm Password), '이름:' (Name), and '이메일 주소:' (Email address). At the bottom left, it says 'Jenkins 2.361.1'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

플러그인을 모두 다운로드 받으면, 유저를 등록하는 화면이 등장합니다. 모두 입력해주시고 **Save and Continue** 버튼을 클릭해 넘어갑니다.

그 후 마지막으로 바로 **Save and Finish** 버튼을 누르고 Jenkins 메인 화면으로 넘어갑니다.

### 4.3.2. 플러그인 설치

왼쪽 탭 **Jenkins 관리** 를 클릭 후 **플러그인 관리** 로 들어가줍니다. **설치 가능** 탭으로 이동해서 **GitLab** 과 **Publish Over SSH** 를 체크하고 다운로드 받아줍니다.

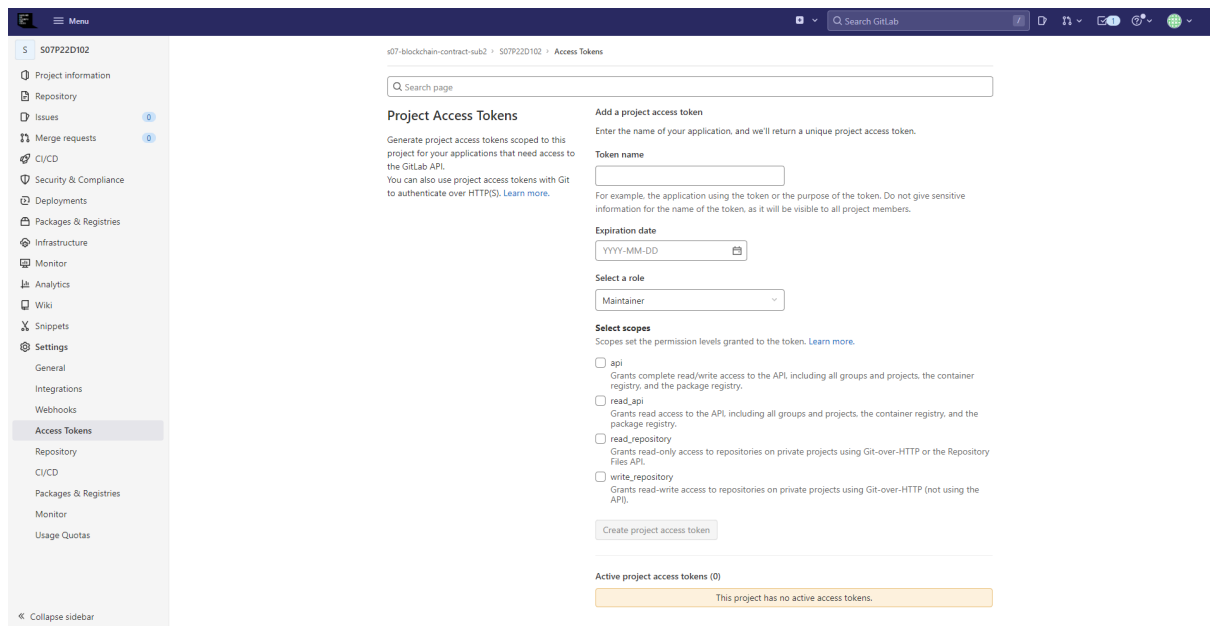
### 4.3.3. 시스템 설정

- Gitlab

왼쪽 탭 **Jenkins 관리** 를 클릭 후 **시스템 설정** 으로 들어갑니다.

쪽 내려가서 **Gitlab** 에서 **Gitlab Connection** 을 설정해줍니다. **Connection name** 은 원하시는 이름으로 설정하시면 되고 **Gitlab host URL** 은 `https://lab.ssafy.com`, Credential은 Gitlab 에서 발급한 Access Token으로 등록해줍니다.

Access Token을 획득하는 방법은 Gitlab Repository 왼쪽 탭에서 **Settings** → **Access Token** 에서 발급 받으실 수 있습니다.



## 4.4. MySQL 컨테이너 생성

다음 명령어로 MySQL 컨테이너를 생성합니다.

```
docker run --name mariadb-server -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mariadb:latest
```

workbench를 이용해 `{hostname}:3306` 으로 해당 DB 서버에 접속한 후 `create database indive` 를 통해서 데이터베이스를 생성해줍니다.

추가로 저희는 테스트용 MySQL 서버를 만들어두었습니다.

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 33066:3306 mysql:latest
```

## 4.5. Redis 컨테이너 생성

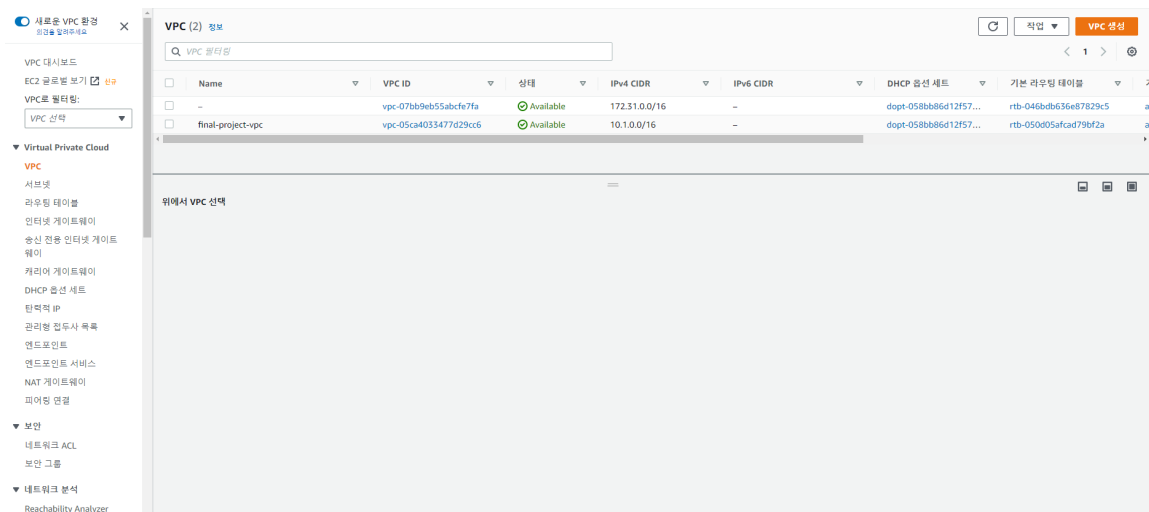
다음 명령어로 Redis 컨테이너를 생성합니다.

```
docker run --name redis-server -d -p 6379:6379 redis
```

# 5. AWS 인프라 구성

## 5.1. VPC 생성

1. `VPC` → `Virtual Private Cloud` → `VPC` → `VPC 생성`



2. VPC 설정

VPC > VPC > VPC 생성

## VPC 생성 정보

VPC는 AWS 클라우드의 격리된 부분으로서, Amazon EC2 인스턴스와 같은 AWS 객체로 채워집니다.

### VPC 설정

**생성할 리소스 정보**  
VPC 리소스 또는 VPC 및 기타 네트워킹 리소스만 생성합니다.

☒ VPC만
 ☐ VPC 등

**이름 태그 - 선택 사항**  
'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

final-project-vpc

**IPv4 CIDR 블록 정보**

☒ IPv4 CIDR 수동 입력  
☐ IPAM 할당 IPv4 CIDR 블록

**IPv4 CIDR**

10.1.0.0/16

**IPv6 CIDR 블록 정보**

☒ IPv6 CIDR 블록 없음  
☐ IPAM 할당 IPv6 CIDR 블록  
☐ Amazon 제공 IPv6 CIDR 블록  
☐ 내가 소유한 IPv6 CIDR

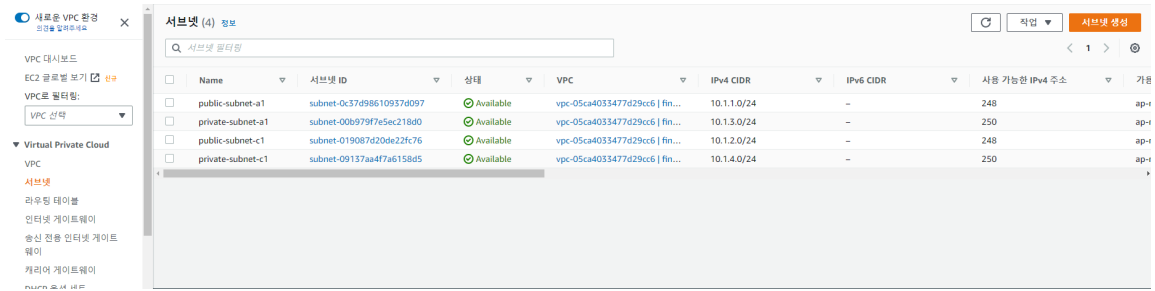
**테넌시 정보**

기본값

- 생성할 리소스 : VPC만
- 이름 태그 : 원하는 이름 태그를 설정한다. 나의 경우에는 final-project-vpc 로 설정했다.
- IPv4 CIDR 블록 : IPv4 CIDR 수동 입력 을 선택한다.
- IPv4 CIDR : 10.1.0.0/16 을 선택한다.
- IPv6 CIDR 블록 : IPv6 CIDR 블록 없음 을 선택한다.
- 테넌시 : 기본값

## 5.2. 서브넷 생성

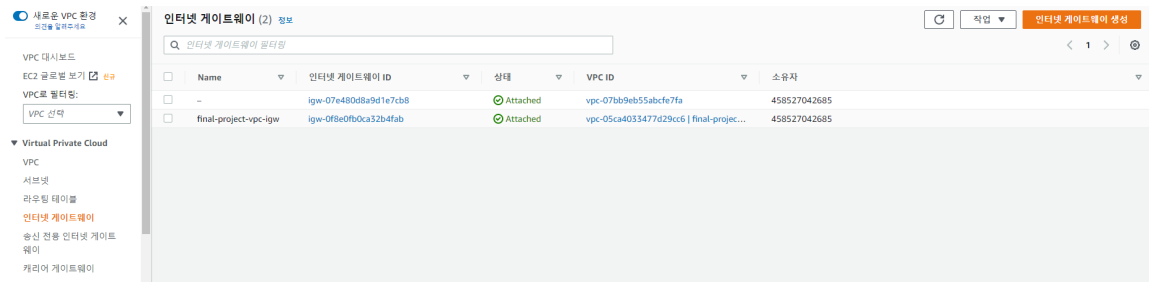
1. VPC → Virtual Private Cloud → 서브넷 → 서브넷 생성



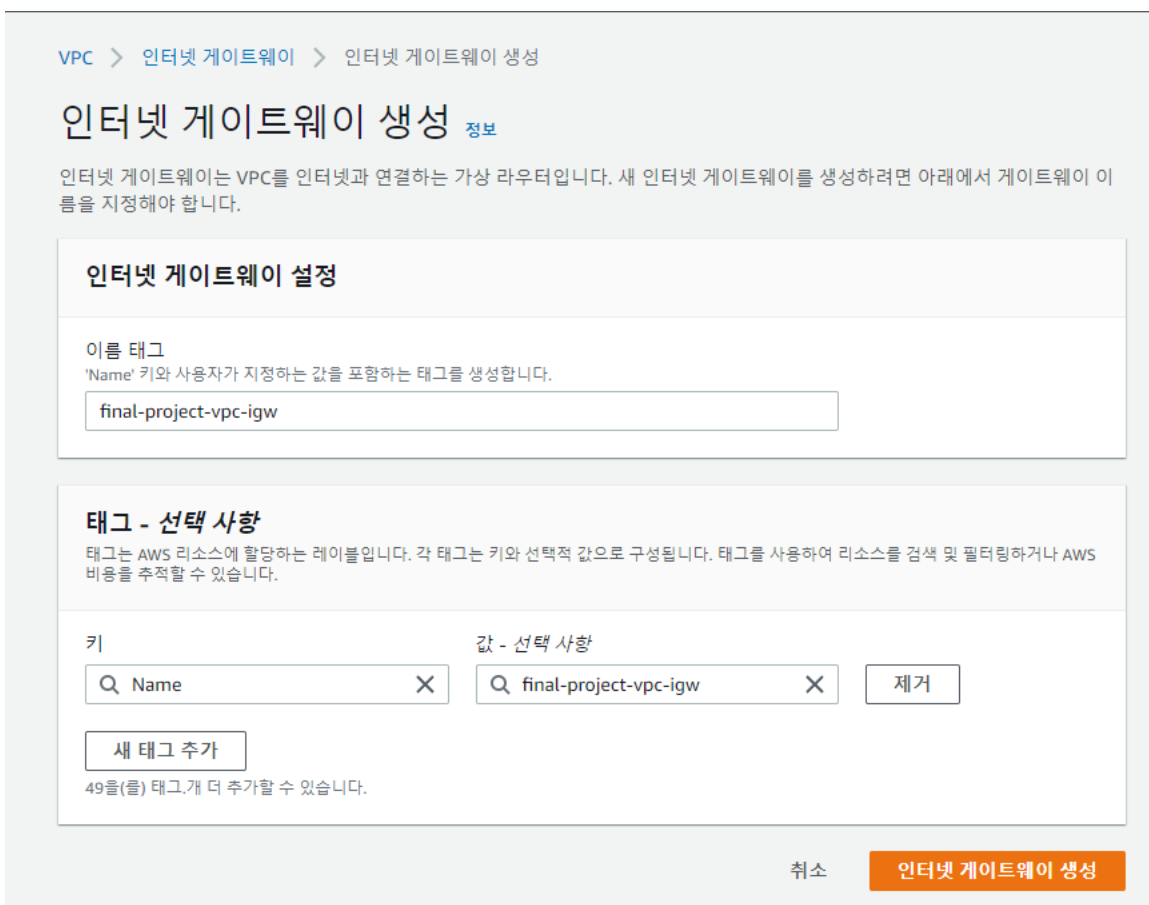
2. VPC ID에 아까 생성한 VPC를 선택한다.
3. 서브넷 설정에서는 총 4개의 서브넷을 생성할 것이다. (public-subnet 2개, private-subnet 2개)
  - a. 첫 번째 서브넷 (public-subnet)
    - 서브넷 이름 : `public-subnet-a1` 으로 입력한다.
    - 가용 영역 : `아시아 태평양 (서울) / ap-northeast-2a` 로 선택한다.
    - IPv4 CIDR 블록 : `10.1.1.0/24` 로 입력한다.
  - b. 두 번째 서브넷 (public-subnet)
    - 서브넷 이름 : `public-subnet-c1` 으로 입력한다.
    - 가용 영역 : `아시아 태평양 (서울) / ap-northeast-2c` 로 선택한다.
    - IPv4 CIDR 블록 : `10.1.2.0/24` 로 입력한다.
  - c. 세 번째 서브넷 (private-subnet)
    - 서브넷 이름 : `private-subnet-a1` 으로 입력한다.
    - 가용 영역 : `아시아 태평양 (서울) / ap-northeast-2a` 로 선택한다.
    - IPv4 CIDR 블록 : `10.1.3.0/24` 로 입력한다.
  - d. 네 번째 서브넷 (private-subnet)
    - 서브넷 이름 : `private-subnet-c1` 으로 입력한다.
    - 가용 영역 : `아시아 태평양 (서울) / ap-northeast-2c` 로 선택한다.
    - IPv4 CIDR 블록 : `10.1.4.0/24` 로 입력한다.
4. 작업이 완료되면 서브넷을 생성한다.

## 5.3. 인터넷 게이트웨이 생성

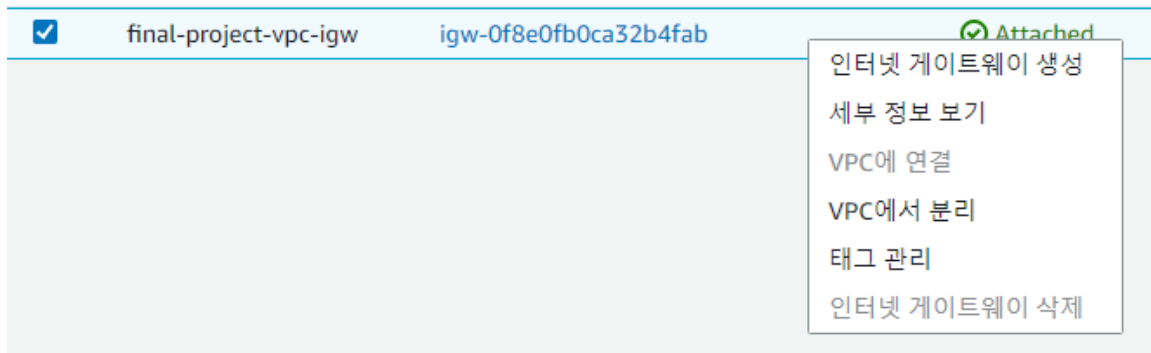
1. VPC → Virtual Private Cloud → 인터넷 게이트웨이 → 인터넷 게이트웨이 생성



2. 이름만 입력하고 바로 생성한다. 나의 경우에는 final-project-vpc-igw 로 입력했다.



3. 생성한 후 인터넷 게이트웨이 인스턴스에 우클릭하여 VPC에 연결 을 클릭한다.



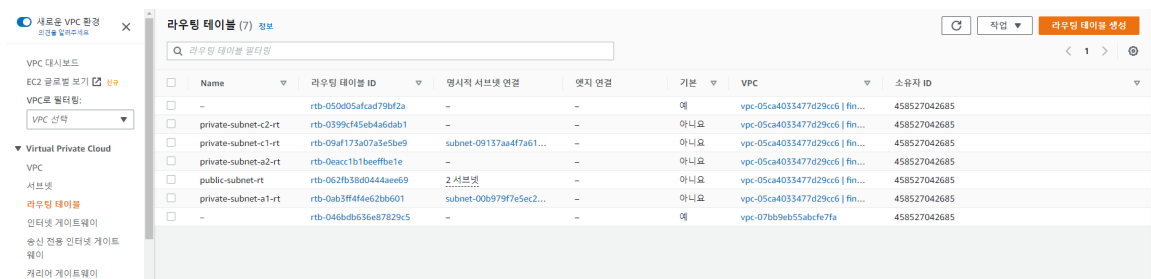
4. VPC에 위에서 생성한 VPC를 선택한 후 저장한다.

## 5.4. NAT 게이트웨이 생성

1. **VPC** → **Virtual Private Cloud** → **NAT 게이트웨이** → **NAT 게이트웨이 생성**
2. NAT 게이트웨이를 설정한다. 총 2개의 게이트웨이를 생성해야 한다. (public-subnet-a1 NAT, public-subnet-c1 NAT)
  - 이름 : 원하는 이름을 입력한다. 나의 경우에는 **nat-gw-a1** 으로 입력했다.
  - 서브넷 : **public-subnet-a1**
  - 연결 유형 : 퍼블릿
  - 탄력적 IP 할당 ID : 오른쪽에 있는 **탄력적 IP 할당** 버튼을 클릭한다.
3. **public-subnet-c1** 의 NAT 게이트웨이도 생성해준다.

## 라우팅 테이블 생성

1. **VPC** → **Virtual Private Cloud** → **라우팅 테이블** → **라우팅 테이블 생성**



2. 총 3개의 라우팅 테이블을 생성해야 한다. (**public-subnet-rt**, **private-subnet-a1-rt**, **private-subnet-c1-rt**)
  - a. public-route-table



- i. 이름과 VPC를 입력한다. VPC는 생성한 VPC를 선택한다.
  - ii. 라우팅 테이블 생성 후 라우팅 탭에서 우측에 있는 **라우팅 편집** 을 클릭한다.
  - iii. **라우팅 추가** 버튼을 클릭한다. 첫번째 대상은 **0.0.0.0/0** , 두번째 대상은 **인터넷 게이트웨이** 선택 후 위에서 생성한 인터넷 게이트웨이로 설정한다.
  - iv. 저장 후 서브넷 연결 탭을 클릭한다. 그 후 **서브넷 연결 편집** 버튼을 클릭한다.
  - v. **public-subnet-a1** 과 **public-subnet-c1** 을 선택한 후 저장한다.
- b. private-route-table
- i. 우선 위와 같이 라우팅 테이블을 생성한다. 참고로 private-route-table은 a1, c1 각각 총 두 개 생성해야 하며 우선 **private-subnet-a1** 에 대한 라우팅 테이블을 먼저 생성한다.
  - ii. 이번에는 **라우팅 편집** 에서 첫번째 대상은 **0.0.0.0/0** , 두번째 대상은 **NAT 게이트웨이** 선택 후 **nat-gw-a1** 을 선택한다.
  - iii. 서브넷 연결은 **private-subnet-a1** 을 선택해준다.
  - iv. 저장한다. 위의 과정을 **private-subnet-c1** 에 대해서도 똑같이 해준다.

## 5.5. public-subnet EC2 인스턴스 생성

1. **EC2** → **인스턴스** → **인스턴스** → **인스턴스 시작**
  - 총 2개의 public 인스턴스를 만들 것이다. 우선 public-subnet-a1의 인스턴스를 만들어보도록 하겠다.
2. 이름 및 태그에 원하는 이름을 설정한다. 나의 경우에는 **public-ec2-a1** 으로 설정
3. 애플리케이션 및 이미지(Amazon Machine Image)
  - **Quick Start** → **Ubuntu**
  - AMI를 **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** 으로 설정
  - 아키텍처는 **64비트(x86)**
4. 인스턴스 유형은 **t2.micro**
5. 키 페어(로그인)은 **새 키페어 생성** 버튼을 클릭하고 .pem으로 만든다. 이 키는 ssh로 인스턴스에 원격 접속할 때 사용된다.
6. 네트워크 설정에서 우측에 있는 **편집** 버튼을 누른다.
  - VPC : 이전에 생성한 VPC를 선택한다. 나의 경우에는 **final-project-vpc** 이다.

- 서브넷 : `public-subnet-a1`
- 퍼블릭 IP 자동 할당 : `비활성화` → 추후에 Elastic IP로 퍼블릭 IP를 할당할 것이다.
- 보안 그룹 : 새로 생성한다. 인바운드 규칙은 총 3개이다.

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0) 제거

유형 정보	프로토콜 정보	포트 범위 정보
ssh	TCP	22

소스 유형 정보	원본 정보	설명 - optional 정보
위치 무관	Q CIDR, 접두사 목록 또는 보안 그룹 ID 0.0.0.0/0 X	예: 관리자 데스크톱용 SSH

▼ 보안 그룹 규칙 2 (TCP, 80, 0.0.0.0/0) 제거

유형 정보	프로토콜 정보	포트 범위 정보
HTTP	TCP	80

소스 유형 정보	원본 정보	설명 - optional 정보
위치 무관	Q CIDR, 접두사 목록 또는 보안 그룹 ID 0.0.0.0/0 X	예: 관리자 데스크톱용 SSH

▼ 보안 그룹 규칙 3 (TCP, 443, 0.0.0.0/0) 제거

유형 정보	프로토콜 정보	포트 범위 정보
HTTPS	TCP	443

소스 유형 정보	원본 정보	설명 - optional 정보
위치 무관	Q CIDR, 접두사 목록 또는 보안 그룹 ID 0.0.0.0/0 X	예: 관리자 데스크톱용 SSH

- 보안 그룹 이름 : `public-ec2-sg`
- 설명 : 원하는 문장을 작성한다.
- 인바운드 보안 그룹 규칙
  1. 유형 : `ssh`, 소스 유형 : `위치 무관` 혹은 `Anywhere` 로 한다.
  2. 유형 : `HTTP`, 소스 유형 : `위치 무관` 혹은 `Anywhere` 로 한다.
  3. 유형 : `HTTPS`, 소스 유형 : `위치 무관` 혹은 `Anywhere` 로 한다.

7. 완료 후 인스턴스를 생성한다.
8. `public-subnet-c1`에 대한 ec2 인스턴스도 생성해준다. 6번 과정에서 서브넷을 `public-subnet-c1`으로 해준다.

## 5.6. Elastic IP 할당

1. `EC2` → `네트워크 및 보안` → `탄력적 IP` → `탄력적 IP 주소 할당`
2. 탄력적 IP 주소 설정
  - 네트워크 경계 그룹 : `ap-northeast-2`
  - 퍼블릭 IPv4 주소 풀 : `Amazon의 IPv4 주소 풀`
3. 태그
  - Key : `Name`
  - Value : `eip-public-ec2-a1`
4. 생성한 IP 주소를 클릭하고 `작업` → `탄력적 IP 주소 연결`
5. 리소스 유형 : `인스턴스`
6. 인스턴스 : 위에서 생성한 ec2 인스턴스를 선택한다.

## 5.7. 배포용 커스텀 AMI 생성

EC2 인스턴스를 임시로 새로 생성해서 ssh로 원격 접속한다.

### CodeDeploy Agent 설치

1. 우선 다음 명령을 차례로 입력한다.

```
sudo apt-get update
```

```
sudo apt install ruby-full
```

```
sudo apt install wget
```

2. 전부 설치가 끝나면 우분투 터미널에서 다음 명령을 입력한다.

```
cd /home/ubuntu
```

3. 다음 명령을 입력해준다.

```
wget https://aws-codedeploy-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/install
```

4. install 폴더가 생성되면 실행 권한을 부여한다.

```
chmod +x ./install
```

5. Ubuntu 20.04에서 최신 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행한다.

```
sudo ./install auto > /tmp/logfile
```

6. 제대로 설치되었는지 확인한다.

```
sudo service codedeploy-agent status
```

```
ubuntu@ip-10-1-1-214:~$ sudo service codedeploy-agent status
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
   Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
   Active: active (running) since Sun 2022-11-20 10:33:25 UTC; 48s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 2 (limit: 2351)
   Memory: 63.8M
   CGroup: /system.slice/codedeploy-agent.service
           └─3887 codedeploy-agent: master 3887
             └─3889 codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 3887

Nov 20 10:33:25 ip-10-1-1-214 systemd[1]: Starting LSB: AWS CodeDeploy Host Agent...
Nov 20 10:33:25 ip-10-1-1-214 codedeploy-agent[3881]: Starting codedeploy-agent:
Nov 20 10:33:25 ip-10-1-1-214 systemd[1]: Started LSB: AWS CodeDeploy Host Agent.
```

## Docker 설치

1. 오래된 버전 삭제하기

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

## 2. repository를 설정한다.

```
sudo apt-get update
```

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

## 3. Docker의 Official GPG Key 를 등록한다.

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

## 4. Stable Repository를 등록한다.

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

## 5. Docker Engine을 설치한다.

```
sudo apt-get update
```

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

## 6. 설치가 완료되었는지 확인한다.

```
docker --version
```

## 7. sudo 명령어 없이 도커를 사용하기 위해 도커 그룹에 유저를 추가한다.

```
sudo usermod -aG docker $USER
```

## 도커파일 생성

1. 도커 파일과 백엔드 배포 파일이 저장될 폴더를 생성한다.

```
mkdir ~/server
```

```
cd ~/server
```

2. vim 편집기로 도커 파일을 생성한다.

```
vim Dockerfile
```

3. **i** 를 누르고 다음 코드를 입력한다.

```
FROM openjdk:17-jdk-alpine
ENV ARTIFACT_NAME=shall-we-meet-then-0.0.1-SNAPSHOT.jar

COPY $ARTIFACT_NAME .

EXPOSE 80
ENTRYPOINT exec java -jar ${ARTIFACT_NAME} --spring.profiles.active=prod
```

4. **esc** 를 누르고 **:wq** 를 입력한 후 저장한다.

## AMI 생성하기

1. 임시로 생성한 인스턴스에 우클릭한다.
2. **이미지 및 템플릿** → **이미지 생성**

**인스턴스 (1/7) 정보**

Find 인스턴스 by attribute or tag (case-sensitive)

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
public-ec2-c1	i-011bd4d712917e937	실행 중	t2.small	2/2개 검사
asg-ec2	i-0316ac9c4788a412c	실행 중	t2.small	2/2개 검사
public-ec2-a1	i-0d75a260dde574658	실행 중	t2.small	2/2개 검사
asg-ec2	i-039bed9c1629a8c26	실행 중	t2.small	2/2개 검사
temp	i-06a2f4a31979c29f2	종료됨	t2.small	-
<b>temp</b>	<b>i-0776eb705ebd5ba07</b>	<b>실행 중</b>	<b>t2.small</b>	<b>2/2개 검사</b>

**인스턴스: i-0776eb705ebd5ba07(temp)**

세부 정보 | 보안 | 네트워킹 | 스토리지 | 상태 연결 | 태그

▼ 인스턴스 요약 정보

인스턴스 ID  
i-0776eb705ebd5ba07 (temp)

IPv6 주소  
-

호스트 이름 유형  
IP 이름: ip-10-1-1-214.ap-northeast-2.compute.internal

프라이빗 리소스 DNS 이름 응답  
IPv4(A)

자동 할당된 IP 주소  
3.36.57.119 [퍼블릭 IP]

인스턴스 시작  
템플릿으로 인스턴스 시작  
서버 마이그레이션

인스턴스 중지  
인스턴스 시작  
인스턴스 재부팅  
인스턴스 최대 절전 모드  
인스턴스 종료  
인스턴스 설정

네트워킹  
보안

이미지 및 템플릿  
모니터링 및 문제 해결

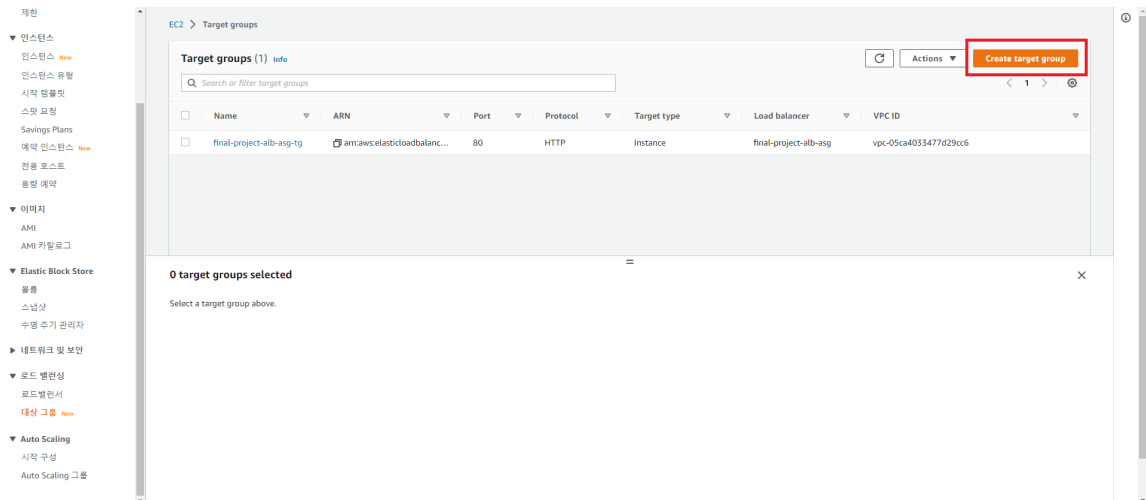
인스턴스 생성  
인스턴스에서 템플릿 생성  
이런 방식으로 더 많이 시작

VPC  
vpc-05ca4033477d29cc6 (final-pr

3. 이미지 이름과 이미지 설명을 입력하고 이미지를 생성한다.

## 5.8. 로드밸런서 타겟 그룹 생성

1. EC2 → 로드 밸런싱 → 대상 그룹 → 대상 그룹 생성



2. **Basic configuration** 은 **Instances** 로 선택한다.

### Basic configuration

Settings in this section cannot be changed after the target group is created.

Choose a target type

☒ **Instances**

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

☐ **IP addresses**

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

☐ **Lambda function**

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

☐ **Application Load Balancer**

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

3. **Target group name** 은 원하는 이름으로 한다. 나는 **final-project-alb-acg-tg** 로 했다.
4. **Protocol** 은 **HTTP** , **Port** 는 **80** 으로 한다.
  - 참고로 대상 그룹은 로드밸런서와 통신하기 때문에 SSL 인증서를 적용하지 않아도 된다. 클라이언트와 서버 간 구간 암호화는 로드밸런서에 적용할 것이다.
5. **VPC** 는 기존에 생성한 **final-project-vpc** 로 한다.



6. **Protocol Version**은 **HTTP1**으로 한다.
7. **Health Check**에서 **Advanced health check settings**를 눌러 **Success Code**에 **200,401**로 변경해준다.
  - 우리 서비스는 JWT를 이용하기 때문에 헬스 체크용 요청이 401로 반환된다. 추후에 200으로 받을 수 있도록 리팩토링하자.
8. **Register targets**에서 아무것도 건드리지 않고 바로 **Create target group**을 클릭한다.

## 5.9. 로드밸런서 생성

1. **EC2** → **로드 밸런싱** → **로드밸런서** → **로드 밸런서 생성**
2. **Application Load Balancer**로 선택

EC2 > Load balancers > Select load balancer type

### Select load balancer type

A complete feature-by-feature comparison along with detailed highlights is also available. [Learn more](#)

#### Load balancer types

#### Application Load Balancer Info

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

#### Network Load Balancer Info

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low

#### Gateway Load Balancer Info

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

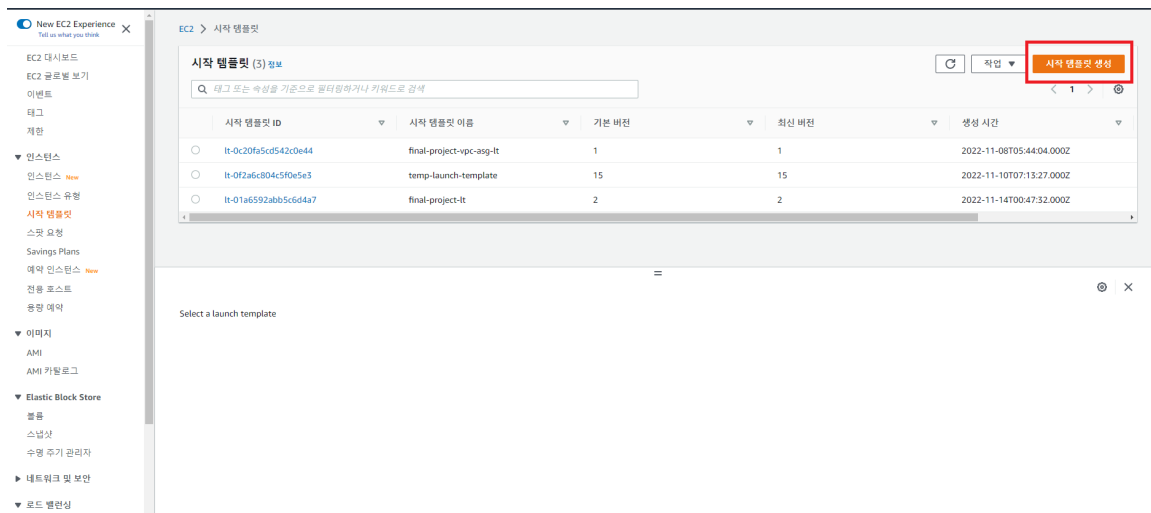
Create

3. **Load balancer name**은 원하는 이름으로 설정한다. 나는 **final-project-alb-acg**로 설정했다.
4. **Scheme**은 **Internet-facing**으로 설정한다.

5. IP address type 은 IPv4 로 설정한다.
6. VPC 는 본인이 생성한 VPC로 설정한다. 나의 경우에는 final-project-vpc 로 설정했다.
7. Mappings 는 ap-northeast-2a , ap-northeast-2c 둘 다 체크해준다. 그 후 Subnet은 각각 private-subnet-a1 , private-subnet-c1 으로 설정한다.
8. Security Group 은 새로 생성해준다. 인바운드 규칙은 SSH , HTTP , HTTPS 를 모든 대상으로 열어준다.
9. Listeners and routing 은 우선 HTTP만 등록해준다. 추후에 ACM 설정 후 HTTPS로 등록한다.
10. Tag 에 Key에 Name, Value에 Load balancer name에 입력한 값을 똑같이 입력한다.
11. Summary 확인 후 로드밸런서를 생성한다.

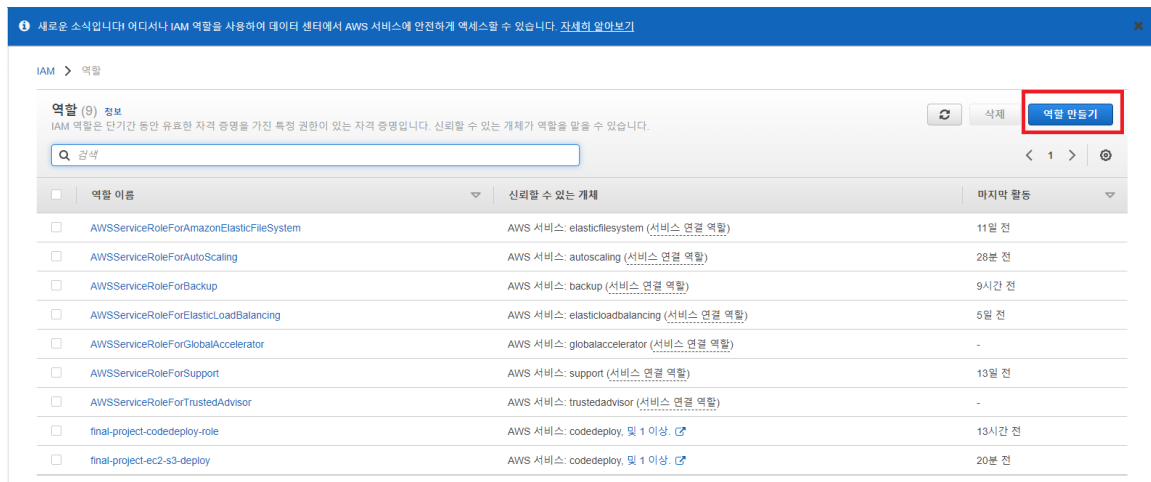
## 5.10. Auto Scaling 시작 템플릿 생성

1. EC2 → 인스턴스 → 시작 템플릿 → 시작 템플릿 생성



2. 시작 템플릿 이름은 원하는 걸로 입력한다. 나의 경우에는 final-project-lt 로 했음
3. AMI는 오토 스케일링을 위해 생성한 이미지를 사용한다. 나의 경우에는 deploy-server-image 이다.
4. 인스턴스 유형은 t2.micro 로 한다. (프로젝트 할 때는 교보재로 받아 t2.small 을 이용했음)
5. private-subnet 에 접속할 전용 pem 키를 하나 생성한다. public-subnet 에 있는 인스턴스에 넣어두고 ssh로 접속할 수 있다.

- 네트워크 설정에서 **서브넷**은 따로 설정하지 않는다.
- 보안 그룹은 **private-subnet**용 보안 그룹을 사용한다. 나의 경우에는 **private-ec2-sg**임.
- 스토리지(볼륨)은 그대로 둔다.
- 리소스 태그도 따로 포함하지 않는다.
- 고급 세부 정보** → **IAM 인스턴스 프로파일**에서 새로운 IAM 프로파일을 생성한다.



- 이 작업은 추후에 CodeDeploy를 이용해 CI/CD 환경을 구축할 때 EC2가 S3에 있는 배포 파일을 다운로드 하기 위해 사용되는 IAM 역할을 만드는 작업이다.

- 신뢰할 수 있는 엔티티 유형에 **AWS 서비스**를 선택하고 사용 사례에 **EC2**를 체크한다.

## 신뢰할 수 있는 엔티티 선택 정보

### 신뢰할 수 있는 엔티티 유형

☒ **AWS 서비스**  
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **AWS 계정**  
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔티티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **웹 자격 증명**  
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ **SAML 2.0 연동**  
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ **사용자 지정 신뢰 정책**  
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

### 사용 사례

EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

#### 일반 사용 사례

- ☒ **EC2**  
Allows EC2 instances to call AWS services on your behalf.
- ☐ **Lambda**  
Allows Lambda functions to call AWS services on your behalf.

다른 AWS 서비스의 사용 사례:

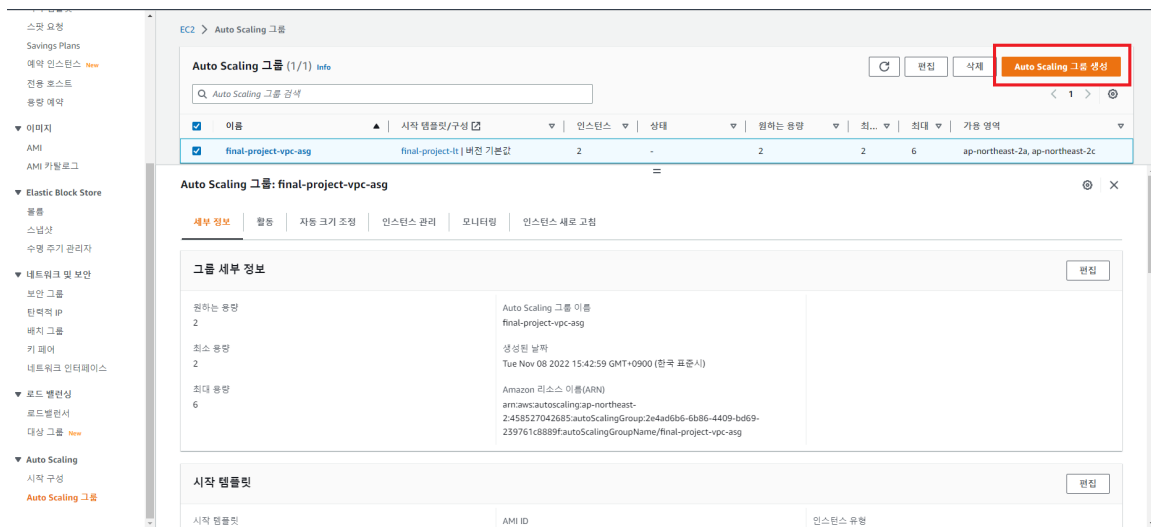
사용 사례를 조회할 서비스 선택

- 권한 정책에서 **AmazonEC2RoleforAWSCodeDeploy**를 추가한다.

- 역할 이름을 입력하고 역할을 생성한다. 나의 경우에는 `final-project-ec2-s3-deploy` 라고 이름 지었다.
- 생성 후 다시 시작 템플릿 생성 페이지로 돌아와서 `IAM` `인스턴스 프로파일` 옆에 있는 새로고침 이미지를 클릭하고, 방금 생성한 `final-project-ec2-s3-deploy` 프로파일로 설정한 후 저장한다.

## 5.11. Auto Scaling 그룹 생성

- `EC2` → `Auto Scaling` → `Auto Scaling 그룹` → `Auto Scaling 그룹 생성`



- Auto Scaling 그룹 이름을 설정한다. 나의 경우에는 `final-project-vpc-asg` 라고 이름 지었다.
- 시작 템플릿은 위에서 생성한 `final-project-lt` 를 선택한다.
- VPC는 생성했던 VPC를 이용한다. 나의 경우에는 `final-project-vpc` 이다.
- 가용 영역 및 서브넷은 `private-subnet-a1` , `private-subnet-c1` 로 설정한다.
- 로드 밸런싱은 `기존 로드 밸런서에 연결` 선택 후 `로드 밸런서 대상 그룹에서 선택` → 예전에 생성했던 대상 그룹인 `final-project-alb-asg-tg` 를 선택한다.
- 그룹 크기는 원하는 용량 2, 최소 용량 2, 최대 용량 3으로 설정한다. 최소 용량은 2로 하고 나머지는 입맛에 맞추면 된다.
- 크기 조정 정책은 `대상 추적 크기 조정 정책` 을 선택한다.
  - 지표 유형 : `평균 CPU 사용률`
  - 대상 값 : `80`

- 인스턴스 요구 사항 : 300

→ 위와 같이 설정하면 대상 그룹에 속한 인스턴스의 평균 CPU 사용률이 80%가 넘어서면 자동으로 스케일 아웃되고, 사용률이 줄어들면 스케일 인된다.

9. 나머지는 필요에 따라 선택하고 Auto Scaling 그룹을 생성한다.

## 6. AWS 프론트 서버 CI/CD 환경 구축

### 6.1. 배포용 S3 버킷 생성

1. S3 → 버킷 → 버킷 만들기
2. 일반 구성
  - 버킷 이름 : 원하는 걸로 설정한다.
  - AWS 리전 : 아시아 태평양(서울) ap-northeast-2
3. 객체 소유권은 ACL 비활성화됨(권장)
4. 이 버킷의 퍼블릭 액세스 차단 설정은 모든 퍼블릭 액세스 차단을 체크 해제한다.

## 이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

### ☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

#### ☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

#### ☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

#### ☐ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

#### ☐ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.



모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

## 5. 버킷 버전 관리는 **비활성화**

## 6. 태그

- 키 : Name
- 값 : 원하는 이름

## 7. 기본 암호화는 **비활성화**

## 8. 버킷 생성 후 해당 버킷 클릭 → 권한 → 버킷 정책 에서 우측에 있는 편집 버튼을 누른다.

## 9. 버킷 정책 편집에서 우측에 있는 정책 생성기 를 누른다.

## 10. 다음과 같이 입력한다.



## AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to [Amazon Web Services \(AWS\)](#) products. policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Queue Policy](#).

Select Type of Policy S3 Bucket Policy ▼

### Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service Amazon S3 ▼ ☐ All Services ('\*')

Use multiple statements to add permissions for more than one service.

Actions 1 Action(s) Selected ▼ ☐ All Actions ('\*')

Amazon Resource Name (ARN)

ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}.  
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)


- Select Type of Policy : S3 Bucket Policy
- Effect : Allow
- Principal : \*
- AWS Service : Amazon S3
- Actions : GetObject
- Amazon Resource Name (ARN) : 이전 정책 편집 페이지에서 버킷 ARN 을 복사한다. 추가로 /\* 를 붙여준다. 즉, 다음과 같은 형태가 된다. 버킷 ARN/\*

11. 정책을 생성한 후 나오는 JSON Document를 복사한 뒤 이전 버킷 정책 편집 페이지에 입력한다.

## 버킷 정책

JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다. 자세히 알아보기 [\[링크\]](#)

버킷 ARN

 arn:aws:s3:::final-project-codedeploy-bucket

정책

```
1  {
2    "Version": "2012-10-17",
3    "Id": "Policy1668403726451",
4    "Statement": [
5      {
6        "Sid": "Stmt1668403725649",
7        "Effect": "Allow",
8        "Principal": "*",
9        "Action": "s3:GetObject",
10       "Resource": "arn:aws:s3:::final-project-codedeploy-bucket/*"
11     }
12   ]
13 }
```

12. **속성** 탭에서 맨 아래에 있는 **정적 웹 사이트 호스팅** 에서 편집 버튼을 누릅니다.

- 정적 웹 사이트 호스팅 : **활성화**
- 호스팅 유형 : **정적 웹 사이트 호스팅**
- 인덱스 문서 : **index.html**
- 나머지는 그대로 둡니다.

## 6.2. Cloudfront 배포

1. **Cloudfront** → **배포** → **배포 생성**
2. 원본 도메인에 위에서 생성한 S3 버킷을 선택합니다.
3. 원본 경로는 그대로 둡니다.
4. 이름에는 원하는 이름을 입력합니다.
5. 뷰어 프로토콜 정책은 **Redirect HTTP to HTTPS**
6. 설정에서 사용자 정의 SSL 인증서는 예전에 생성한 ACM SSL 인증서를 선택합니다.



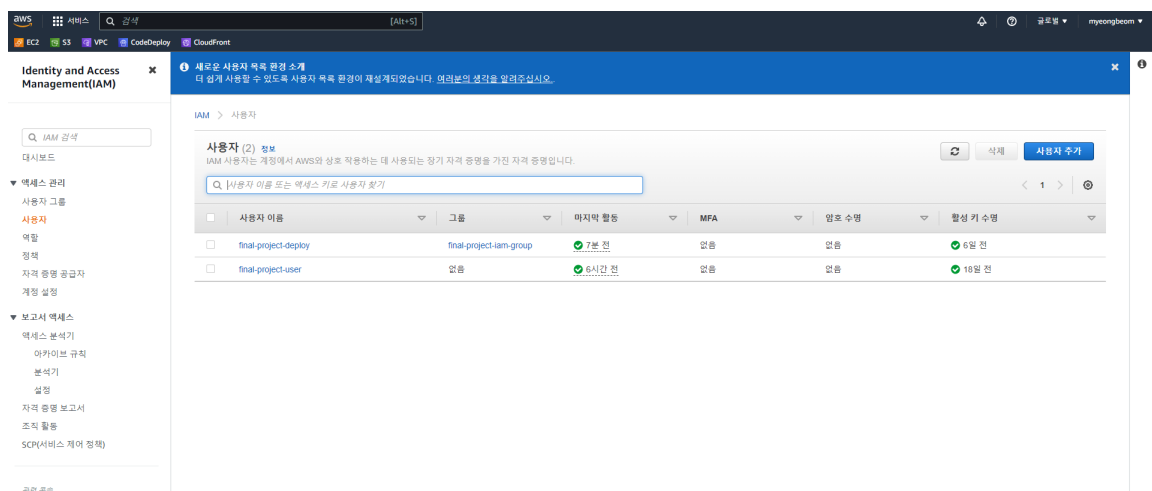
7. 기본값 루트 객체는 `index.html` 을 입력해줍니다.

8. 나머지는 전부 그대로 둡니다.

## 6.3. Cloudfront 무효화 설정

1. Cloudfront 무효화를 위한 설정을 해야한다.

1. Cloudfront의 invalidation을 생성하기 위한 IAM 사용자를 생성해야한다. IAM → 사용자 → 사용자 추가



2. 사용자 이름을 아무거나 입력하고 액세스 키는 프로그래밍 방식 액세스를 선택한다.

### 사용자 추가

1 2 3 4 5

#### 사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. 자세히 알아보기

사용자 이름\* cloudfront-invalidation-user

+ 다른 사용자 추가

#### AWS 액세스 유형 선택

이러한 사용자가 주로 AWS에 액세스하는 방법을 선택합니다. 프로그래밍 방식의 액세스만 선택하면 사용자가 위임된 역할을 사용하여 콘솔에 액세스하는 것을 방지할 수 없습니다. 액세스 키와 자동 생성된 암호가 마지막 단계에서 제공됩니다. 자세히 알아보기

- AWS 자격 증명 유형 선택\* ☒ 액세스 키 – 프로그래밍 방식 액세스  
AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키 을(를) 활성화합니다.
- ☐ 암호 – AWS 관리 콘솔 액세스  
사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호 을(를) 활성화합니다.

3. 기존 정책 직접 연결 → 정책 생성 버튼을 누른다.

4. 시각적 편집기 에서 다음과 같이 설정한다.
  - 서비스 : cloudfront
  - 작업 : CreateInvalidation
  - 리소스 : ARN 추가 → CloudFront\_distribution ARN 지정 에 무효화할 cloudfront의 ARN 입력
  - 요청 조건 : 그대로
5. 나머지 입력 후 정책 생성

## 6.4. 젠킨스 컨테이너에 aws cli 설치

1. 젠킨스 컨테이너의 sh로 들어간다. → `docker exec -it {젠킨스 컨테이너 이름} sh`
2. 우선 jenkins container에 aws cli를 설치해준다.

```
apt-get update
apt install python3-pip -y
pip3 install awscli
```

3. awscli의 유저 설정을 해준다. → `aws configure`

```
AWS Access Key ID [*****TUMF]: {아까 생성한 IAM 사용자 access key}
AWS Secret Access Key [*****uAgT]: {아까 생성한 IAM 사용자 secret key}
Default region name [ap-northeast-2]: ap-northeast-2
Default output format [json]: json
```

## 6.5. Cloudfront CI/CD 환경 구축 (리액트 배포)

1. Jenkins에 접속한다.
2. 플러그인 중 NodeJS 와 S3 publisher 를 설치해준다.



S3 publisher 0.12.3436.v674b\_46258039

Artifact Uploaders

aws

This is a plugin to upload files to Amazon S3 buckets.

3. **NodeJS** 는 GlobalConfiguration으로 가서 **Install Automatically** 체크 후 저장해준다.  
그리고 S3 publisher는 시스템 설정에서 **Amazon S3 profiles** 에서 IAM에서 생성한 access key와 secret key를 등록해야한다. **profile name** 은 aws-s3-profile로 한다.
4. **Pipeline** 으로 아이템을 생성한다.

Enter an item name

> Required field

- Freestyle project**  
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.
- Maven project**  
Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
다양한 환경에서의 테스트, 플러를 특정 빌드, 기타 동등 적인 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

OK

If you don't want to create a new item from other existing, you can use this option:

5. **빌드 유발** 에서 **Build when a change is pushed to GitLab. GitLab webhook URL: ~~** 선택

**빌드 유발**

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://k7d105.p.ssafy.io:8088/project/cloudfront-deploy-project> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☐ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

6. **Gitlab** 에서 웹 훅 설정도 해주기! URL은 **Gitlab webhook URL** 뒤에 URL 주소를 적어주고 Secret token은 더 보기 후 밑으로 내려가면 생성할 수 있다.

Search page

## Webhooks

[Webhooks](#) enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

### URL

 http://k7d105.p.ssafy.io:8088/project/cloudfront-deploy-project

URL must be percent-encoded if it contains one or more special characters.

### Secret token

 **secret**

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

## 7. Pipeline 스크립트를 입력한다.

```
pipeline {
  agent any
  tools {nodejs "NodeJS"}

  stages {
    stage('prepare') {
      steps {
        git credentialsId: 'this', url: 'https://lab.ssafy.com/mungmnb777/
aws-cicd-test-repo.git'
      }
    }

    stage('build') {
      steps {
        dir('FE/shall-we-meet-then') {
          sh 'npm install'
          sh 'CI=false npm run build'
        }
      }
    }

    stage('upload S3') {
      steps {
        s3Upload consoleLogLevel: 'INFO',
dontSetBuildResultOnFailure: false,
dontWaitForConcurrentBuildCompletion: false,
entries: [[
          bucket: 'final-project-react-deploy-kr',
          excludedFile: '',
          flatten: false,
          gzipFiles: false,
          keepForever: false,
          managedArtifacts: false,
          noUploadOnFailure: true,
          selectedRegion: 'ap-northeast-2',
          showDirectlyInBrowser: false,
          sourceFile: 'FE/shall-we-meet-then/build/**',
          storageClass: 'STANDARD',
          uploadFromSlave: false,
          useServerSideEncryption: false]],
pluginFailureResultConstraint: 'FAILURE',
```

```

        profileName: 'aws-s3-profile',
        userMetadata: []
    }
}

stage('cloudfront invalidation') {
    steps {
        sh 'aws cloudfront create-invalidation --distribution-id E325L508
95MPE4 --paths "/"'
    }
}
}
}

```

## 7. AWS 백엔드 스프링부트 서버 CI/CD 환경 구축

### 7.1. 배포용 S3 버킷 생성

1. S3 → 버킷 → 버킷 만들기
2. 일반 구성
  - 버킷 이름 : 원하는 걸로 설정한다.
  - AWS 리전 : 아시아 태평양(서울) ap-northeast-2
3. 객체 소유권은 ACL 비활성화됨(권장)
4. 이 버킷의 퍼블릭 액세스 차단 설정은 모든 퍼블릭 액세스 차단을 체크 해제한다.

## 이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

### ☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

#### ☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

#### ☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

#### ☐ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

#### ☐ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.



모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

## 5. 버킷 버전 관리는 **비활성화**

## 6. 태그

- 키 : Name
- 값 : 원하는 이름

## 7. 기본 암호화는 **비활성화**

## 8. 버킷 생성 후 해당 버킷 클릭 → 권한 → 버킷 정책 에서 우측에 있는 편집 버튼을 누른다.

## 9. 버킷 정책 편집에서 우측에 있는 정책 생성기 를 누른다.

## 10. 다음과 같이 입력한다.



## AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to [Amazon Web Services \(AWS\)](#) products. policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Queue Policy](#).

Select Type of Policy S3 Bucket Policy ▼

### Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal   
Use a comma to separate multiple values.

AWS Service Amazon S3 ▼ ☐ All Services ('\*')

Use multiple statements to add permissions for more than one service.

Actions 1 Action(s) Selected ▼ ☐ All Actions ('\*')

Amazon Resource Name (ARN)   
ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}.  
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)


- Select Type of Policy : S3 Bucket Policy
- Effect : Allow
- Principal : \*
- AWS Service : Amazon S3
- Actions : GetObject
- Amazon Resource Name (ARN) : 이전 정책 편집 페이지에서 버킷 ARN 을 복사한다. 추가로 /\* 를 붙여준다. 즉, 다음과 같은 형태가 된다. 버킷 ARN/\*

11. 정책을 생성한 후 나오는 JSON Document를 복사한 뒤 이전 버킷 정책 편집 페이지에 입력한다.

### 버킷 정책

JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다. 자세히 알아보기 [\[링크\]](#)

버킷 ARN

 arn:aws:s3:::final-project-codedeploy-bucket

정책

```
1  {
2    "Version": "2012-10-17",
3    "Id": "Policy1668403726451",
4    "Statement": [
5      {
6        "Sid": "Stmt1668403725649",
7        "Effect": "Allow",
8        "Principal": "*",
9        "Action": "s3:GetObject",
10       "Resource": "arn:aws:s3:::final-project-codedeploy-bucket/*"
11     }
12   ]
13 }
```

## 7.2. CodeDeploy 설정

### 애플리케이션 생성

1. **CodeDeploy** → **애플리케이션** → **애플리케이션 생성**
2. 애플리케이션 이름은 원하는 이름으로 설정한다.
3. 컴퓨팅 플랫폼은 **EC2/온프레미스**로 설정한다.

### IAM 역할 생성

1. **IAM** → **액세스 관리** → **역할** → **역할 만들기**
2. 신뢰할 수 있는 엔티티 유형 : **AWS 서비스**
3. 사용 사례 : 다른 AWS 서비스의 사용 사례 → **CodeDeploy**
4. 권한 추가에서는 바로 **다음** 버튼을 누른다.
5. 권한을 생성한다.

### 배포 그룹 생성



1. CodeDeploy → 애플리케이션 → 애플리케이션 생성 → 생성한 애플리케이션 → 배포 그룹 → 배포 그룹 생성
2. 배포 그룹 이름은 원하는 이름으로 입력한다.
3. 서비스 역할은 위에서 생성한 CodeDeployRole로 설정해준다.
4. 배포 유형은 현재 위치 로 한다.
5. 환경 구성은 Amazon EC2 Auto Scaling 그룹 으로 설정하고 이전에 생성한 오토스케일링 그룹으로 설정한다.
6. 배포 설정은 CodeDeployDefault.AllAtOnce 로 한다.
7. 로드 밸런서는 활성화하고 Application Load Balancer 또는 Network Load Balancer 체크 후 이전에 생성한 대상 그룹을 선택한다.

## 배포 구성 생성

1. CodeDeploy → 배포 구성 → 배포 구성 생성
2. 배포 구성 이름은 원하는 것으로 설정한다.
3. 컴퓨팅 플랫폼은 EC2/온프레미스
4. 최소 정상 호스트는 숫자
5. 값은 1 로 설정한 후 생성한다.

## Jenkins 설정

1. 우선 Jenkins 관리 → 플러그인 관리 → Available plugins 로 들어가서 CodeDeploy 플러그인을 설치한다.
2. 설치 후 새로운 Item 에서 Freestyle project 를 선택하고 OK를 누른다.
3. 소스 코드 관리는 Git 을 체크한다.

- Repository URL : 깃랩의 리포지토리 URL을 입력한다.
  - Credentials : 계정 정보를 입력한다. ID와 비밀번호로 해도 되고, 토큰을 이용하여 인증해도 된다.
  - Branches to build : develop으로 설정한다.
  - 나머지는 그대로 둔다.
4. 빌드 유발에서 **Build when a change is pushed to Gitlab** 을 체크한다.
- Enabled Gitlab triggers : Push Events만 체크한다.
  - 그리고 고급 버튼을 눌러 아래쪽의 Allowed branches를 **Filter branches by name** 으로 체크하고 Include에 **develop** 을 입력한다. 에러메시지가 떠도 무시한다.
  - 맨 아래의 Secret token의 Generate 버튼을 눌러 복사한다.
5. Gitlab 리포지토리 페이지로 이동한다. **Settings** → **Webhooks** 에 가서 URL과 Secret token을 입력한다. URL은 **Build when a change is pushed to Gitlab** 우측에 적혀있다.

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://k7d105.p.ssafy.io:8088/project/final-project-aws-cicd> ?

6. 빌드 환경은 그대로 둔다.
7. Build Steps에서 Execute Shell을 열고 다음과 같은 코드를 입력한다.

```
cd BE/shall-we-meet-then
```

```

cp /var/jenkins_home/properties/application-aws.properties ${WORKSPACE}/BE/shall-we-meet-then/src/main/resources/application-aws.properties

cp /var/jenkins_home/properties/application-mail.properties ${WORKSPACE}/BE/shall-we-meet-then/src/main/resources/application-mail.properties

chmod +x gradlew

./gradlew clean build -x test

cp build/libs/*.jar /var/jenkins_home/workspace/final-project-aws-cicd
cp appspec.yml /var/jenkins_home/workspace/final-project-aws-cicd
cp scripts -r /var/jenkins_home/workspace/final-project-aws-cicd

```

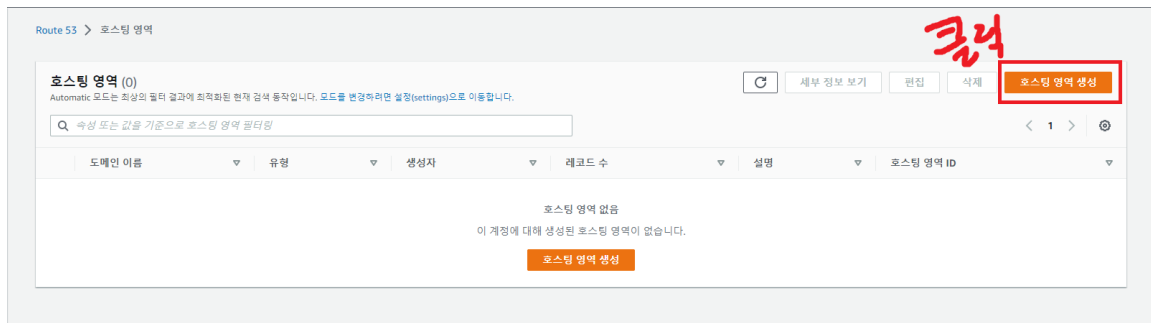
8. 빌드 후 조치에서 **Deploy an application to AWS CodeDeploy** 를 선택한다.

- AWS CodeDeploy Application Name : 위에서 생성한 CodeDeploy 애플리케이션의 이름을 입력한다.
- AWS CodeDeploy Deployment Group : 위에서 생성한 배포 그룹의 이름을 입력한다.
- AWS CodeDeploy Deployment Config : 위에서 생성한 배포 구성의 이름을 입력한다.
- AWS Region : **AP\_NORTHEAST\_2** 를 입력한다.
- S3 Bucket : 위에서 생성한 S3 버킷의 이름을 입력한다.
- Include Files : **\*.jar, appspec.yml, scripts/\*** 을 입력한다.
- Use Access/Secret Keys는 다음과 같은 권한을 가지고 있는 IAM 사용자를 생성해 해당 사용자의 Access Key와 Secret Key를 입력해준다.

## 8. ACM과 Route 53을 활용한 도메인 네임 설정

### 8.1. 가비아 → Route 53 호스팅 설정

1. AWS Route 53으로 이동한다.
2. 왼쪽 탭에서 **호스팅 영역** 을 클릭한 후 **호스팅 영역 생성** 버튼을 클릭한다.



### 3. 각 입력창에 값을 입력한다.

#### 호스팅 영역 구성

호스팅 영역은 example.com 같은 도메인과 관련 하위 도메인에 대한 트래픽을 라우팅하는 방식에 대한 정보를 포함하는 컨테이너입니다.

도메인 이름

정보

트래픽을 라우팅할 도메인의 이름입니다.

유효한 문자: a-z, 0-9 및 ! " # \$ % & ' ( ) \* + , - / : ; < = > ? @ [ \ ] ^ \_ ` { | } . ~

설명 - 선택 사항

정보

이 값을 사용하면 이름이 동일한 호스팅 영역을 구별할 수 있습니다.

설명은 최대 256자입니다. 0/256

유형

정보

유형은 인터넷 또는 Amazon VPC에서 트래픽을 라우팅할지 여부를 가리킵니다.

☒ 퍼블릭 호스팅 영역
 퍼블릭 호스팅 영역은 인터넷에서 트래픽을 라우팅하는 방식을 결정합니다.

☐ 프라이빗 호스팅 영역
 프라이빗 호스팅 영역은 Amazon VPC 내에서 트래픽을 라우팅하는 방식을 결정합니다.

#### 태그 정보

호스팅 영역에 태그를 적용하면 호스팅 영역을 쉽게 구성하고 식별할 수 있습니다.

리소스와 연결된 태그가 없습니다.

태그 추가

최대 50개의 태그를 더 추가할 수 있습니다.

취소

호스팅 영역 생성

- **도메인 이름** : 가비아에서 구매한 도메인 이름을 입력한다.
- **설명** : 해당 도메인에 대한 설명을 입력한다. (입력 안해도 됨)
- **유형** : 퍼블릭 호스팅 영역을 선택한다.

4. Route 53 호스팅 영역이 생성 완료되었다. **NS** 유형의 값/트래픽 라우팅 대상을 기억한다.

Route 53 > 호스팅 영역 > shallwemeetthen.com

**퍼블릭 shallwemeetthen.com** 정보 영역 삭제 레코드 테스트 쿼리 로깅 구성

▶ 호스팅 영역 세부 정보 호스팅 영역 편집

**레코드(2)** | DNSSEC 서명 | 호스팅 영역 태그(0)

**레코드 (2) 정보**  
Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. 모드를 변경하려면 설정(settings)으로 이동합니다.

↺ 레코드 삭제 영역 파일 가져오기 레코드 생성

🔍 속성 또는 값을 기준으로 레코드 필터링 유형 ▼ 라우팅 정책 ▼ 별칭 ▼ < 1 > ⚙️

<input type="checkbox"/>	레코드 이름	유형	라우팅 ...	차별...	값/트래픽 라우팅 대상
<input type="checkbox"/>	shallwemeetthen.com	NS	단순	-	ns-1147.awsdns-15.org. ns-160.awsdns-20.com. ns-543.awsdns-03.net. ns-1805.awsdns-33.co.uk.
<input type="checkbox"/>	shallwemeetthen.com	SOA	단순	-	ns-1147.awsdns-15.org. awsdns-hostmaster.amazon.com. 1 7200 90...

5. 가비아로 이동해서 **가비아** > **My가비아** > **도메인통합관리** > **도메인관리** > **도메인상세페이지** 로 이동한다. **네임서버**의 설정 버튼을 누르고, 네임서버의 호스트명을 Route 53의 값/트래픽 라우팅 대상 값으로 전부 변경해준다.

홈 > 전제도메인 > 도메인 상세 페이지 메뉴열

< shallwemeetthen.com 등록 확인증 인증 코드

등록일: 2022-11-03 만기일: 2023-11-03 (남은 기간: 354일) 만기 전 삭제 만기일 맞춤 도메인 연장 연장 알림

소유자	수정	소유권 이전	관리자	수정
이명범 mungmnb777@gmail.com 010-4568-9094			이명범 mungmnb777@gmail.com 010-4568-9094	

**네임서버** 설정

1차	ns.gabia.co.kr	8차	데이터 없음
2차	ns1.gabia.co.kr	9차	데이터 없음
3차	ns.gabia.net	10차	데이터 없음
4차	데이터 없음	11차	데이터 없음
5차	데이터 없음	12차	데이터 없음
6차	데이터 없음	13차	데이터 없음
7차	데이터 없음		

6. 네임 서버 등록이 완료되었다.

## 8.2. ACM을 활용한 SSL 인증서 발급

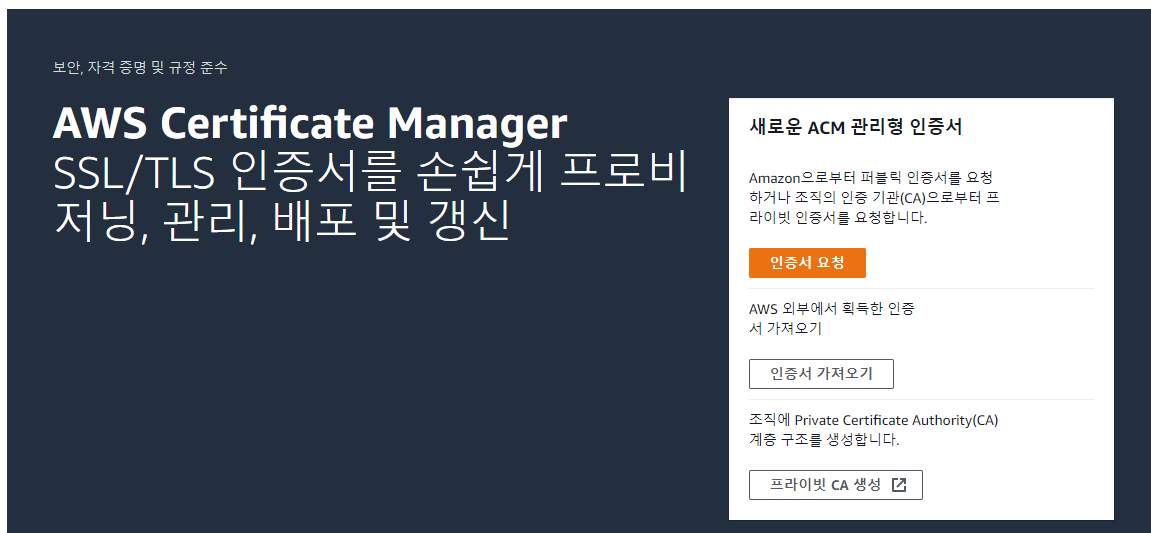
### ACM이란?

ACM이란 AWS Certificate Manager로 웹 사이트에서 사용하는 SSL/TLS 인증서를 발급 및 관리해주는 서비스이다. SSL은 클라이언트와 서버 간 구간 암호화를 적용해주는 기술이

다. 웹사이트에서 나가고 들어오는 데이터를 암호화하는 기능을 가진 보안 인증서로 클라이언트와 서버 간의 통신을 제 3자가 보증해주는 전자 문서다. https를 사용하기 위해서는 SSL 인증이 필수이다.

## ACM으로 SSL 인증서 요청 방법

1. AWS ACM 접속 후 **인증서 요청** 버튼을 클릭한다.



2. **퍼블릭 인증서** 를 선택 후 **다음** 을 클릭한다.



3. 도메인 이름을 적은 후 DNS 검증을 선택한다. 키 알고리즘은 **RSA 2048** 로 유지한다.

AWS Certificate Manager > 인증서 > 인증서 요청 > 퍼블릭 인증서 요청

## 퍼블릭 인증서 요청

**도메인 이름**  
인증서에 대해 하나 이상의 도메인 이름을 제공합니다.

완전히 정규화된 도메인 이름 **정보**

**이 인증서에 다른 이름 추가**

이 인증서에 이름을 추가할 수 있습니다. 예를 들어, 'www.example.com'에 대한 인증서를 요청하는 경우 고객이 두 이름 중 하나로 사이트에 접속할 수 있도록 'example.com'이라는 이름을 추가할 수 있습니다.

**검증 방법 정보**  
도메인 소유권을 검증하기 위한 방법 선택

☒ **DNS 검증 - 권장**  
인증서 요청에서 도메인에 대한 DNS 구성을 수정할 권한이 있는 경우 이 옵션을 선택합니다.

☐ **이메일 검증**  
인증서 요청에서 도메인에 대한 DNS 구성을 수정할 권한을 소유하지 않거나 확인할 수 없는 경우 이 옵션을 선택합니다.

4. 생성 후 인증서로 들어가보면 **CNAME** 정보를 확인할 수 있다. **Route 53**에서 **레코드 생성** 을 클릭한다.

AWS Certificate Manager > 인증서 > 64e7fd5e-174b-4e2a-9896-6232385bd76c

64e7fd5e-174b-4e2a-9896-6232385bd76c 삭제

**인증서 상태**

식별자: 64e7fd5e-174b-4e2a-9896-6232385bd76c  
ARN: arn:aws:acm:ap-northeast-2:458527042685:certificate/64e7fd5e-174b-4e2a-9896-6232385bd76c  
유형: Amazon 발급

상태: 검증 대기 중 **정보**

**도메인 (1)** Route 53에서 레코드 생성 CSV로 내보내기

도메인	상태	경신 상태	유형	CNAME 이름	CNAME 값
*.shallwemeetthen.com	<span>검증 대기 중</span>	-	CNAME	<input type="text" value="secret"/>	<input type="text" value="secret"/>

5. 레코드를 생성한다.

AWS Certificate Manager > 인증서 > Amazon Route 53에서 DNS 레코드 생성

### Amazon Route 53에서 DNS 레코드 생성 (1/1)

1 일치 < 1 >

<input checked="" type="checkbox"/>	도메인	검증 상태	유형	CNAME 이름	CNAME 값	도메인이 Route 53에 있습니까?
<input checked="" type="checkbox"/>	*.shallwemeetthen.com	<span>검증 대기 중</span>	CNAME	<input type="text" value="비밀"/>	<input type="text" value="비밀"/>	예

취소 레코드 생성

6. 가비아로 접속해준다. **가비아** > **My가비아** > **도메인통합관리** > **도메인관리** > **DNS 정보**에 들어가서 **DNS 관리**를 클릭해준다.

**gabia. 도메인 관리**

전체 도메인 01개

홈 > DNS 정보

> 전체 도메인

> 도메인 정보 변경

> **DNS 정보**

> 도메인 보안

> 예약 도메인 관리

> 관심 도메인

> Adultblock

**DNS 정보**

DNS 정보는 도메인에 연결된 서비스를 확인할 수 있습니다.  
DNS 설정, 도메인 연결, 포워딩, 파킹 서비스 등의 신청 및 설정은 'DNS 관리'에서 가능합니다.

[DNS 관리](#)

번호	도메인명 > 전체 >	DNS 정보
1	shallwemeetthen.com	A

7. **DNS 설정**의 **레코드 수정**을 클릭한다. **레코드 추가** 후 값을 입력하고 저장한다.

DNS 관리

sh DNS 레코드 수정

이력 확인 [역설 다운로드](#)

타입 >	호스트	값/위치	TTL	우선 순위	서비스 >	상태
등록된 레코드가 없습니다.						

[+ 레코드 추가](#) [저장](#) [취소](#)

웹 파킹 [설정](#)

모바일 파킹 [설정](#)

- **타입** : **CNAME**으로 입력한다.
- **호스트** : **aws**로 입력한다.
- **값/위치** : 위에서 생성한 AWS **레코드**의 CNAME 값을 입력해준다.
- **TTL** : 600으로 그대로 둔다.
- **상태** : **확인** 버튼을 누른다.

8. 모든 과정을 끝나치고 10~20분을 기다리면 인증서의 상태가 **발급됨**으로 변한다.

AWS Certificate Manager > 인증서

인증서 (1)

[새](#) [만료 이벤트 관리](#) [가져오기](#) [요청](#)

인증서 ID	도메인 이름	유형	상태	사용 중	경신 자격
64e7fd5e-174b-4e2a-9896-6232385bd76c	*.shallwemeetthen.com	Amazon 발급	<b>발급됨</b>	아니요	부적격

## AWS 로드밸런서에 인증서 적용



1. EC2 → 로드밸런서 → 적용할 로드밸런서 선택 → 리스너 로 이동한다. 그리고 리스너 추가 버튼을 누른다.
2. 리스너를 생성한다.

## Add listener

### Listener details

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Protocol
Port

HTTPS
:
443
1-65535

Default actions [Info](#)

Specify the default actions for traffic on this listener. Default actions apply to traffic that does not meet the conditions of rules on your listener. Rules can be configured after the listener is created.

Add action ▼

### Secure listener settings [Info](#)

Security policy

Your load balancer uses a Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL connections with clients.

ELBSecurityPolicy-2016-08 ▼

[Compare security policies](#)

Default SSL/TLS certificate

The certificate used if a client connects without SNI protocol, or if there are no matching certificates. This certificate will automatically be added to your listener certificate list.

From ACM ▼

\*.shallwemeetthen.com  
64e7fd5e-174b-4e2a-9896-6232385bd76c
▼

↻

[Request new ACM certificate](#)

- Protocol : HTTPS
  - Port : 443
  - Default actions : Forward to 를 클릭한다. 그리고 Target Group 은 로드밸런서가 적용되는 타겟 그룹을 선택한다.
  - Default SSL/TLS certificate : From ACM 을 선택한 후 위에서 생성한 인증서를 선택한다.
3. 다시 Route 53 으로 돌아간다. 호스팅 영역 → 레코드 생성 을 클릭한다.

## 빠른 레코드 생성 정보

Quick create record

마법사로 전환

▼ 레코드 1

삭제

레코드 이름 정보

레코드 유형 정보

A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅 ▼

루트 도메인에 대한 레코드를 생성하려면 비워 둡니다.

☒ 별칭

트래픽 라우팅 대상 정보

Application/Classic Load Balancer에 대한 별칭 ▼

아시아 태평양(서울) [ap-northeast-2] ▼

Q 로드 밸런서 선택

라우팅 정책 정보

대상 상태 평가

단순 라우팅 ▼

☒ 예

다른 레코드 추가

취소

레코드 생성

- **레코드 이름** : 원하는 subdomain을 입력한다.
- **레코드 유형** : A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅
- **별칭** 체크 박스를 켜준다.
  - **트래픽 라우팅 대상** : Application/Classic Load Balancer에 대한 별칭
  - **지역** : 아시아 태평양(서울)
  - **로드 밸런서** : 라우팅할 로드밸런서를 선택한다.